ACL HLT 2011

The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies

Proceedings of the Conference

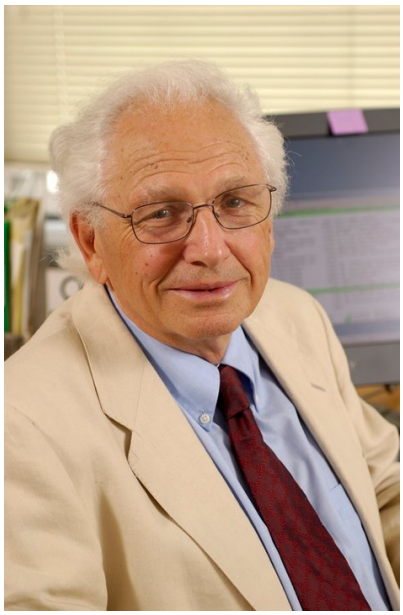
19-24 June, 2011
Portland, Oregon, USA

Production and Manufacturing by
Omnipress, Inc.
2600 Anderson Street
Madison, WI 53704 USA

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-932432-87-9



We dedicate the ACL 2011 proceedings to the memory of Fred Jelinek (1932-2010), who received ACL's Lifetime Achievement Award in 2009. His award acceptance speech can be found in Computational Linguistics 35(4), and an obituary by Mark Liberman appeared in Computational Linguistics 36(4). Several other newspaper and professional society obituaries have described his extraordinary personal life and career.

Fred's influence on computational linguistics is almost impossible to overstate. In the 1970s and 1980s, he and his colleagues at IBM developed the statistical paradigm that dominates our field today, including a great many specific techniques for modeling, parameter estimation, and search that continue to enjoy wide use. Even more fundamentally, as Mark Liberman recounts in his obituary, Fred led the field away from a mode where lone inventors defended their designs by appealing to aesthetics and anecdotes, to a more communal and transparent process of evaluating methods objectively through controlled comparisons on training and test sets.

Under Fred's visionary leadership, the IBM group revolutionized speech recognition by adopting a statistical, data-driven perspective that was deeply at odds with the rationalist ethos of the time. The group began with Fred's information-theoretic reconceptualization of the task as recovering a source signal (text) after it had passed through a noisy channel. They then worked out the many components needed for a full speech recognizer, along with the training algorithms for each component and global decoding algorithms. Steve Young, in an obituary in the IEEE SLTC Newsletter, describes Fred as not a pioneer but the pioneer of speech recognition.

In the 1980s, the IBM speech group's work on language modeling drew them toward deeper analysis of text. Fred and his colleagues introduced NLP methods such as word clustering, HMM part-of-speech tagging, history-based parsing, and prefix probability computation in PCFGs. They famously turned their noisy-channel lens on machine translation, founding the field of statistical MT with a series of ingenious and highly influential models.

After Fred moved to Johns Hopkins University in 1993, he worked tirelessly to improve language modeling by incorporating syntactic and other long-range dependencies as well as semantic classes. He also presided for 16 years over the Johns Hopkins Summer Workshops, whose 51 teams from 1995-2010 attacked a wide range of topics in human language technology, many making groundbreaking advances in the field.

There is a popular conception that Fred was somehow hostile to linguistics. Certainly he liked to entertain others by repeating his 1988 quip that "Any time a linguist leaves the group, the recognition rate goes up." Yet he had tried to leave information theory for linguistics as early as 1962, influenced by Noam Chomsky's lectures and his wife Milena's earlier studies with Roman Jakobson. He always strove for clean formal models just as linguists do. He was deeply welcoming toward any attempt to improve models through better linguistics, as long as they had a large number of parameters. Indeed, it was one of the major frustrations of his career that it was so difficult to beat n-gram language models, when humans were evidently using additional linguistic and world knowledge to obtain much better predictive performance. As he said in an award acceptance speech in 2004, "My colleagues and I always hoped that linguistics will eventually allow us to strike gold."

Fred was skeptical only about the relevance of armchair linguistics to engineering, believing that there was far more variation in the data than could be described compactly by humans. For this reason, while he was quite interested in recovering or exploiting latent linguistic structure, he trusted human-annotated linguistic data to be a better description of that structure than human-conceived linguistic rules. Statistical models could be aided even by imperfect or incomplete annotations, such as unaligned orthographic transcriptions, bilingual corpora, or syntactic analyses furnished by ordinary speakers. Fred pushed successfully for the development of such resources, notably the IBM/Lancaster Treebank and its successor, the Penn Treebank.

Fred influenced many of us personally. He was warm-hearted, witty, cultured, thoughtful about the scientific process, a generous mentor, and always frank, honest, and unpretentious. The changes that he brought to our field are largely responsible for its recent empirical progress and commercial success. They have also helped make it attractive to many bright, technically sophisticated young researchers. This proceedings volume, which is dedicated to his memory, testifies to the overwhelming success of his leadership and vision.

By Jason Eisner, on behalf of ACL 2011 Organizing Committee

Preface: General Chair

Welcome to the 49th Annual Meeting of the Association for Computational Linguistics in Portland, Oregon. ACL is perhaps the longest-running conference series in computer science. Amazingly, it is still growing. We expect this year's ACL to attract an even larger number of participants than usual, since 2011 happens to be an off-year for COLING, EACL and NAACL.

The yearly success of ACL results from the dedication and hard work of many people. This year is no exception. I would like to thank all of them for volunteering their time and energy in service to our community.

I thank the Program Co-Chairs Rada Mihalcea and Yuji Matsumoto for putting together a wonderful main conference program, including 164 long papers, 128 short papers and much anticipated keynote speeches by David Ferrucci and Lera Boroditsky. Tutorial Co-Chairs, Patrick Pantel and Andy Way solicited proposals and selected six fascinating tutorials in a wide range of topics. The Workshop Co-Chairs, Hal Daume III and John Carroll, organized a joint selection process with EMNLP 2011. The program consists of 3 two-day workshops and 13 one-day workshops, a new record number for ACL. Sadao Kurohashi, Chair of System Demonstrations, assembled a committee and oversaw the review of 46 demo submissions.

The Student Session is organized by Co-Chairs, Sasa Petrovic, Emily Pitler, Ethan Selfridge and Faculty Advisors: Miles Osborne, Thamar Solorio. They introduced a new, poster-only format to be held in conjunction with the main ACL poster session. They also obtained NSF funding to provide travel support for all student session authors.

Special thank goes to Publication Chair, Guodong Zhou and his assistant Hong Yu. They produced the entire proceedings of the conference.

We are indebted to Brain Roark and the local arrangement committee for undertaking a phenomenal amount detailed work over the course of two years to host this conference, such as allocating appropriate space to meet all the needs of the scientific program, compiling and printing of the conference handbook, arranging a live tango band for the banquet and dance, to name just a few. The local arrangement committee consists of: Nate Bodenstab (webmeister), Peter Heeman (exhibitions), Christian Monson (student volunteers), Zak Shafran and Meg Mitchell (social), Richard Sproat (local sponsorship), Mahsa Yarmohammadi and Masoud Rouhizadeh (student housing coordinators) and Aaron Dunlop (local publications coordinator).

I want to express my gratitude to Ido Dagan, Chair of the ACL Conference Coordination Committee, Dragomir Radev, ACL Secretary, and Priscilla Rasmussen, ACL Business Manager, for their advice and guidance throughout the process.

ACL 2011 has two Platinum Sponsors (Google and Baidu), one Gold Sponsor (Microsoft), two Silver sponsors (Pacific Northwest National Lab and Yahoo!), and seven Bronze Sponsors and six Supporters. We are grateful for the financial support from these organizations. I would like to thank and applaud the tremendous effort by the ACL sponsorship committee: Srinivas Bangalore (AT&T), Massimiliano Ciaramita (Google), Kevin Duh (NTT), Michael Gamon (Microsoft), Stephen Pulman (Oxford), Priscilla Rasmussen (ACL), and Haifeng Wang (Baidu).

Finally, I would like to thank all the area chairs, workshop organizers, tutorial presenters, authors, reviewers and conference attendees for their participation and contribution. I hope everyone will have a great time sharing ideas and inspiring one another at this conference.

ACL 2011 General Chair
Dekang Lin, Google, Inc.

Preface: Program Committee Co-Chairs

Welcome to the program of the 2011 Conference of the Association for Computational Linguistics! ACL continues to grow, and this year the number of paper submissions broke once again the record set by previous years. We received a total of 1,146 papers, out of which 634 were submitted as long papers and 512 were submitted as short papers. 25.7

To achieve the goal of a broad technical program, we followed the initiative from last year and solicited papers under four main different categories: *theoretical computational linguistics*, *empirical/data-driven approaches*, *resources/evaluation*, and *applications/tools*. We also continued to accept other types of papers (e.g., surveys or challenge papers), although unlike the previous year, no separate category was created for these papers. The papers falling under one of the four categories were reviewed using specialized reviewed forms; we also had a general review form that was used to review the papers that did not fall under one of the four main categories.

A new initiative this year was to also accept papers accompanied by supplemental materials (software and/or datasets). In addition to the regular review of the research quality of the paper, the accompanied resources were also reviewed for their quality, and the acceptance or rejection decisions were made based on the quality of both the paper and the supplemental materials. Among all the submissions, a total of 84 papers were accompanied by a software package and 117 papers were accompanied by a dataset. Among all the accepted papers, 30 papers are accompanied by software and 35 papers are accompanied by a dataset. These materials will be hosted on the ACL web site under http://www.aclweb.org/supplementals.

We are delighted to have two distinguished invited speakers: Dr. David Ferrucci (Principal Investigator, IBM Research), who will talk about his team's work on building *Watson* – a deep question answering system that achieved champion-level performance at Jeopardy!, and Lera Boroditsky (Assistant Professor, Stanford University), who will give a presentation on her research on how the languages we speak shape the way we think. In addition, the recipient of the ACL Lifetime Achievement Award will present a plenary lecture during the final day of the conference.

As in previous years, there will be three awards, one for the best long paper, one for the best long paper by a student, and one for the best short paper. The candidates for the best paper awards were nominated by the area chairs, who took into consideration the feedback they received from the reviewers on whether a paper might merit a best paper prize. From among the nominations we received, we selected the top five candidates for the long and short papers, and the final awards were then selected by the area chairs together with the program co-chairs. The recipients of the best paper awards will present their papers in a plenary session during the second day of the conference.

There are many individuals to thank for their contributions to the conference program. First and foremost, we would like to thank the authors who submitted their work to ACL. The growing number of submissions reflects how broad and active our field is. We are deeply indebted to the area chairs and the reviewers for their hard work. They enabled us to select an exciting program and to provide valuable feedback to the authors. We thank the general conference chair Dekang Lin and the local arrangements committee headed by Brian Roark for their help and advice, as well as last year's program committee co-chairs, Stephen Clark and Sandra Carberry, for sharing their experiences. Additional thanks go to

the publications chair, Guodong Zhang, who put this volume together, and Yu Hong, who helped him with this task.

We are most grateful to Priscilla Rasmussen, who helped us with various logistic and organizational aspects of the conference. Rich Gerber and the START team responded to our questions quickly, and helped us manage the large number of submissions smoothly.

Enjoy the conference!

ACL 2011 Program Co-Chairs
Yuji Matsumoto, Nara Institute of Science and Technology
Rada Mihalcea, University of North Texas

Organizing Committee

General Chair

Dekang Lin, Google

Local Arrangements Chair

Brian Roark, Oregon Health & Science University

Program Co-Chairs

Yuji Matsumoto, Nara Institute of Science and Technology
Rada Mihalcea, University of North Texas

Local Arrangements Committee

Nate Bodenstab, Oregon Health & Science University
Aaron Dunlop, Oregon Health & Science University
Peter Heeman, Oregon Health & Science University
Meg Mitchell, Oregon Health & Science University
Christian Monson, Nuance
Zak Shafran, Oregon Health & Science University
Richard Sproat, Oregon Health & Science University
Masoud Rouhizadeh, Oregon Health & Science University
Mahsa Yarmohammadi, Oregon Health & Science University

Publications Chair

Guodong Zhou, Suzhou University

Sponsorship Chairs

Haifeng Wang, Baidu
Kevin Duh, National Inst. of Information and Communications Technology
Massimiliano Ciaramita, Google
Michael Gamon, Microsoft
Priscilla Rasmussen, Association for Computational Linguistics
Srinivas Bangalore, AT&T
Stephen Pulman, Oxford University

Tutorial Co-chairs

Patrick Pantel, Microsoft Research
Andy Way, Dublin City University

Workshop Co-chairs

Hal Daume III, University of Maryland
John Carroll, University of Sussex

Demo Chair

Sadao Kurohashi, Kyoto University

Mentoring

Chair
- Tim Baldwin, University of Melbourne

Committee
- Chris Biemann, TU Darmstadt
- Mark Dras, Macquarie University
- Jeremy Nicholson, University of Melbourne

Student Research Workshop

Student Co-chairs
- Sasa Petrovic, University of Edinburgh
- Emily Pitler, University of Pennsylvania
- Ethan Selfridge, Oregon Health & Science University

Faculty Advisors
- Miles Osborne, University of Edinburgh
- Thamar Solorio, University of Alabama at Birmingham

ACL Conference Coordination Committee

Ido Dagan, Bar Ilan University (chair)
Chris Brew, Ohio State University
Graeme Hirst, University of Toronto
Lori Levin, Carnegie Mellon University
Christopher Manning, Stanford University
Dragomir Radev, University of Michigan
Owen Rambow, Columbia University
Priscilla Rasmussen, Association for Computational Linguistics
Suzanne Stevenson, University of Toronto

ACL Business Manager

Priscilla Rasmussen, Association for Computational Linguistics

Program Committee

Program Co-chairs

Yuji Matsumoto, Nara Institute of Science and Technology
Rada Mihalcea, University of North Texas

Area Chairs

Razvan Bunescu, Ohio University
Xavier Carreras, Technical University of Catalonia
Anna Feldman, Montclair University
Pascale Fung, Hong Kong University of Science and Technology
Chu-Ren Huang, Hong Kong Polytechnic University
Kentaro Inui, Tohoku University
Greg Kondrak, University of Alberta
Shankar Kumar, Google
Yang Liu, University of Texas at Dallas
Bernardo Magnini, Fondazione Bruno Kessler
Elliott Macklovitch, Marque d'Or
Katja Markert, University of Leeds
Lluis Marquez, Technical University of Catalonia
Diana McCarthy, Lexical Computing Ltd
Ryan McDonald, Google
Alessandro Moschitti, University of Trento
Vivi Nastase, Heidelberg Institute for Theoretical Studies
Manabu Okumura, Tokyo Institute of Technology
Vasile Rus, University of Memphis
Fabrizio Sebastiani, National Research Council of Italy
Michel Simard, National Research Council of Canada
Thamar Solorio, University of Alabama at Birmingham
Svetlana Stoyanchev, Open University
Carlo Strapparava, Fondazione Bruno Kessler
Dan Tufis, Romanian Academy of Artificial Intelligence
Xiaojun Wan, Peking University
Taro Watanabe, National Inst. of Information and Communications Technology
Alexander Yates, Temple University
Deniz Yuret, Koc University

Program Committee

Ahmed Abbasi, Eugene Agichtein, Eneko Agirre, Lars Ahrenberg, Gregory Aist, Enrique Alfonseca, Laura Alonso i Alemany, Gianni Amati, Alina Andreevskaia, Ion Androutsopoulos, Abhishek Arun, Masayuki Asahara, Nicholas Asher, Giuseppe Attardi, Necip Fazil Ayan

Collin Baker, Jason Baldridge, Tim Baldwin, Krisztian Balog, Carmen Banea, Verginica Barbu

Mititelu, Marco Baroni, Regina Barzilay, Roberto Basili, John Bateman, Tilman Becker, Lee Becker, Beata Beigman-Klebanov, Cosmin Bejan, Ron Bekkerman, Daisuke Bekki, Kedar Bellare, Anja Belz, Sabine Bergler, Shane Bergsma, Raffaella Bernardi, Nicola Bertoldi, Pushpak Bhattacharyya, Archana Bhattarai, Tim Bickmore, Chris Biemann, Dan Bikel, Alexandra Birch, Maria Biryukov, Alan Black, Roi Blanco, John Blitzer, Phil Blunsom, Gemma Boleda, Francis Bond, Kalina Bontcheva, Johan Bos, Gosse Bouma, Kristy Boyer, S.R.K. Branavan, Thorsten Brants, Eric Breck, Ulf Brefeld, Chris Brew, Ted Briscoe, Samuel Brody

Michael Cafarella, Aoife Cahill, Chris Callison-Burch, Rafael Calvo, Nicoletta Calzolari, Nicola Cancedda, Claire Cardie, Giuseppe Carenini, Claudio Carpineto, Marine Carpuat, Xavier Carreras, John Carroll, Ben Carterette, Francisco Casacuberta, Helena Caseli, Julio Castillo, Mauro Cettolo, Hakan Ceylan, Joyce Chai, Pi-Chuan Chang, Vinay Chaudhri, Berlin Chen, Ying Chen, Hsin-Hsi Chen, John Chen, Colin Cherry, David Chiang, Yejin Choi, Jennifer Chu-Carroll, Grace Chung, Kenneth Church, Massimiliano Ciaramita, Philipp Cimiano, Stephen Clark, Shay Cohen, Trevor Cohn, Nigel Collier, Michael Collins, John Conroy, Paul Cook, Ann Copestake, Bonaventura Coppola, Fabrizio Costa, Koby Crammer, Dan Cristea, Montse Cuadros, Silviu-Petru Cucerzan, Aron Culotta, James Curran

Walter Daelemans, Robert Damper, Hoa Dang, Dipanjan Das, Hal Daume, Adria de Gispert, Marie-Catherine de Marneffe, Gerard de Melo, Maarten de Rijke, Vera Demberg, Steve DeNeefe, John DeNero, Pascal Denis, Ann Devitt, Giuseppe Di Fabbrizio, Mona Diab, Markus Dickinson, Mike Dillinger, Bill Dolan, Doug Downey, Markus Dreyer, Greg Druck, Kevin Duh, Chris Dyer, Marc Dymetman

Markus Egg, Koji Eguchi, Andreas Eisele, Jacob Eisenstein, Jason Eisner, Michael Elhadad, Tomaz Erjavec, Katrin Erk, Hugo Escalante, Andrea Esuli

Hui Fang, Alex Chengyu Fang, Benoit Favre, Anna Feldman, Christiane Fellbaum, Donghui Feng, Raquel Fernandez, Nicola Ferro, Katja Filippova, Jenny Finkel, Seeger Fisher, Margaret Fleck, Dan Flickinger, Corina Forascu, Kate Forbes-Riley, Mikel L. Forcada, Eric Fosler-Lussier, Jennifer Foster, George Foster, Anette Frank, Alex Fraser, Dayne Freitag, Guohong Fu, Hagen Fuerstenau, Pascale Fung, Sadaoki Furui

Evgeniy Gabrilovich, Robert Gaizauskas, Michel Galley, Michael Gamon, Kuzman Ganchev, Jianfeng Gao, Claire Gardent, Thomas Gärtner, Albert Gatt, Dmitriy Genzel, Kallirroi Georgila, Carlo Geraci, Pablo Gervas, Shlomo Geva, Daniel Gildea, Alastair Gill, Dan Gillick, Jesus Gimenez, Kevin Gimpel, Roxana Girju, Claudio Giuliano, Amir Globerson, Yoav Goldberg, Sharon Goldwater, Carlos Gomez Rodriguez, Julio Gonzalo, Brigitte Grau, Stephan Greene, Ralph Grishman, Tunga Gungor, Zhou GuoDong, Iryna Gurevych, David Guthrie

Nizar Habash, Ben Hachey, Barry Haddow, Gholamreza Haffari, Aria Haghighi, Udo Hahn, Jan Hajic, Dilek Hakkani-Tür, Keith Hall, Jirka Hana, John Hansen, Sanda Harabagiu, Mark Hasegawa-Johnson, Koiti Hasida, Ahmed Hassan, Katsuhiko Hayashi, Ben He, Xiaodong He, Ulrich Heid, Michael Heilman, Ilana Heintz, Jeff Heinz, John Henderson, James Henderson, Iris

Hendrickx, Aurelie Herbelot, Erhard Hinrichs, Tsutomu Hirao, Julia Hirschberg, Graeme Hirst, Julia Hockenmaier, Tracy Holloway King, Bo-June (Paul) Hsu, Xuanjing Huang, Liang Huang, Jimmy Huang, Jian Huang, Chu-Ren Huang, Juan Huerta, Rebecca Hwa

Nancy Ide, Gonzalo Iglesias, Gabriel Infante-López, Diana Inkpen, Radu Ion, Elena Irimia, Pierre Isabelle, Mitsuru Ishizuka, Aminul Islam, Abe Ittycheriah, Tomoharu Iwata

Martin Jansche, Sittichai Jiampojamarn, Jing Jiang, Valentin Jijkoun, Richard Johansson, Mark Johnson, Aravind Joshi

Nanda Kambhatla, Min-Yen Kan, Kyoko Kanzaki, Rohit Kate, Junichi Kazama, Bill Keller, Andre Kempe, Philipp Keohn, Fazel Keshtkar, Adam Kilgarriff, Jin-Dong Kim, Su Nam Kim, Brian Kingsbury, Katrin Kirchhoff, Ioannis Klapaftis, Dan Klein, Alexandre Klementiev, Kevin Knight, Rob Koeling, Oskar Kohonen, Alexander Kolcz, Alexander Koller, Kazunori Komatani, Terry Koo, Moshe Koppel, Valia Kordoni, Anna Korhonen, Andras Kornai, Zornitsa Kozareva, Lun-Wei Ku, Sandra Kuebler, Marco Kuhlmann, Roland Kuhn, Mikko Kurimo, Oren Kurland, Olivia Kwong

Krista Lagus, Philippe Langlais, Guy Lapalme, Mirella Lapata, Dominique Laurent, Alberto Lavelli, Matthew Lease, Gary Lee, Kiyong Lee, Els Lefever, Alessandro Lenci, James Lester, Gina-Anne Levow, Tao Li, Shoushan LI, Fangtao Li, Zhifei Li, Haizhou Li, Hang Li, Wenjie Li, Percy Liang, Chin-Yew Lin, Frank Lin, Mihai Lintean, Ken Litkowski, Diane Litman, Marina Litvak, Yang Liu, Bing Liu, Qun Liu, Jingjing Liu, Elena Lloret, Birte Loenneker-Rodman, Adam Lopez, Annie Louis, Xiaofei Lu, Yue Lu

Tengfei Ma, Wolfgang Macherey, Klaus Macherey, Elliott Macklovitch, Nitin Madnani, Bernardo Magnini, Suresh Manandhar, Gideon Mann, Chris Manning, Daniel Marcu, David Martínez, Andre Martins, Yuval Marton, Sameer Maskey, Spyros Matsoukas, Mausam, Arne Mauser, Jon May, David McAllester, Andrew McCallum, David McClosky, Ryan McDonald, Bridget McInnes, Tara McIntosh, Kathleen McKeown, Paul McNamee, Yashar Mehdad, Qiaozhu Mei, Arul Menezes, Paola Merlo, Donald Metzler, Adam Meyers, Haitao Mi, Jeff Mielke, Einat Minkov, Yusuke Miyao, Dunja Mladenic, Marie-Francine Moens, Saif Mohammad, Dan Moldovan, Diego Molla, Christian Monson, Manuel Montes y Gomez, Raymond Mooney, Robert Moore, Tatsunori Mori, Glyn Morrill, Sara Morrissey, Alessandro Moschitti, Jack Mostow, Smaranda Muresan, Gabriel Murray, Gabriele Musillo, Sung-Hyon Myaeng

Tetsuji Nakagawa, Mikio Nakano, Preslav Nakov, Ramesh Nallapati, Vivi Nastase, Borja Navarro-Colorado, Roberto Navigli, Mark-Jan Nederhof, Matteo Negri, Ani Nenkova, Graham Neubig, Guenter Neumann, Vincent Ng, Hwee Tou Ng, Patrick Nguyen, Jian-Yun Nie, Rodney Nielsen, Joakim Nivre, Tadashi Nomoto, Scott Nowson

Diarmuid Ó Séaghdha, Sharon O'Brien, Franz Och, Stephan Oepen, Kemal Oflazer, Jong-Hoon Oh, Constantin Orasan, Miles Osborne, Gozde Ozbal

Sebastian Pado, Tim Paek, Bo Pang, Patrick Pantel, Soo-Min Pantel, Ivandre Paraboni, Cecile Paris, Marius Pasca, Gabriella Pasi, Andrea Passerini, Rebecca J. Passonneau, Siddharth Patwardhan, Adam Pauls, Adam Pease, Ted Pedersen, Anselmo Penas, Anselmo Peñas, Jing Peng, Fuchun Peng, Gerald Penn, Marco Pennacchiotti, Wim Peters, Slav Petrov, Emanuele Pianta, Michael Picheny, Daniele Pighin, Manfred Pinkal, David Pinto, Stelios Piperidis, Paul Piwek, Benjamin Piwowarski, Massimo Poesio, Livia Polanyi, Simone Paolo Ponzetto, Hoifung Poon, Ana-Maria Popescu, Andrei Popescu-Belis, Maja Popovic, Martin Potthast, Richard Power, Sameer Pradhan, John Prager, Rashmi Prasad, Partha Pratim Talukdar, Adam Przepiórkowski, Vasin Punyakanok, Matthew Purver, Sampo Pyysalo

Silvia Quarteroni, Ariadna Quattoni, Chris Quirk

Stephan Raaijmakers, Dragomir Radev, Filip Radlinski, Bhuvana Ramabhadran, Ganesh Ramakrishnan, Owen Rambow, Aarne Ranta, Delip Rao, Ari Rappoport, Lev Ratinov, Antoine Raux, Emmanuel Rayner, Roi Reichart, Ehud Reiter, Steve Renals, Philip Resnik, Giuseppe Riccardi, Sebastian Riedel, Stefan Riezler, German Rigau, Ellen Riloff, Laura Rimell, Eric Ringger, Horacio Rodríguez, Paolo Rosso, Antti-Veikko Rosti, Rachel Edita Roxas, Alex Rudnicky, Marta Ruiz Costa-Jussa, Vasile Rus, Graham Russell, Anton Rytting

Rune Sætre, Kenji Sagae, Horacio Saggion, Tapio Salakoski, Agnes Sandor, Sudeshna Sarkar, Anoop Sarkar, Giorgio Satta, Hassan Sawaf, Frank Schilder, Anne Schiller, David Schlangen, Sabine Schulte im Walde, Tanja Schultz, Holger Schwenk, Donia Scott, Yohei Seki, Satoshi Sekine, Stephanie Seneff, Jean Senellart, Violeta Seretan, Burr Settles, Serge Sharoff, Dou Shen, Wade Shen, Libin Shen, Kiyoaki Shirai, Luo Si, Grigori Sidorov, Mário Silva, Fabrizio Silvestri, Khalil Simaan, Michel Simard, Gabriel Skantze, Noah Smith, Matthew Snover, Rion Snow, Benjamin Snyder, Stephen Soderland, Marina Sokolova, Thamar Solorio, Swapna Somasundaran, Lucia Specia, Valentin Spitkovsky, Richard Sproat, Manfred Stede, Mark Steedman, Amanda Stent, Mark Stevenson, Svetlana Stoyanchev, Veselin Stoyanov, Michael Strube, Sara Stymne, Keh-Yih Su, Fangzhong Su, Jian Su, L Venkata Subramaniam, David Suendermann, Maosong Sun, Mihai Surdeanu, Richard Sutcliffe, Charles Sutton, Jun Suzuki, Stan Szpakowicz, Idan Szpektor

Hiroya Takamura, David Talbot, Irina Temnikova, Michael Tepper, Simone Teufel, Stefan Thater, Allan Third, Jörg Tiedemann, Christoph Tillmann, Ivan Titov, Takenobu Tokunaga, Kentaro Torisawa, Kristina Toutanova, Isabel Trancoso, Richard Tsai, Vivian Tsang, Dan Tufis

Takehito Utsuro

Shivakumar Vaithyanathan, Alessandro Valitutti, Antal van den Bosch, Hans van Halteren, Gertjan van Noord, Lucy Vanderwende, Vasudeva Varma, Tony Veale, Olga Vechtomova, Paola Velardi, Rene Venegas, Ashish Venugopal, Jose Luis Vicedo, Evelyne Viegas, David Vilar, Begona Villada Moiron, Sami Virpioja, Andreas Vlachos, Stephan Vogel, Piek Vossen

Michael Walsh, Xiaojun Wan, Xinglong Wang, Wei Wang, Haifeng Wang, Justin Washtell, Andy

Way, David Weir, Ben Wellner, Ji-Rong Wen, Chris Wendt, Michael White, Ryen White, Richard Wicentowski, Jan Wiebe, Sandra Williams, Jason Williams, Theresa Wilson, Shuly Wintner, Kam-Fai Wong, Fei Wu

Deyi Xiong, Peng Xu, Jinxi Xu, Nianwen Xue

Scott Wen-tau Yih, Emine Yilmaz

David Zajic, Fabio Zanzotto, Richard Zens, Torsten Zesch, Hao Zhang, Bing Zhang, Min Zhang, Huarui Zhang, Jun Zhao, Bing Zhao, Jing Zheng, Li Hai Zhou, Michael Zock, Andreas Zollmann, Geoffrey Zweig, Pierre Zweigenbaum

Secondary Reviewers

Omri Abend, Rodrigo Agerri, Paolo Annesi, Wilker Aziz, Tyler Baldwin, Verginica Barbu Mititelu, David Batista, Delphine Bernhard, Stephen Boxwell, Janez Brank, Chris Brockett, Tim Buckwalter, Wang Bukang, Alicia Burga, Steven Burrows, Silvia Calegari, Marie Candito, Marina Cardenas, Bob Carpenter, Paula Carvalho, Diego Ceccarelli, Asli Celikyilmaz, Soumaya Chaffar, Bin Chen, Danilo Croce, Daniel Dahlmeier, Hong-Jie Dai, Mariam Daoud, Steven DeNeefe, Leon Derczynski, Elina Desypri, Sobha Lalitha Devi, Gideon Dror, Loic Dugast, Eraldo Fernandes, Jody Foo, Kotaro Funakoshi, Jing Gao, Wei Gao, Diman Ghazi, Julius Goth, Joseph Grafsgaard, Eun Young Ha, Robbie Haertel, Matthias Hagen, Enrique Henestroza, Hieu Hoang, Maria Holmqvist, Dennis Hoppe, Yunhua Hu, Yun Huang, Radu Ion, Elena Irimia, Jagadeesh Jagarlamudi, Antonio Juárez-González, Sun Jun, Evangelos Kanoulas, Aaron Kaplan, Caroline Lavecchia, Lianhau Lee, Michael Levit, Ping Li, Thomas Lin, Wang Ling, Ying Liu, José David Lopes, Bin Lu, Jia Lu, Saab Mansour, Raquel Martinez-Unanue, Haitao Mi, Simon Mille, Teruhisa Misu, Behrang Mohit, Sílvio Moreira, Rutu Mulkar-Mehta, Jason Naradowsky, Sudip Naskar, Heung-Seon Oh, You Ouyang, Lluís Padró, Sujith Ravi, Marta Recasens, Luz Rello, Stefan Rigo, Alan Ritter, Alvaro Rodrigo, Hasim Sak, Kevin Seppi, Aliaksei Severyn, Chao Shen, Shuming Shi, Laurianne Sitbon, Jun Sun, György Szarvas, Eric Tang, Alberto Téllez-Valero, Luong Minh Thang, Gabriele Tolomei, David Tomás, Diana Trandabat, Zhaopeng Tu, Gokhan Tur, Kateryna Tymoshenko, Fabienne Venant, Esaú Villatoro-Tello, Joachim Wagner, Dan Walker, Wei Wei, Xinyan Xiao, Jun Xie, Hao Xiong, Gu Xu, Jun Xu, Huichao Xue, Taras Zagibalov, Beñat Zapirain, Kalliopi Zervanou, Renxian Zhang, Daqi Zheng, Arkaitz Zubiaga

Table of Contents

Conference Program

Monday, June 20, 2011

(8:45-9:00) Opening Session

(9:00-10:00) Invited Talk 1:IBM Watson Deep QA System (tentative title) by David Ferrucci

(10:00-10:30) Coffee Break

Session 1-A: (10:30-12:10) MT: Methods

A Word-Class Approach to Labeling PSCFG Rules for Machine Translation
Andreas Zollmann and Stephan Vogel

Deciphering Foreign Language
Sujith Ravi and Kevin Knight

Effective Use of Function Words for Rule Generalization in Forest-Based Translation
Xianchao Wu, Takuya Matsuzaki and Jun'ichi Tsujii

Combining Morpheme-based Machine Translation with Post-processing Morpheme Prediction
Ann Clifton and Anoop Sarkar

Session 1-B: (10:30-12:10) Machine Learning Methods 1

Evaluating the Impact of Coder Errors on Active Learning
Ines Rehbein and Josef Ruppenhofer

A Fast and Accurate Method for Approximate String Search
Ziqi Wang, Gu Xu, Hang Li and Ming Zhang

Domain Adaptation by Constraining Inter-Domain Variability of Latent Feature Representation
Ivan Titov

Exact Decoding of Syntactic Translation Models through Lagrangian Relaxation
Alexander M. Rush and Michael Collins

Monday, June 20, 2011 (continued)

Session 1-C: (10:30-12:10) Information Retrieval

Jigs and Lures: Associating Web Queries with Structured Entities
Patrick Pantel and Ariel Fuxman

Semi-Supervised SimHash for Efficient Document Similarity Search
Qixia Jiang and Maosong Sun

Joint Annotation of Search Queries
Michael Bendersky, W. Bruce Croft and David A. Smith

Query Weighting for Ranking Model Adaptation
Peng Cai, Wei Gao, Aoying Zhou and Kam-Fai Wong

Session 1-D: (10:30-12:10) Sentiment Analysis/Opinion Mining 1

Automatically Extracting Polarity-Bearing Topics for Cross-Domain Sentiment Classification
Yulan He, Chenghua Lin and Harith Alani

Using Multiple Sources to Construct a Sentiment Sensitive Thesaurus for Cross-Domain Sentiment Classification
Danushka Bollegala, David Weir and John Carroll

Learning Word Vectors for Sentiment Analysis
Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng and Christopher Potts

Target-dependent Twitter Sentiment Classification
Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu and Tiejun Zhao

Monday, June 20, 2011 (continued)

Session 1-E: (10:30-12:10) Language Resource

A Comprehensive Dictionary of Multiword Expressions
Kosho Shudo, Akira Kurahone and Toshifumi Tanabe

Multi-Modal Annotation of Quest Games in Second Life
Sharon Gower Small, Jennifer Strommer-Galley and Tomek Strzalkowski

A New Dataset and Method for Automatically Grading ESOL Texts
Helen Yannakoudakis, Ted Briscoe and Ben Medlock

Collecting Highly Parallel Data for Paraphrase Evaluation
David Chen and William Dolan

(12:10 - 2:00) Lunch

Session 2-A: (2:00-3:40) MT: Models & Evaluation

A Large Scale Distributed Syntactic, Semantic and Lexical Language Model for Machine Translation
Ming Tan, Wenli Zhou, Lei Zheng and Shaojun Wang

Goodness: A Method for Measuring Machine Translation Confidence
Nguyen Bach, Fei Huang and Yaser Al-Onaizan

MEANT: An inexpensive, high-accuracy, semi-automatic metric for evaluating translation utility based on semantic roles
Chi-kiu Lo and Dekai Wu

An exponential translation model for target language morphology
Michael Subotin

Session 2-B: (2:00-3:40) Machine Learning Methods 2

Bayesian Inference for Zodiac and Other Homophonic Ciphers
Sujith Ravi and Kevin Knight

Interactive Topic Modeling
Yuening Hu, Jordan Boyd-Graber and Brianna Satinoff

Faster and Smaller N-Gram Language Models
Adam Pauls and Dan Klein

Learning to Win by Reading Manuals in a Monte-Carlo Framework
S.R.K Branavan, David Silver and Regina Barzilay

Session 2-C: (2:00-3:40) Linguistic Creativity

Creative Language Retrieval: A Robust Hybrid of Information Retrieval and Linguistic Creativity
Tony Veale

Local Histograms of Character N-grams for Authorship Attribution
Hugo Jair Escalante, Thamar Solorio and Manuel Montes-y-Gomez

Word Maturity: Computational Modeling of Word Knowledge
Kirill Kireyev and Thomas K Landauer

Finding Deceptive Opinion Spam by Any Stretch of the Imagination
Myle Ott, Yejin Choi, Claire Cardie and Jeffrey T. Hancock

Monday, June 20, 2011 (continued)

Session 2-D: (2:00-3:40) Sentiment Analysis/Opinion Mining 2

Joint Bilingual Sentiment Classification with Unlabeled Parallel Corpora
Bin Lu, Chenhao Tan, Claire Cardie and Benjamin K. Tsou

A Pilot Study of Opinion Summarization in Conversations
Dong Wang and Yang Liu

Contrasting Opposing Views of News Articles on Contentious Issues
Souneil Park, Kyung Soon Lee and Junehwa Song

Content Models with Attitude
Christina Sauper, Aria Haghighi and Regina Barzilay

Session 2-E: (2:00-3:40) NLP for Web 2.0

Recognizing Named Entities in Tweets
Xiaohua LIU, Shaodian ZHANG, Furu WEI and Ming ZHOU

Lexical Normalisation of Short Text Messages: Makn Sens a #twitter
Bo Han and Timothy Baldwin

Topical Keyphrase Extraction from Twitter
Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achanauparp, Ee-Peng Lim and Xiaoming Li

Event Discovery in Social Media Feeds
Edward Benson, Aria Haghighi and Regina Barzilay

Monday, June 20, 2011 (continued)

(3:40-4:10) Coffee Break

Session 3-A: (4:10-5:50) Transliteration/Alignment

How do you pronounce your name? Improving G2P with transliterations
Aditya Bhargava and Grzegorz Kondrak

Unsupervised Word Alignment with Arbitrary Features
Chris Dyer, Jonathan H. Clark, Alon Lavie and Noah A. Smith

Model-Based Aligner Combination Using Dual Decomposition
John DeNero and Klaus Macherey

An Algorithm for Unsupervised Transliteration Mining with an Application to Word Alignment
Hassan Sajjad, Alexander Fraser and Helmut Schmid

Session 3-B: (4:10-5:50) Parsing 1

Beam-Width Prediction for Efficient Context-Free Parsing
Nathan Bodenstab, Aaron Dunlop, Keith Hall and Brian Roark

Optimal Head-Driven Parsing Complexity for Linear Context-Free Rewriting Systems
Pierluigi Crescenzi, Daniel Gildea, Andrea Marino, Gianluca Rossi and Giorgio Satta

Prefix Probability for Probabilistic Synchronous Context-Free Grammars
Mark-Jan Nederhof and Giorgio Satta

A Comparison of Loopy Belief Propagation and Dual Decomposition for Integrated CCG Supertagging and Parsing
Michael Auli and Adam Lopez

Monday, June 20, 2011 (continued)

Session 3-C: (4:10-5:50) Summarization

Jointly Learning to Extract and Compress
Taylor Berg-Kirkpatrick, Dan Gillick and Dan Klein

Discovery of Topically Coherent Sentences for Extractive Summarization
Asli Celikyilmaz and Dilek Hakkani-Tur

Coherent Citation-Based Summarization of Scientific Papers
Amjad Abu-Jbara and Dragomir Radev

A Class of Submodular Functions for Document Summarization
Hui Lin and Jeff Bilmes

Session 3-D: (4:10-5:50) Relation Extraction

Semi-supervised Relation Extraction with Large-scale Word Clustering
Ang Sun, Ralph Grishman and Satoshi Sekine

In-domain Relation Discovery with Meta-constraints via Posterior Regularization
Harr Chen, Edward Benson, Tahira Naseem and Regina Barzilay

Knowledge-Based Weak Supervision for Information Extraction of Overlapping Relations
Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer and Daniel S. Weld

Exploiting Syntactico-Semantic Structures for Relation Extraction
Yee Seng Chan and Dan Roth

Monday, June 20, 2011 (continued)

Session 3-E: (4:10-5:50) Semantics

Together We Can: Bilingual Bootstrapping for WSD
Mitesh M. Khapra, Salil Joshi, Arindam Chatterjee and Pushpak Bhattacharyya

Which Noun Phrases Denote Which Concepts?
Jayant Krishnamurthy and Tom Mitchell

Semantic Representation of Negation Using Focus Detection
Eduardo Blanco and Dan Moldovan

Learning Dependency-Based Compositional Semantics
Percy Liang, Michael Jordan and Dan Klein

(6:00-8:30) Poster Session (Long papers)

(6:00-8:30) Poster Session (Short papers)

Tuesday, June 21, 2011

Session 4-A: (9:00-10:30) Best Paper Session

Unsupervised Part-of-Speech Tagging with Bilingual Graph-Based Projections
Dipanjan Das and Slav Petrov

Global Learning of Typed Entailment Rules
Jonathan Berant, Ido Dagan and Jacob Goldberger

Tuesday, June 21, 2011 (continued)

(10:30-11:00) Coffee Break

(3:30-4:00) Coffee Break

Session 7-A: (4:00-5:40) SMT: Phrase-based Models

Incremental Syntactic Language Models for Phrase-based Translation
Lane Schwartz, Chris Callison-Burch, William Schuler and Stephen Wu

An Unsupervised Model for Joint Phrase Alignment and Extraction
Graham Neubig, Taro Watanabe, Eiichiro Sumita, Shinsuke Mori and Tatsuya Kawahara

Learning Hierarchical Translation Structure with Linguistic Annotations
Markos Mylonakis and Khalil Sima'an

Phrase-Based Translation Model for Question Retrieval in Community Question Answer Archives
Guangyou Zhou, Li Cai, Jun Zhao and Kang Liu

Session 7-B: (4:00-5:40) Parsing 2

Neutralizing Linguistically Problematic Annotations in Unsupervised Dependency Parsing Evaluation
Roy Schwartz, Omri Abend, Roi Reichart and Ari Rappoport

Dynamic Programming Algorithms for Transition-Based Dependency Parsers
Marco Kuhlmann, Carlos Gómez-Rodríguez and Giorgio Satta

Shift-Reduce CCG Parsing
Yue Zhang and Stephen Clark

Web-Scale Features for Full-Scale Parsing
Mohit Bansal and Dan Klein

Tuesday, June 21, 2011 (continued)

Session 7-C: (4:00-5:40) Spoken Language Processing

The impact of language models and loss functions on repair disfluency detection
Simon Zwarts and Mark Johnson

Learning Sub-Word Units for Open Vocabulary Speech Recognition
Carolina Parada, Mark Dredze, Abhinav Sethy and Ariya Rastrow

Computing and Evaluating Syntactic Complexity Features for Automated Scoring of Spontaneous Non-Native Speech
Miao Chen and Klaus Zechner

N-Best Rescoring Based on Pitch-accent Patterns
Je Hun Jeon, Wen Wang and Yang Liu

Session 7-D: (4:00-5:40) Natural Language Processing Applications

Lexically-Triggered Hidden Markov Models for Clinical Document Coding
Svetlana Kiritchenko and Colin Cherry

Learning to Grade Short Answer Questions using Semantic Similarity Measures and Dependency Graph Alignments
Michael Mohler, Razvan Bunescu and Rada Mihalcea

Age Prediction in Blogs: A Study of Style, Content, and Online Behavior in Pre- and Post-Social Media Generations
Sara Rosenthal and Kathleen McKeown

Extracting Social Power Relationships from Natural Language
Philip Bramsen, Martha Escobar-Molano, Ami Patel and Rafael Alonso

Tuesday, June 21, 2011 (continued)

Session 7-E: (4:00-5:40) Coreference Resolution

Bootstrapping coreference resolution using word associations
Hamidreza Kobdani, Hinrich Schuetze, Michael Schiehlen and Hans Kamp

Large-Scale Cross-Document Coreference Using Distributed Inference and Hierarchical Models
Sameer Singh, Amarnag Subramanya, Fernando Pereira and Andrew McCallum

A Cross-Lingual ILP Solution to Zero Anaphora Resolution
Ryu Iida and Massimo Poesio

Coreference Resolution with World Knowledge
Altaf Rahman and Vincent Ng

(7:00-11:00) Banquet

Wednesday, June 22, 2011

(9:00-10:00) Invited Talk 2: How do the languages we speak shape the ways we think? by Lera Boroditsky

(10:00-10:30) Coffee Break

Session 5-A: (10:30-12:10) SMT: Tree-based Models

How to train your multi bottom-up tree transducer
Andreas Maletti

Binarized Forest to String Translation
Hao Zhang, Licheng Fang, Peng Xu and Xiaoyun Wu

Learning to Transform and Select Elementary Trees for Improved Syntax-based Machine Translations
Bing Zhao, Young-Suk Lee, Xiaoqiang Luo and Liu Li

Rule Markov Models for Fast Tree-to-String Translation
Ashish Vaswani, Haitao Mi, Liang Huang and David Chiang

Wednesday, June 22, 2011 (continued)

Session 5-B: (10:30-12:10) Morphology/POS Induction

A Hierarchical Pitman-Yor Process HMM for Unsupervised Part of Speech Induction
Phil Blunsom and Trevor Cohn

Using Deep Morphology to Improve Automatic Error Detection in Arabic Handwriting Recognition
Nizar Habash and Ryan Roth

A Discriminative Model for Joint Morphological Disambiguation and Dependency Parsing
John Lee, Jason Naradowsky and David A. Smith

Unsupervised Bilingual Morpheme Segmentation and Alignment with Context-rich Hidden Semi-Markov Models
Jason Naradowsky and Kristina Toutanova

Session 5-C: (10:30-12:10) Error Correction

A Graph Approach to Spelling Correction in Domain-Centric Search
Zhuowei Bao, Benny Kimelfeld and Yunyao Li

Grammatical Error Correction with Alternating Structure Optimization
Daniel Dahlmeier and Hwee Tou Ng

Algorithm Selection and Model Adaptation for ESL Correction Tasks
Alla Rozovskaya and Dan Roth

Automated Whole Sentence Grammar Correction Using a Noisy Channel Model
Y. Albert Park and Roger Levy

Wednesday, June 22, 2011 (continued)

Session 5-D: (10:30-12:10) Information Extraction

A Generative Entity-Mention Model for Linking Entities with Knowledge Base
Xianpei Han and Le Sun

Simple supervised document geolocation with geodesic grids
Benjamin Wing and Jason Baldridge

Piggyback: Using Search Engines for Robust Cross-Domain Named Entity Recognition
Stefan Rüd, Massimiliano Ciaramita, Jens Müller and Hinrich Schütze

Template-Based Information Extraction without the Templates
Nathanael Chambers and Dan Jurafsky

Session 5-E: (10:30-12:10) Discourse

Classifying arguments by scheme
Vanessa Wei Feng and Graeme Hirst

Automatically Evaluating Text Coherence Using Discourse Relations
Ziheng Lin, Hwee Tou Ng and Min-Yen Kan

Underspecifying and Predicting Voice for Surface Realisation Ranking
Sina Zarrieß, Aoife Cahill and Jonas Kuhn

Recognizing Authority in Dialogue with an Integer Linear Programming Constrained Model
Elijah Mayfield and Carolyn Penstein Rosé

Wednesday, June 22, 2011 (continued)

(12:10 - 2:00) Lunch

(1:30-3:00) ACL Business Meeting

(3:00-3:30) Coffee Break

Session 6-A: (3:30-4:45) MT: Reordering Models

Reordering Metrics for MT
Alexandra Birch and Miles Osborne

Reordering with Source Language Collocations
Zhanyi Liu, Haifeng Wang, Hua Wu, Ting Liu and Sheng Li

A Joint Sequence Translation Model with Integrated Reordering
Nadir Durrani, Helmut Schmid and Alexander Fraser

Session 6-B: (3:30-4:45) Grammar

Integrating surprisal and uncertain-input models in online sentence comprehension: formal techniques and empirical results
Roger Levy

Metagrammar engineering: Towards systematic exploration of implemented grammars
Antske Fokkens

Simple Unsupervised Grammar Induction from Raw Text with Cascaded Finite State Models
Elias Ponvert, Jason Baldridge and Katrin Erk

Wednesday, June 22, 2011 (continued)

Session 6-C: (3:30-4:45) Generation/Paraphrasing

Extracting Paraphrases from Definition Sentences on the Web
Chikara Hashimoto, Kentaro Torisawa, Stijn De Saeger, Jun'ichi Kazama and Sadao Kurohashi

Learning From Collective Human Behavior to Introduce Diversity in Lexical Choice
Vahed Qazvinian and Dragomir R. Radev

Ordering Prenominal Modifiers with a Reranking Approach
Jenny Liu and Aria Haghighi

Session 6-D: (3:30-4:45) Event-Role Extraction

Unsupervised Semantic Role Induction via Split-Merge Clustering
Joel Lang and Mirella Lapata

Using Cross-Entity Inference to Improve Event Extraction
Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou and Qiaoming Zhu

Peeling Back the Layers: Detecting Event Role Fillers in Secondary Contexts
Ruihong Huang and Ellen Riloff

Session 6-E: (3:30-4:20) Knowledge Base Extension

Knowledge Base Population: Successful Approaches and Challenges
Heng Ji and Ralph Grishman

Nonlinear Evidence Fusion and Propagation for Hyponymy Relation Mining
Fan Zhang, Shuming Shi, Jing Liu, Shuqi Sun and Chin-Yew Lin

Wednesday, June 22, 2011 (continued)

(5:00-6:10) Life time achievement award and closing

Monday, June 20, 2011

(6:00-8:30) Poster Session (Long papers)

A Pronoun Anaphora Resolution System based on Factorial Hidden Markov Models
Dingcheng Li, Tim Miller and William Schuler

Disentangling Chat with Local Coherence Models
Micha Elsner and Eugene Charniak

An Affect-Enriched Dialogue Act Classification Model for Task-Oriented Dialogue
Kristy Boyer, Joseph Grafsgaard, Eun Young Ha, Robert Phillips and James Lester

Fine-Grained Class Label Markup of Search Queries
Joseph Reisinger and Marius Pasca

Creating a manually error-tagged and shallow-parsed learner corpus
Ryo Nagata, Edward Whittaker and Vera Sheinman

Crowdsourcing Translation: Professional Quality from Non-Professionals
Omar F. Zaidan and Chris Callison-Burch

A Statistical Tree Annotator and Its Applications
Xiaoqiang Luo and Bing Zhao

Consistent Translation using Discriminative Learning - A Translation Memory-inspired Approach
Yanjun Ma, Yifan He, Andy Way and Josef van Genabith

Machine Translation System Combination by Confusion Forest
Taro Watanabe and Eiichiro Sumita

Hypothesis Mixture Decoding for Statistical Machine Translation
Nan Duan, Mu Li and Ming Zhou

Minimum Bayes-risk System Combination
Jesús González-Rubio, Alfons Juan and Francisco Casacuberta

Adjoining Tree-to-String Translation
Yang Liu, Qun Liu and Yajuan Lü

Enhancing Language Models in Statistical Machine Translation with Backward N-grams and Mutual Information Triggers
Deyi Xiong, Min Zhang and Haizhou Li

Translating from Morphologically Complex Languages: A Paraphrase-Based Approach
Preslav Nakov and Hwee Tou Ng

Gappy Phrasal Alignment By Agreement
Mohit Bansal, Chris Quirk and Robert Moore

Translationese and Its Dialects
Moshe Koppel and Noam Ordan

Rare Word Translation Extraction from Aligned Comparable Documents
Emmanuel Prochasson and Pascale Fung

Using Bilingual Parallel Corpora for Cross-Lingual Textual Entailment
Yashar Mehdad, Matteo Negri and Marcello Federico

Using Large Monolingual and Bilingual Corpora to Improve Coordination Disambiguation
Shane Bergsma, David Yarowsky and Kenneth Church

Unsupervised Decomposition of a Document into Authorial Components
Moshe Koppel, Navot Akiva, Idan Dershowitz and Nachum Dershowitz

Discovering Sociolinguistic Associations with Structured Sparsity
Jacob Eisenstein, Noah A. Smith and Eric P. Xing

Local and Global Algorithms for Disambiguation to Wikipedia
Lev Ratinov, Dan Roth, Doug Downey and Mike Anderson

A Stacked Sub-Word Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging
Weiwei Sun

Language-independent compound splitting with morphological operations
Klaus Macherey, Andrew Dai, David Talbot, Ashok Popat and Franz Och

Parsing the Internal Structure of Words: A New Paradigm for Chinese Word Segmentation
Zhongguo Li

A Simple Measure to Assess Non-response
Anselmo Peñas and Alvaro Rodrigo

Improving Question Recommendation by Exploiting Information Need
Shuguang Li and Suresh Manandhar

Semi-Supervised Frame-Semantic Parsing for Unknown Predicates
Dipanjan Das and Noah A. Smith

A Bayesian Model for Unsupervised Semantic Parsing
Ivan Titov and Alexandre Klementiev

Unsupervised Learning of Semantic Relation Composition
Eduardo Blanco and Dan Moldovan

Unsupervised Discovery of Domain-Specific Knowledge from Text
Dirk Hovy, Chunliang Zhang, Eduard Hovy and Anselmo Peñas

Latent Semantic Word Sense Induction and Disambiguation
Tim Van de Cruys and Marianna Apidianaki

Confidence Driven Unsupervised Semantic Parsing
Dan Goldwasser, Roi Reichart, James Clarke and Dan Roth

Aspect Ranking: Identifying Important Product Aspects from Online Consumer Reviews
Jianxing Yu, Zheng-Jun Zha, Meng Wang and Tat-Seng Chua

Collective Classification of Congressional Floor-Debate Transcripts
Clinton Burfoot, Steven Bird and Timothy Baldwin

Integrating history-length interpolation and classes in language modeling
Hinrich Schütze

Structural Topic Model for Latent Topical Structure Analysis
Hongning Wang, Duo Zhang and ChengXiang Zhai

Automatic Labelling of Topic Models
Jey Han Lau, Karl Grieser, David Newman and Timothy Baldwin

Using Bilingual Information for Cross-Language Document Summarization
Xiaojun Wan

Exploiting Web-Derived Selectional Preference to Improve Statistical Dependency Parsing
Guangyou Zhou, Jun Zhao, Kang Liu and Li Cai

Effective Measures of Domain Similarity for Parsing
Barbara Plank and Gertjan van Noord

Efficient CCG Parsing: A versus Adaptive Supertagging*
Michael Auli and Adam Lopez

Improving Arabic Dependency Parsing with Form-based and Functional Morphological Features
Yuval Marton, Nizar Habash and Owen Rambow

Partial Parsing from Bitext Projections
Prashanth Mannem and Aswarth Dara

Ranking Class Labels Using Query Sessions
Marius Pasca

Insights from Network Structure for Text Mining
Zornitsa Kozareva and Eduard Hovy

Event Extraction as Dependency Parsing
David McClosky, Mihai Surdeanu and Christopher Manning

Extracting Comparative Entities and Predicates from Texts Using Comparative Type Classification
Seon Yang and Youngjoong Ko

Invited Talk 1

Building Watson: An Overview of the DeepQA Project

David Ferrucci, Principal Investigator, IBM Research

Monday, June 20, 2011 9:00-10:00

Computer systems that can directly and accurately answer peoples' questions over a broad domain of human knowledge have been envisioned by scientists and writers since the advent of computers themselves. Open domain question answering holds tremendous promise for facilitating informed decision making over vast volumes of natural language content. Applications in business intelligence, healthcare, customer support, enterprise knowledge management, social computing, science and government could all benefit from computer systems capable of deeper language understanding. The DeepQA project is aimed at exploring how advancing and integrating Natural Language Processing (NLP), Information Retrieval (IR), Machine Learning (ML), Knowledge Representation and Reasoning (KR&R) and massively parallel computation can greatly advance the science and application of automatic Question Answering. An exciting proof-point in this challenge was developing a computer system that could successfully compete against top human players at the Jeopardy! quiz show (www.jeopardy.com).

Attaining champion-level performance at Jeopardy! requires a computer to rapidly and accurately answer rich open-domain questions, and to predict its own performance on any given question. The system must deliver high degrees of precision and confidence over a very broad range of knowledge and natural language content with a 3-second response time. To do this, the DeepQA team advanced a broad array of NLP techniques to find, generate, evidence and analyze many competing hypotheses over large volumes of natural language content to build Watson (www.ibmwatson.com). An important contributor to Watson's success is its ability to automatically learn and combine accurate confidences across a wide array of algorithms and over different dimensions of evidence. Watson produced accurate confidences to know when to "buzz in" against its competitors and how much to bet. High precision and accurate confidence computations are critical for real business settings where helping users focus on the right content sooner and with greater confidence can make all the difference. The need for speed and high precision demands a massively parallel computing platform capable of generating, evaluating and combing 1000's of hypotheses and their associated evidence. In this talk, I will introduce the audience to the Jeopardy! Challenge, explain how Watson was built on DeepQA to ultimately defeat the two most celebrated human Jeopardy Champions of all time and I will discuss applications of the Watson technology beyond in areas such as healthcare.

Dr. David Ferrucci is the lead researcher and Principal Investigator (PI) for the Watson/Jeopardy! project. He has been a Research Staff Member at IBM's T.J. Watson's Research Center since 1995 where he heads up the Semantic Analysis and Integration department. Dr. Ferrucci focuses on technologies for automatically discovering valuable knowledge in natural language content and using it to enable better decision making.

Invited Talk 2

How do the languages we speak shape the ways we think?

Lera Boroditsky, Assistant Professor, Stanford University

Wednesday, June 22, 2011 9:00-10:00

Do people who speak different languages think differently? Does learning new languages change the way you think? Do polyglots think differently when speaking different languages? Are some thoughts unthinkable without language? I will describe data from experiments conducted around the world that reveal the powerful and often surprising ways that the languages we speak shape the ways we think.

Lera Boroditsky is an assistant professor of psychology at Stanford University and Editor in Chief of Frontiers in Cultural Psychology. Boroditsky's research centers on how knowledge emerges out of the interactions of mind, world, and language, and the ways that languages and cultures shape human thinking. To this end, Boroditsky's laboratory has collected data around the world, from Indonesia to Chile to Turkey to Aboriginal Australia. Her research has been widely featured in the media and has won multiple awards, including the CAREER award from the National Science Foundation, the Searle Scholars award, and the McDonnell Scholars award.

A Word-Class Approach to Labeling PSCFG Rules for Machine Translation

Andreas Zollmann and **Stephan Vogel**
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{zollmann,vogel+}@cs.cmu.edu

Abstract

In this work we propose methods to label probabilistic synchronous context-free grammar (PSCFG) rules using only word tags, generated by either part-of-speech analysis or unsupervised word class induction. The proposals range from simple tag-combination schemes to a phrase clustering model that can incorporate an arbitrary number of features.

Our models improve translation quality over the single generic label approach of Chiang (2005) and perform on par with the syntactically motivated approach from Zollmann and Venugopal (2006) on the NIST large Chinese-to-English translation task. These results persist when using automatically learned word tags, suggesting broad applicability of our technique across diverse language pairs for which syntactic resources are not available.

1 Introduction

The Probabilistic Synchronous Context Free Grammar (PSCFG) formalism suggests an intuitive approach to model the long-distance and lexically sensitive reordering phenomena that often occur across language pairs considered for statistical machine translation. As in monolingual parsing, nonterminal symbols in translation rules are used to generalize beyond purely lexical operations. *Labels* on these nonterminal symbols are often used to enforce syntactic constraints in the generation of bilingual sentences and imply conditional independence assumptions in the translation model. Several techniques have been recently proposed to automatically identify and estimate parameters for PSCFGs (or related synchronous grammars) from parallel corpora (Galley et al., 2004; Chiang, 2005; Zollmann and Venugopal, 2006; Liu et al., 2006; Marcu et al., 2006).

While all of these techniques rely on word-alignments to suggest lexical relationships, they differ in the way in which they assign labels to nonterminal symbols of PSCFG rules. Chiang (2005) describes a procedure to extract PSCFG rules from word-aligned (Brown et al., 1993) corpora, where all nonterminals share the same generic label X. In Galley et al. (2004) and Marcu et al. (2006), target language parse trees are used to identify rules and label their nonterminal symbols, while Liu et al. (2006) use source language parse trees instead. Zollmann and Venugopal (2006) directly extend the rule extraction procedure from Chiang (2005) to heuristically label any phrase pair based on target language parse trees. Label-based approaches have resulted in improvements in translation quality over the single X label approach (Zollmann et al., 2008; Mi and Huang, 2008); however, all the works cited here rely on stochastic parsers that have been trained on manually created syntactic treebanks. These treebanks are difficult and expensive to produce and exist for a limited set of languages only.

In this work, we propose a labeling approach that is based merely on part-of-speech analysis of the source or target language (or even both). Towards the ultimate goal of building end-to-end machine translation systems without *any* human annotations, we also experiment with automatically inferred word classes using distributional clustering (Kneser and Ney, 1993). Since the number of classes is a parameter of the clustering method and the resulting nonterminal size of our grammar is a function of the number of word classes, the PSCFG grammar complexity can be adjusted to the specific translation task at hand.

Finally, we introduce a more flexible labeling approach based on K-means clustering, which allows

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1–11,
Portland, Oregon, June 19-24, 2011.

the incorporation of an arbitrary number of word-class based features, including phrasal contexts, can make use of multiple tagging schemes, and also allows non-class features such as phrase sizes.

2 PSCFG-based translation

In this work we experiment with PSCFGs that have been automatically learned from word-aligned parallel corpora. PSCFGs are defined by a source terminal set (source vocabulary) $\mathcal{T}_S$, a target terminal set (target vocabulary) $\mathcal{T}_T$, a shared nonterminal set $\mathcal{N}$ and rules of the form: $A \rightarrow \langle \gamma, \alpha, w \rangle$ where

- $A \in \mathcal{N}$ is a labeled nonterminal referred to as the left-hand-side of the rule,
- $\gamma \in (\mathcal{N} \cup \mathcal{T}_S)^*$ is the source side of the rule,
- $\alpha \in (\mathcal{N} \cup \mathcal{T}_T)^*$ is the target side of the rule,
- $w \in [0, \infty)$ is a non-negative real-valued weight assigned to the rule; in our model, w is the product of features ϕ_i raised to the power of weight λ_i.

Chiang (2005) learns a single-nonterminal PSCFG from a bilingual corpus by first identifying initial phrase pairs using the technique from Koehn et al. (2003), and then performing a generalization operation to generate phrase pairs with gaps, which can be viewed as PSCFG rules with generic 'X' nonterminal left-hand-sides and substitution sites. Bilingual features ϕ_i that judge the quality of each rule are estimated based on rule extraction frequency counts.

3 Hard rule labeling from word classes

We now describe a simple method of inducing a multi-nonterminal PSCFG from a parallel corpus with word-tagged target side sentences. The same procedure can straightforwardly be applied to a corpus with tagged source side sentences. We use the simple term 'tag' to stand for any kind of word-level analysis—a syntactic, statistical, or other means of grouping word types or tokens into classes, possibly based on their position and context in the sentence, POS tagging being the most obvious example.

As in Chiang's hierarchical system, we rely on an external phrase-extraction procedure such as the one of Koehn et al. (2003) to provide us with a set of phrase pairs for each sentence pair in the training corpus, annotated with their respective start and end positions in the source and target sentences. Let $\boldsymbol{f} = f_1 \cdots f_m$ be the current source sentence, $\boldsymbol{e} = e_1 \cdots e_n$ the current target sentence, and $\boldsymbol{t} = t_1 \cdots t_n$ its corresponding target tag sequence. We convert each extracted phrase pair, represented by its source span $\langle i, j \rangle$ and target span $\langle k, \ell \rangle$, into an initial rule

$$t_k\text{-}t_\ell \rightarrow f_i \cdots f_j \mid e_k \cdots e_\ell$$

by assigning it a nonterminal "t_k-t_ℓ" constructed by combining the tag of the target phrase's left-most word with the tag of its right-most word.

The creation of complex rules based on all initial rules obtained from the current sentence now proceeds just as in Chiang's model.

Consider the target-tagged example sentence pair:

Ich habe ihn gesehen | I/PRP saw/VBD him/PRP

Then (depending on the extracted phrase pairs), the resulting initial rules could be:

1: PRP-PRP → Ich | I
2: PRP-PRP → ihn | him
3: VBD-VBD → gesehen | saw
4: VBD-PRP → habe ihn gesehen | saw him
5: PRP-PRP → Ich habe ihn gesehen | I saw him

Now, by abstracting-out initial rule 2 from initial rule 4, we obtain the complex rule:

VBD-PRP → habe PRP-PRP$_1$ gesehen | saw PRP-PRP$_1$

Intuitively, the labeling of initial rules with tags marking the boundary of their target sides results in complex rules whose nonterminal occurrences impose weak syntactic constraints on the rules eligible for substitution in a PSCFG derivation: The left and right boundary word tags of the inserted rule's target side have to match the respective boundary word tags of the phrase pair that was replaced by a nonterminal when the complex rule was created from a training sentence pair. Since consecutive words within a rule stem from consecutive words in the training corpus and thus are already consistent, the boundary word tags are more informative than tags of words between the boundaries for the task of combining different rules in a derivation, and are therefore a more appropriate choice for the creation of grammar labels than tags of inside words.

Accounting for phrase size A drawback of the current approach is that a single-word rule such as

PRP-PRP → Ich | I

can have the same left-hand-side nonterminal as a long rule with identical left and right boundary tags, such as (when using target-side tags):

PRP-PRP → Ich habe ihn gesehen | I saw him

We therefore introduce a means of distinguishing between one-word, two-word, and multiple-word phrases as follows: Each one-word phrase with tag T simply receives the label T, instead of T-T. Two-word phrases with tag sequence T_1T_2 are labeled T_1-T_2 as before. Phrases of length greater two with tag sequence $T_1 \cdots T_n$ are labeled $T_1..T_n$ to denote that tags were omitted from the phrase's tag sequence. The resulting number of grammar nonterminals based on a tag vocabulary of size t is thus given by $2t^2 + t$.

An alternative way of accounting for phrase size is presented by Chiang et al. (2008), who introduce *structural distortion features* into a hierarchical phrase-based model, aimed at modeling nonterminal reordering given source span length. Our approach instead uses distinct grammar rules and labels to discriminate phrase size, with the advantage of enabling all translation models to estimate distinct weights for distinct size classes and avoiding the need of additional models in the log-linear framework; however, the increase in the number of labels and thus grammar rules decreases the reliability of estimated models for rare events due to increased data sparseness.

Extension to a bilingually tagged corpus While the availability of syntactic annotations for both source *and* target language is unlikely in most translation scenarios, some form of word tags, be it part-of-speech tags or learned word clusters (cf. Section 3) might be available on both sides. In this case, our grammar extraction procedure can be easily extended to impose both source and target constraints on the eligible substitutions simultaneously.

Let N_f be the nonterminal label that would be assigned to a given initial rule when utilizing the source-side tag sequence, and N_e the assigned label according to the target-side tag sequence. Then our bilingual tag-based model assigns '$N_f + N_e$' to the initial rule. The extraction of complex rules proceeds as before. The number of nonterminals in this model, based on a source tag vocabulary of size s and a target tag vocabulary of size t, is thus given by s^2t^2 for the regular labeling method and $(2s^2 + s)(2t^2 + t)$ when accounting for phrase size.

Consider again our example sentence pair (now also annotated with source-side part-of-speech tags):

Ich/PRP habe/AUX ihn/PRP gesehen/VBN
I/PRP saw/VBD him/PRP

Given the same phrase extraction method as before, the resulting initial rules for our bilingual model, when also accounting for phrase size, are as follows:

1: PRP+PRP → Ich | I
2: PRP+PRP → ihn | him
3: VBN+VBD → gesehen | saw
4: AUX..VBN+VBD-PRP → habe ihn gesehen | saw him
5: PRP..VBN+PRP..PRP → Ich habe ihn gesehen | I saw him

Abstracting-out rule 2 from rule 4, for instance, leads to the complex rule:

AUX..VBN+VBD-PRP → habe PRP+PRP$_1$ gesehen | saw PRP+PRP$_1$

Unsupervised word class assignment by clustering As an alternative to POS tags, we experiment with unsupervised word clustering methods based on the exchange algorithm (Kneser and Ney, 1993). Its objective function is maximizing the likelihood

$$\prod_{i=1}^{n} P(w_i|w_1, \ldots, w_{i-1})$$

of the training data $w = w_1, \ldots, w_n$ given a partially class-based bigram model of the form

$$P(w_i|w_1, \ldots, w_{i-1}) \approx p(c(w_i)|w_{i-1}) \cdot p(w_i|c(w_i))$$

where $c : \mathcal{V} \rightarrow \{1, \ldots, N\}$ maps a word (type, not token) w to its class $c(w)$, $\mathcal{V}$ is the vocabulary, and N the fixed number of classes, which has to be chosen *a priori*. We use the publicly available implementation MKCLS (Och, 1999) to train this model. As training data we use the respective side of the parallel training data for the translation system.

We also experiment with the extension of this model by Clark (2003), who incorporated morphological information by imposing a Bayesian prior on the class mapping c, based on N individual distributions over strings, one for each word class. Each such distribution is a character-based hidden Markov model, thus encouraging the grouping of morphologically similar words into the same class.

4 Clustering phrase pairs directly using the K-means algorithm

Even though we have only made use of the first and last words' classes in the labeling methods described so far, the number of resulting grammar nonterminals quickly explodes. Using a scheme based on source and target phrases with accounting for phrase size, with 36 word classes (the size of the Penn English POS tag set) for both languages, yields a grammar with $(36+2*36^2)^2 = 6.9$m nonterminal labels.

Quite plausibly, phrase labeling should be informed by more than just the classes of the first and last words of the phrase. Taking phrase context into account, for example, can aid the learning of syntactic properties: a phrase beginning with a determiner and ending with a noun, with a verb as right context, is more likely to be a noun phrase than the same phrase with another noun as right context. In the current scheme, there is no way of distinguishing between these two cases. Similarly, it is conceivable that using non-boundary words inside the phrase might aid the labeling process.

When relying on unsupervised learning of the word classes, we are forced to chose a fixed number of classes. A smaller number of word clusters will result in smaller number of grammar nonterminals, and thus more reliable feature estimation, while a larger number has the potential to discover more subtle syntactic properties. Using multiple word clusterings simultaneously, each based on a different number of classes, could turn this global, hard trade-off into a local, soft one, informed by the number of phrase pair instances available for a given granularity.

Lastly, our method of accounting for phrase size is somewhat displeasing: While there is a hard partitioning of one-word and two-word phrases, no distinction is made between phrases of length greater than two. Marking phrase sizes greater than two explicitly by length, however, would create many sparse, low-frequency rules, and one of the strengths of PSCFG-based translation is the ability to substitute flexible-length spans into nonterminals of a derivation. A partitioning where phrase size is instead merely a feature informing the labeling process seems more desirable.

We thus propose to represent each phrase pair instance (including its bilingual one-word contexts) as feature vectors, i.e., points of a vector space. We then use these data points to partition the space into clusters, and subsequently assign each phrase pair instance the cluster of its corresponding feature vector as label.

The feature mapping Consider the phrase pair instance

$$(f_0)f_1 \cdots f_m(f_{m+1}) \mid (e_0)e_1 \cdots e_n(e_{n+1})$$

(where $f_0, f_{m+1}, e_0, e_{n+1}$ are the left and right, source and target side contexts, respectively). We begin with the case of only a single, target-side word class scheme (either a tagger or an unsupervised word clustering/POS induction method). Let $C = \{c_1, \ldots, c_N\}$ be its set of word classes. Further, let c_0 be a short-hand for the result of looking up the class of a word that is out of bounds (e.g., the left context of the first word of a sentence, or the second word of a one-word phrase). We now map our phrase pair instance to the real-valued vector (where $\mathbb{1}_{[P]}$ is the indicator function defined as 1 if property P is true, and 0 otherwise):

$$\Big\langle \mathbb{1}_{[e_1=c_0]}, \ldots, \mathbb{1}_{[e_1=c_N]}, \mathbb{1}_{[e_n=c_0]}, \ldots, \mathbb{1}_{[e_n=c_N]},$$
$$\alpha_{\text{sec}}\mathbb{1}_{[e_2=c_0]}, \ldots, \alpha_{\text{sec}}\mathbb{1}_{[e_2=c_N]},$$
$$\alpha_{\text{sec}}\mathbb{1}_{[e_{n-1}=c_0]}, \ldots, \alpha_{\text{sec}}\mathbb{1}_{[e_{n-1}=c_N]},$$
$$\frac{\alpha_{\text{ins}}\sum_{i=1}^{n}\mathbb{1}_{[e_i=c_0]}}{n}, \ldots, \frac{\alpha_{\text{ins}}\sum_{i=1}^{n}\mathbb{1}_{[e_i=c_N]}}{n},$$
$$\alpha_{\text{cntxt}}\mathbb{1}_{[e_0=c_0]}, \ldots, \alpha_{\text{cntxt}}\mathbb{1}_{[e_0=c_N]},$$
$$\alpha_{\text{cntxt}}\mathbb{1}_{[e_{n+1}=c_0]}, \ldots, \alpha_{\text{cntxt}}\mathbb{1}_{[e_{n+1}=c_N]},$$
$$\alpha_{\text{phrsize}}\sqrt{N+1}\log_{10}(n)\Big\rangle$$

The α parameters determine the influence of the different types of information. The elements in the first line represent the phrase boundary word classes, the next two lines the classes of the second and penultimate word, followed by a line representing the accumulated contents of the whole phrase, followed by two lines pertaining to the context word classes. The final element of the vector is proportional to the logarithm of the phrase length.[1] We chose the logarithm assuming that length deviation of syntactic phrasal units is not constant, but proportional to the average length. Thus, all other features being equal, the distance between a two-word and a four-word phrase is

[1] The $\sqrt{N+1}$ factor serves to make the feature's influence independent of the number of word classes by yielding the same distance (under L_2) as $N+1$ identical copies of the feature.

the same as the distance between a four-word and an eight-word phrase.

We will mainly use the Euclidean (L_2) distance to compare points for clustering purposes. Our feature space is thus the Euclidean vector space $\mathbb{R}^{7N+8}$.

To additionally make use of source-side word classes, we append elements analogous to the ones above to the vector, all further multiplied by a parameter α_{src} that allows trading off the relevance of source-side and target-side information. In the same fashion, we can incorporate multiple tagging schemes (e.g., word clusterings of different granularities) into the same feature vector. As finer-grained schemes have more elements in the feature vector than coarser-grained ones, and thus exert more influence, we set the α parameter for each scheme to $1/N$ (where N is the number of word classes of the scheme).

The K-means algorithm To create the clusters, we chose the K-means algorithm (Steinhaus, 1956; MacQueen, 1967) for both its computational efficiency and ease of implementation and parallelization. Given an initial mapping from the data points to K clusters, the procedure alternates between (i) computing the centroid of each cluster and (ii) reallocating each data point to the closest cluster centroid, until convergence.

We implemented two commonly used initialization methods: Forgy and Random Partition. The Forgy method randomly chooses K observations from the data set and uses these as the initial means. The Random Partition method first randomly assigns a cluster to each observation and then proceeds straight to step (ii). Forgy tends to spread the initial means out, while Random Partition places all of them close to the center of the data set. As the resulting clusters looked similar, and Random Partition sometimes led to a high rate of empty clusters, we settled for Forgy.

5 Experiments

We evaluate our approach by comparing translation quality, as evaluated by the IBM-BLEU (Papineni et al., 2002) metric on the NIST Chinese-to-English translation task using MT04 as development set to train the model parameters λ, and MT05, MT06 and MT08 as test sets. Even though a key advantage of our method is its applicability to resource-poor languages, we used a language pair for which linguistic resources are available in order to determine how close translation performance can get to a fully syntax-based system. Accordingly, we use Chiang's hierarchical phrase based translation model (Chiang, 2007) as a base line, and the syntax-augmented MT model (Zollmann and Venugopal, 2006) as a 'target line', a model that would not be applicable for language pairs without linguistic resources.

We perform PSCFG rule extraction and decoding using the open-source "SAMT" system (Venugopal and Zollmann, 2009), using the provided implementations for the hierarchical and syntax-augmented grammars. Apart from the language model, the lexical, phrasal, and (for the syntax grammar) label-conditioned features, and the rule, target word, and glue operation counters, Venugopal and Zollmann (2009) also provide both the hierarchical and syntax-augmented grammars with a rareness penalty $1/\operatorname{cnt}(r)$, where $\operatorname{cnt}(r)$ is the occurrence count of rule r in the training corpus, allowing the system to learn penalization of low-frequency rules, as well as three indicator features firing if the rule has one, two unswapped, and two swapped nonterminal pairs, respectively.[2] Further, to mitigate badly estimated PSCFG derivations based on low-frequency rules of the much sparser syntax model, the syntax grammar also contains the hierarchical grammar as a backbone (cf. Zollmann and Vogel (2010) for details and empirical analysis).

We implemented our rule labeling approach within the SAMT rule extraction pipeline, resulting in comparable features across all systems. For all systems, we use the bottom-up chart parsing decoder implemented in the SAMT toolkit with a reordering limit of 15 source words, and correspondingly extract rules from initial phrase pairs of maximum source length 15. All rules have at most two nonterminal symbols, which must be non-consecutive on the source side, and rules must contain at least one source-side terminal symbol. The beam settings for the hierarchical system are 600 items per 'X' (generic rule) cell, and 600 per 'S' (glue) cell.[3] Due to memory limitations, the multi-nonterminal grammars have to be pruned more harshly: We al-

[2] Penalization or reward of purely-lexical rules can be indirectly learned by trading off these features with the rule counter feature.

[3] For comparison, Chiang (2007) uses 30 and 15, respectively, and further prunes items that deviate too much in score from the best item. He extracts initial phrases of maximum length 10.

low 100 'S' items, and a total of 500 non-'S' items, but maximally 40 items per nonterminal. For all systems, we further discard non-initial rules occurring only once.[4] For the multi-nonterminal systems, we generally further discard all non-generic non-initial rules occurring less than 6 times, but we additionally give results for a 'slow' version of the Syntax target-line system and our best word class based systems, where only single-occurrences were removed.

For parameter tuning, we use the L_0-regularized minimum-error-rate training tool provided by the SAMT toolkit. Each system is trained separately to adapt the parameters to its specific properties (size of nonterminal set, grammar complexity, features sparseness, reliance on the language model, etc.).

The parallel training data comprises of 9.6M sentence pairs (206M Chinese and 228M English words). The source and target language parses for the syntax-augmented grammar, as well as the POS tags for our POS-based grammars were generated by the Stanford parser (Klein and Manning, 2003).

The results are given in Table 1. Results for the Syntax system are consistent with previous results (Zollmann et al., 2008), indicating improvements over the hierarchical system. Our approach, using target POS tags ('POS-tgt (no phr. s.)'), outperforms the hierarchical system on all three tests sets, and gains further improvements when accounting for phrase size ('POS-tgt'). The latter approach is roughly on par with the corresponding Syntax system, slightly outperforming it on average, but not consistently across all test sets. The same is true for the 'slow' version ('POS-tgt-slow').

The model based on bilingually tagged training instances ('POS-src&tgt') does not gain further improvements over the merely target-based one, but actually performs worse. We assume this is due to the huge number of nonterminals of 'POS-src&tgt' ($(2 * 33^2 + 33)(2 * 36^2 + 36) = 5.8$M in principle) compared to 'POS-tgt' ($2 * 36^2 + 36 = 2628$), increasing the sparseness of the grammar and thus leading to less reliable statistical estimates.

We also experimented with a source-tag based model ('POS-src'). In line with previous findings for syntax-augmented grammars (Zollmann and Vogel, 2010), the source-side-based grammar does not reach the translation quality of its target-based counterpart; however, the model still outperforms the hierarchical system on all test sets. Further, decoding is much faster than for 'POS-ext-tgt' and even slightly faster than 'Hierarchical'. This is due to the fact that for the source-tag based approach, a given chart cell in the CYK decoder, represented by a start and end position in the source sentence, almost uniquely determines the nonterminal any hypothesis in this cell can have: Disregarding part-of-speech tag ambiguity and phrase size accounting, that nonterminal will be the composition of the tags of the start and end source words spanned by that cell. At the same time, this demonstrates that there is hence less of a role for the nonterminal labels to resolve translational ambiguity in the source based model than in the target based model.

Performance of the word-clustering based models To empirically validate the unsupervised clustering approaches, we first need to decide how to determine the number of word classes, N. A straightforward approach is to run experiments and report test set results for many different N. While this would allow us to reliably conclude the optimal number N, a comparison of that best-performing clustering method to the hierarchical, syntax, and POS systems would be tainted by the fact that N was effectively tuned on the test sets. We therefore choose N merely based on development set performance. Unfortunately, variance in development set BLEU scores tends to be higher than test set scores, despite of SAMT MERT's inbuilt algorithms to overcome local optima, such as random restarts and zeroing-out. We have noticed that using an L_0-penalized BLEU score[5] as MERT's objective on the merged n-best lists over all iterations is more stable and will therefore use this score to determine N.

Figure 1 (left) shows the performance of the distributional clustering model ('Clust') and its morphology-sensitive extension ('Clust-morph') according to this score for varying values of $N = 1, \ldots, 36$ (the number Penn treebank POS tags, used for the 'POS' models, is 36).[6] For 'Clust', we see a comfortably wide plateau of nearly-identical scores from $N = 7, \ldots, 15$. Scores for 'Clust-morph' are lower throughout, and peak at $N = 7$.

Looking back at Table 1, we now compare the clustering models chosen by the procedure above—

[4] As shown in Zollmann et al. (2008), the impact of these rules on translation quality is negligible.

[5] Given by: BLEU $-\beta \times |\{i \in \{1, \ldots, K\} | \lambda_i \neq 0\}|$, where $\lambda_1, \ldots, \lambda_K$ are the feature weights and the constant β (which we set to 0.00001) is the regularization penalty.

[6] All these models account for phrase size.

	Dev (MT04)	MT05	MT06	MT08	**TestAvg**	Time
Hierarchical	38.63	36.51	33.26	25.77	**31.85**	14.3
Syntax	39.39	37.09	34.01	26.53	**32.54**	18.1
Syntax-slow	39.69	37.56	34.66	26.93	**33.05**	34.6
POS-tgt (no phr. s.)	39.31	37.29	33.79	26.13	**32.40**	27.7
POS-tgt	39.14	37.29	33.97	26.77	**32.68**	19.2
POS-src	38.74	36.75	33.85	26.76	**32.45**	12.2
POS-src&tgt	38.78	36.71	33.65	26.52	**32.29**	18.8
POS-tgt-slow	39.86	37.78	34.37	27.14	**33.10**	44.6
Clust-7-tgt	39.24	36.74	34.00	26.93	**32.56**	24.3
Clust-7-morph-tgt	39.08	36.57	33.81	26.40	**32.26**	23.6
Clust-7-src	38.68	36.17	33.23	26.55	**31.98**	11.1
Clust-7-src&tgt	38.71	36.49	33.65	26.33	**32.16**	15.8
Clust-7-tgt-slow	39.48	37.70	34.31	27.24	**33.08**	45.2
kmeans-POS-src&tgt	39.11	37.23	33.92	26.80	**32.65**	18.5
kmeans-POS-src&tgt-L_1	39.33	36.92	33.81	26.59	**32.44**	17.6
kmeans-POS-src&tgt-cosine	39.15	37.07	33.98	26.68	**32.58**	17.7
kmeans-POS-src&tgt ($\alpha_{\text{ins}} = .5$)	39.07	36.88	33.71	26.26	**32.28**	16.5
kmeans-Clust-7-src&tgt	39.19	36.96	34.26	26.97	**32.73**	19.3
kmeans-Clust-7..36-src&tgt	39.09	36.93	34.24	26.92	**32.70**	17.3
kmeans-POS-src&tgt-slow	39.28	37.16	34.38	27.11	**32.88**	36.3
kmeans-Clust-7..36-s&t-slow	39.18	37.12	34.13	27.35	**32.87**	34.3

Table 1: Translation quality in % case-insensitive IBM-BLEU (i.e., brevity penalty based on closest reference length) for Chinese-English NIST-large translation tasks, comparing baseline Hierarchical and Syntax systems with POS and clustering based approaches proposed in this work. 'TestAvg' shows the average score over the three test sets. 'Time' is the average decoding time per sentence in seconds on one CPU.

resulting in $N = 7$ for the morphology-unaware model ('Clust-7-tgt') as well as the morphology-aware model ('Clust-7-morph-tgt')—to the other systems. 'Clust-7-tgt' improves over the hierarchical base line on all three test sets and is on par with the corresponding Syntax and POS target lines. The same holds for the 'Clust-7-tgt-slow' version. We also experimented with a model variant based on seven source and seven target language clusters ('Clust-7-src&tgt') and a source-only labeled model ('Clust-7-src')—both performing worse.

Surprisingly, the morphology-sensitive clustering model ('Clust-7-morph-tgt'), while still improving over the hierarchical system, performs worse than the morphology-unaware model. An inspection of the trained word clusters showed that the model, while far superior to the morphology-unaware model in e.g. mapping all numbers to the same class, is overzealous in discovering morphological regularities (such as the '-ed' suffix) to partition functionally only slightly dissimilar words (such present-tense and past-tense verbs) into different classes. While these subtle distinctions make for good partitionings when the number of clusters is large, they appear to lead to inferior results for our task that relies on coarse-grained partitionings of the vocabulary. Note that there are no 'src' or 'src&tgt' systems for 'Clust-morph', as Chinese, being a monosyllabic writing system, does not lend itself to morphology-sensitive clustering.

K-means clustering based models To establish suitable values for the α parameters and investigate the impact of the number of clusters, we looked at the development performance over various parameter combinations for a K-means model based on source and/or target part-of-speech tags.[7] As can be seen from Figure 1 (right), our method reaches its peak performance at around 50 clusters and then levels off slightly. Encouragingly, in contrast to the hard labeling procedure, K-means actually improves when adding source-side information. The optimal ratio of weighting source and target classes is 0.5:1, corresponding to $\alpha_{\text{src}} = .5$. Incorporating context information also helps, and does best for $\alpha_{\text{cntxt}} = 0.25$, i.e. when giving contexts 1/4 the influence of the phrase boundary words.

[7] We set $\alpha_{\text{sec}} = .25$, $\alpha_{\text{ins}} = 0$, and $\alpha_{\text{phrsize}} = .5$ throughout.

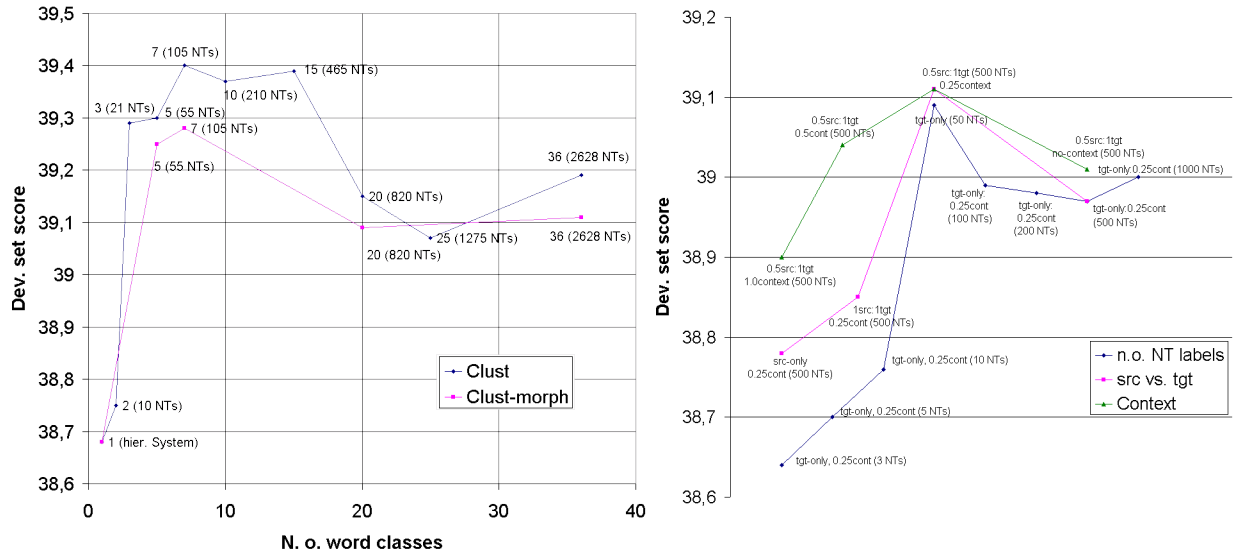


Figure 1: ***Left***: Performance of the distributional clustering model 'Clust' and its morphology-sensitive extension 'Clust-morph' according to L_0-penalized development set BLEU score for varying numbers N of word classes. For each data point N, its corresponding n.o. nonterminals of the induced grammar is stated in parentheses.
Right: Dev. set performance of K-means for various n.o. labels and values of α_{src} and α_{cntxt}.

Entry 'kmeans-POS-src&tgt' in Table 1 shows the test set results for the development-set best K-means configuration (i.e., $\alpha_{\text{src}} = .5$, $\alpha_{\text{cntxt}} = 0.25$, and using 500 clusters). While beating the hierarchical baseline, it is only minimally better than the much simpler target-based hard labeling method 'POS-tgt'. We also tried K-means variants in which the Euclidean distance metric is replaced by the city block distance L_1 and the cosine dissimilarity, respectively, with slightly worse outcomes. Configuration 'kmeans-POS-src&tgt ($\alpha_{\text{ins}} = .5$)' investigates the incorporation of non-boundary word tags inside the phrase. Unfortunately, these features appear to deteriorate performance, presumably because given a fixed number of clusters, accounting for contents inside the phrase comes at the cost of neglect of boundary words, which are more relevant to producing correctly reordered translations.

The two completely unsupervised systems 'kmeans-Clust-7-src&tgt' (based on 7-class MKCLS distributional word clustering) and 'kmeans-Clust-7..36-src&tgt' (using six different word clustering models simultaneously: all the MKCLS models from Figure 1 (left) except for the two-, three- and five-class models) have the best results, outperforming the other K-means models as well as 'Syntax' and 'POS-tgt' on average, but not on all test sets.

Lastly, we give results for 'slow' K-means configurations ('kmeans-POS-src&tgt-slow' and 'kmeans-Clust-7..36-s&t-slow'). Unfortunately (or fortunately, from a pragmatic viewpoint), the models are outperformed by the much simpler 'POS-tgt-slow' and 'Clust-7-tgt-slow' models.

6 Related work

Hassan et al. (2007) improve the statistical phrase-based MT model by injecting *supertags*, lexical information such as the POS tag of the word and its subcategorization information, into the phrase table, resulting in generalized phrases with placeholders in them. The supertags are also injected into the language model. Our approach also generates phrase labels and placeholders based on word tags (albeit in a different manner and without the use of subcategorization information), but produces PSCFG rules for use in a parsing-based decoding system.

Unsupervised *synchronous* grammar induction, apart from the contribution of Chiang (2005) discussed earlier, has been proposed by Wu (1997) for inversion transduction grammars, but as Chiang's model only uses a single generic nonterminal label. Blunsom et al. (2009) present a nonparametric PSCFG translation model that directly induces a grammar from parallel sentences without the use of or constraints from a word-alignment model, and

Cohn and Blunsom (2009) achieve the same for tree-to-string grammars, with encouraging results on small data. Our more humble approach treats the training sentences' word alignments and phrase pairs, obtained from external modules, as ground truth and employs a straight-forward generalization of Chiang's popular rule extraction approach to labeled phrase pairs, resulting in a PSCFG with multiple nonterminal labels.

Our phrase pair clustering approach is similar in spirit to the work of Lin and Wu (2009), who use K-means to cluster (monolingual) phrases and use the resulting clusters as features in discriminative classifiers for a named-entity-recognition and a query classification task. Phrases are represented in terms of their contexts, which can be more than one word long; words within the phrase are not considered. Further, each context contributes one dimension per vocabulary word (not per word class as in our approach) to the feature space, allowing for the discovery of subtle semantic similarities in the phrases, but at much greater computational expense. Another distinction is that Lin and Wu (2009) work with phrase types instead of phrase instances, obtaining a phrase type's contexts by averaging the contexts of all its phrase instances.

Nagata et al. (2006) present a reordering model for machine translation, and make use of clustered phrase pairs to cope with data sparseness in the model. They achieve the clustering by reducing phrases to their head words and then applying the MKCLS tool to these pseudo-words.

Kuhn et al. (2010) cluster the phrase pairs of an SMT phrase table based on their co-occurrence counts and edit distances in order to arrive at semantically similar phrases for the purpose of phrase table smoothing. The clustering proceeds in a bottom-up fashion, gradually merging similar phrases while alternating back and forth between the two languages.

7 Conclusion and discussion

In this work we proposed methods of labeling phrase pairs to create automatically learned PSCFG rules for machine translation. Crucially, our methods only rely on "shallow" lexical tags, either generated by POS taggers or by automatic clustering of words into classes. Evaluated on a Chinese-to-English translation task, our approach improves translation quality over a popular PSCFG baseline—the hierarchical model of Chiang (2005) —and performs on par with the model of Zollmann and Venugopal (2006), using heuristically generated labels from parse trees. Using automatically obtained word clusters instead of POS tags yields essentially the same results, thus making our methods applicable to all languages pairs with parallel corpora, whether syntactic resources are available for them or not.

We also propose a more flexible way of obtaining the phrase labels from word classes using K-means clustering. While currently the simple hard-labeling methods perform just as well, we hope that the ease of incorporating new features into the K-means labeling method will spur interesting future research.

When considering the constraints and independence relationships implied by each labeling approach, we can distinguish between approaches that label rules differently within the context of the sentence that they were extracted from, and those that do not. The Syntax system from Zollmann and Venugopal (2006) is at one end of this extreme. A given target span might be labeled differently depending on the syntactic analysis of the sentence that it is a part of. On the other extreme, the clustering based approach labels phrases based on the contained words alone.[8] The POS grammar represents an intermediate point on this spectrum, since POS tags can change based on surrounding words in the sentence; and the position of the K-means model depends on the influence of the phrase contexts on the clustering process. Context *insensitive* labeling has the advantage that there are less alternative left-hand-side labels for initial rules, producing grammars with less rules, whose weights can be more accurately estimated. This could explain the strong performance of the word-clustering based labeling approach.

All source code underlying this work is available under the GNU Lesser General Public License as part of the Hadoop-based 'SAMT' system at:
`www.cs.cmu.edu/~zollmann/samt`

Acknowledgments

We thank Jakob Uszkoreit and Ashish Venugopal for helpful comments and suggestions and Yahoo! for the access to the M45 supercomputing cluster.

[8] Note, however, that the creation of clusters itself did take the context of the clustered words into account.

References

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A Gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of ACL*, Singapore, August.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2).

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, Honolulu, Hawaii, October.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

David Chiang. 2007. Hierarchical phrase based translation. *Computational Linguistics*, 33(2).

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *Proceedings of the European chapter of the Association for Computational Linguistics (EACL)*, pages 59–66.

Trevor Cohn and Phil Blunsom. 2009. A Bayesian model of syntax-directed tree to string grammar induction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Singapore.

Michael Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*.

Hany Hassan, Khalil Sima'an, and Andy Way. 2007. Supertagged phrase-based statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, Prague, Czech Republic, June.

Dan Klein and Christoper Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Reinhard Kneser and Hermann Ney. 1993. Improved clustering techniques for class-based statistical language modelling. In *Proceedings of the 3rd European Conference on Speech Communication and Technology*, pages 973–976, Berlin, Germany.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*.

Roland Kuhn, Boxing Chen, George Foster, and Evan Stratford. 2010. Phrase clustering for smoothing TM probabilities - or, how to extract paraphrases from phrase tables. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 608–616, Beijing, China, August.

Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*.

J. B. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Sydney, Australia.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Masaaki Nagata, Kuniko Saito, Kazuhide Yamamoto, and Kazuteru Ohashi. 2006. A clustered global phrase reordering model for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 713–720.

Franz Josef Och. 1999. An efficient method for determining bilingual word classes. In *Proceedings of the European chapter of the Association for Computational Linguistics (EACL)*, pages 71–76.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Hugo Steinhaus. 1956. Sur la division des corps matériels en parties. *Bull. Acad. Polon. Sci. Cl. III. 4*, pages 801–804.

Ashish Venugopal and Andreas Zollmann. 2009. Grammar based statistical MT on Hadoop: An end-to-end toolkit for large scale PSCFG based MT. *The Prague Bulletin of Mathematical Linguistics*, 91:67–78.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3).

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation, HLT/NAACL*.

Andreas Zollmann and Stephan Vogel. 2010. New parameterizations and features for PSCFG-based machine translation. In *Proceedings of the 4th Workshop on Syntax and Structure in Statistical Translation (SSST)*, Beijing, China.

Andreas Zollmann, Ashish Venugopal, Franz J. Och, and Jay Ponte. 2008. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In *Proceedings of the Conference on Computational Linguistics (COLING)*.

Deciphering Foreign Language

Sujith Ravi and Kevin Knight
University of Southern California
Information Sciences Institute
Marina del Rey, California 90292
{sravi,knight}@isi.edu

Abstract

In this work, we tackle the task of machine translation (MT) without parallel training data. We frame the MT problem as a decipherment task, treating the foreign text as a cipher for English and present novel methods for training translation models from non-parallel text.

1 Introduction

Bilingual corpora are a staple of statistical machine translation (SMT) research. From these corpora, we estimate translation model parameters: word-to-word translation tables, fertilities, distortion parameters, phrase tables, syntactic transformations, etc. Starting with the classic IBM work (Brown et al., 1993), training has been viewed as a maximization problem involving hidden word alignments (a) that are assumed to underlie observed sentence pairs (e, f):

$$\arg\max_{\theta} \prod_{e,f} P_\theta(f|e) \quad (1)$$

$$= \arg\max_{\theta} \prod_{e,f} \sum_{a} P_\theta(f, a|e) \quad (2)$$

Brown et al. (1993) give various formulas that boil $P_\theta(f, a|e)$ down to the specific parameters to be estimated.

Of course, for many language pairs and domains, parallel data is not available. In this paper, we address the problem of *learning a full translation model from non-parallel data*, and we use the learned model to translate new foreign strings. As successful work develops along this line, we expect more domains and language pairs to be conquered by SMT.

How can we learn a translation model from non-parallel data? Intuitively, we try to construct translation model tables which, when applied to observed foreign text, consistently yield sensible English. This is essentially the same approach taken by cryptanalysts and epigraphers when they deal with source texts.

In our case, we observe a large number of foreign strings f, and we apply maximum likelihood training:

$$\arg\max_{\theta} \prod_{f} P_\theta(f) \quad (3)$$

Following Weaver (1955), we imagine that this corpus of foreign strings "is really written in English, but has been coded in some strange symbols," thus:

$$\arg\max_{\theta} \prod_{f} \sum_{e} P(e) \cdot P_\theta(f|e) \quad (4)$$

The variable e ranges over all possible English strings, and $P(e)$ is a language model built from large amounts of English text that is unrelated to the foreign strings. Re-writing for hidden alignments, we get:

$$\arg\max_{\theta} \prod_{f} \sum_{e} P(e) \cdot \sum_{a} P_\theta(f, a|e) \quad (5)$$

Note that this formula has the same free $P_\theta(f, a|e)$ parameters as expression (2). We seek to manipulate these parameters in order to learn the

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 12–21,
Portland, Oregon, June 19-24, 2011.

same full translation model. We note that for each f, not only is the alignment a still hidden, but now the English translation e is hidden as well.

A language model $P(e)$ is typically used in SMT decoding (Koehn, 2009), but here $P(e)$ actually plays a central role in training translation model parameters. To distinguish the two, we refer to (5) as *decipherment*, rather than decoding.

We can now draw on previous decipherment work for solving simpler substitution/transposition ciphers (Bauer, 2006; Knight et al., 2006). We must keep in mind, however, that foreign language is a much more demanding code, involving highly nondeterministic mappings and very large substitution tables.

The contributions of this paper are therefore:

- We give first results for training a full translation model from non-parallel text, and we apply the model to translate previously-unseen text. This work is thus distinguished from prior work on extracting or augmenting partial lexicons using non-parallel corpora (Rapp, 1995; Fung and McKeown, 1997; Koehn and Knight, 2000; Haghighi et al., 2008). It also contrasts with self-training (McClosky et al., 2006), which requires a parallel seed and often does not engage in iterative maximization.

- We develop novel methods to deal with large-scale vocabularies inherent in MT problems.

2 Word Substitution Decipherment

Before we tackle machine translation without parallel data, we first solve a simpler problem—word substitution decipherment. Here, we do not have to worry about hidden alignments since there is only one alignment. In a word substitution cipher, every word in the natural language (plaintext) sequence is substituted by a cipher token, according to a substitution key. The key is deterministic—there exists a 1-to-1 mapping between cipher units and the plaintext words they encode.

For example, the following English plaintext sequences:

```
I SAW THE BOY .
THE BOY RAN .
```

may be enciphered as:

```
xyzz fxyy crqq tmnz lxwz
crqq tmnz gdxx lxwz
```

according to the key:

```
THE → crqq, SAW → fxyy, RAN → gdxx,
.  → lxwz, BOY → tmnz, I → xyzz
```

The goal of word substitution decipherment is to guess the original plaintext from given cipher data without any knowledge of the substitution key.

Word substitution decipherment is a good test-bed for unsupervised statistical NLP techniques for two reasons—(1) we face large vocabularies and corpora sizes typically seen in large-scale MT problems, so our methods need to scale well, (2) similar decipherment techniques can be applied for solving NLP problems such as unsupervised part-of-speech tagging.

Probabilistic decipherment: Our decipherment method follows a noisy-channel approach. We first model the process by which the ciphertext sequence $c = c_1...c_n$ is generated. The generative story for decipherment is described here:

1. Generate an English plaintext sequence $e = e_1...e_n$, with probability $P(e)$.

2. Substitute each plaintext word e_i with a ciphertext token c_i, with probability $P_\theta(c_i|e_i)$ in order to generate the ciphertext sequence $c = c_1...c_n$.

We model $P(e)$ using a statistical word n-gram English language model (LM). During decipherment, our goal is to estimate the channel model parameters θ. Re-writing Equations 3 and 4 for word substitution decipherment, we get:

$$\arg\max_\theta \prod_c P_\theta(c) \tag{6}$$

$$= \arg\max_\theta \prod_c \sum_e P(e) \cdot \prod_{i=1}^{n} P_\theta(c_i|e_i) \tag{7}$$

Challenges: Unlike letter substitution ciphers (having only 26 plaintext letters), here we have to deal with large-scale vocabularies (10k-1M word types) and corpora sizes (100k cipher tokens). This poses some serious scalability challenges for word substitution decipherment.

We propose novel methods that can deal with these challenges effectively and solve word substitution ciphers:

1. *EM solution:* We would like to use the Expectation Maximization (EM) algorithm (Dempster et al., 1977) to estimate θ from Equation 7, but EM training is not feasible in our case. First, EM cannot scale to such large vocabulary sizes (running the forward-backward algorithm for each iteration requires $O(V^2)$ time). Secondly, we need to instantiate the entire channel and resulting derivation lattice before we can run EM, and this is too big to be stored in memory. So, we introduce a new training method (*Iterative EM*) that fixes these problems.

2. *Bayesian decipherment:* We also propose a novel decipherment approach using Bayesian inference. Typically, Bayesian inference is very slow when applied to such large-scale problems. Our method overcomes these challenges and does fast, efficient inference using (a) a novel strategy for selecting sampling choices, and (b) a parallelized sampling scheme.

In the next two sections, we describe these methods in detail.

2.1 Iterative EM

We devise a method which overcomes memory and running time efficiency issues faced by EM. Instead of instantiating the entire channel model (with all its parameters), we iteratively train the model in small steps. The training procedure is described here:

1. Identify the top K frequent word types in both the plaintext and ciphertext data. Replace all other word tokens with *Unknown*. Now, instantiate a small channel with just $(K+1)^2$ parameters and use the EM algorithm to train this model to maximize likelihood of cipher data.

2. Extend the plaintext and ciphertext vocabularies from the previous step by adding the next K most frequent word types (so the new vocabulary size becomes $2K+1$). Regenerate the plaintext and ciphertext data.

3. Instantiate a new $(2K+1) \times (2K+1)$ channel model. From the previous EM-trained channel, identify all the $e \rightarrow c$ mappings that were assigned a probability $P(c|e) > 0.5$. Fix these mappings in the new channel, i.e. set $P(c|e) = 1.0$. From the new channel, eliminate all other parameters $e \rightarrow c_j$ associated with the plaintext word type e (where $c_j \neq c$). This yields a much smaller channel with size $< (2K+1)^2$. Retrain the new channel using EM algorithm.

4. Goto Step 2 and repeat the procedure, extending the channel size iteratively in each stage.

Finally, we decode the given ciphertext c by using the Viterbi algorithm to choose the plaintext decoding e that maximizes $P(e) \cdot P_{\theta_{trained}}(c|e)^3$, stretching the channel probabilities (Knight et al., 2006).

2.2 Bayesian Decipherment

Bayesian inference methods have become popular in natural language processing (Goldwater and Griffiths, 2007; Finkel et al., 2005; Blunsom et al., 2009; Chiang et al., 2010; Snyder et al., 2010). These methods are attractive for their ability to manage uncertainty about model parameters and allow one to incorporate prior knowledge during inference.

Here, we propose a novel decipherment approach using Bayesian learning. Our method holds several other advantages over the EM approach—(1) inference using smart sampling strategies permits efficient training, allowing us to scale to large data/vocabulary sizes, (2) incremental scoring of derivations during sampling allows efficient inference even when we use higher-order n-gram LMs, (3) there are no memory bottlenecks since the full channel model and derivation lattice are never instantiated during training, and (4) prior specification allows us to learn *skewed* distributions that are useful here—word substitution ciphers exhibit 1-to-1 correspondence between plaintext and cipher types.

We use the same generative story as before for decipherment, except that we use Chinese Restaurant Process (CRP) formulations for the source and channel probabilities. We use an English word bigram LM as the base distribution (P_0) for the source model and specify a uniform P_0 distribution for the

channel.[1] We perform inference using point-wise Gibbs sampling (Geman and Geman, 1984). We define a sampling operator that samples plaintext word choices for every cipher token, one at a time. Using the exchangeability property, we efficiently score the probability of each derivation in an incremental fashion. In addition, we make further improvements to the sampling procedure which makes it faster.

Smart sample-choice selection: In the original sampling step, for each cipher token we have to sample from a list of all possible plaintext choices (10k-1M English words). There are 100k cipher tokens in our data which means we have to perform $\sim 10^9$ sampling operations to make one entire pass through the data. We have to then repeat this process for 2000 iterations. Instead, we now reduce our choices in each sampling step.

Say that our current plaintext hypothesis contains English words X, Y and Z at positions $i-1$, i and $i+1$ respectively. In order to sample at position i, we choose the top K English words Y ranked by $P(\text{X Y Z})$, which can be computed offline from a statistical word bigram LM. If this probability is 0 (i.e., X and Z never co-occurred), we randomly pick K words from the plaintext vocabulary. We set K = 100 in our experiments. This significantly reduces the sampling possibilities (10k-1M reduces to 100) at each step and allows us to scale to large plaintext vocabulary sizes without enumerating all possible choices at every cipher position.[2]

Parallelized Gibbs sampling: Secondly, we parallelize our sampling step using a Map-Reduce framework. In the past, others have proposed parallelized sampling schemes for topic modeling applications (Newman et al., 2009). In our method, we split the entire corpus into separate chunks and we run the sampling procedure on each chunk in parallel. At the end of each sampling iteration, we combine the samples corresponding to each chunk and collect the counts of all events—this forms our cache for the next sampling iteration. In practice, we observe that the parallelized sampling run converges quickly and runs much faster than the conventional point-wise sampling—for example, 3.1 hours (using 10 nodes) versus 11 hours for one of the word substitution experiments. We also notice a higher speedup when scaling to larger vocabularies.[3]

Decoding the ciphertext: After the sampling run has finished, we choose the final sample and extract a trained version of the channel model $P_\theta(c|e)$ from this sample following the technique of Chiang et al. (2010). We then use the Viterbi algorithm to choose the English plaintext e that maximizes $P(e) \cdot P_{\theta_{trained}}(c|e)^3$.

2.3 Experiments and Results

Data: For the word substitution experiments, we use two corpora:

- *Temporal expression corpus* containing short English temporal expressions such as "THE NEXT MONTH", "THE LAST THREE YEARS", etc. The cipher data contains 5000 expressions (9619 tokens, 153 word types). We also have access to a separate English corpus (which is not parallel to the ciphertext) containing 125k temporal expressions (242k word tokens, 201 word types) for LM training.
- *Transtac corpus* containing full English sentences. The data consists of 10k cipher sentences (102k tokens, 3397 word types); and a plaintext corpus of 402k English sentences (2.7M word tokens, 25761 word types) for LM training. We use all the cipher data for decipherment training but evaluate on the first 1000 cipher sentences.

The cipher data was originally generated from English text by substituting each English word with a unique cipher word. We use the plaintext corpus to

[1] For word substitution decipherment, we want to keep the language model probabilities fixed during training, and hence we set the prior on that model to be high ($\alpha = 10^4$). We use a sparse Dirichlet prior for the channel ($\beta = 0.01$). We use the output from Iterative EM decoding (using 101 x 101 channel) as initial sample and run the sampler for 2000 iterations. During sampling, we use a linear annealing schedule decreasing the temperature from $1 \rightarrow 0.08$.

[2] Since we now sample from an approximate distribution, we have to correct this with the Metropolis-Hastings algorithm. But in practice we observe that samples from our proposal distribution are accepted with probability > 0.99, so we skip this step.

[3] *Type sampling* could be applied on top of our methods to further optimize performance. But more complex problems like MT do not follow the same principles (1-to-1 key mappings) as seen in word substitution ciphers, which makes it difficult to identify type dependencies.

Method	Decipherment Accuracy (%)		
	Temporal expr.	*Transtac*	
		9k	100k
0. EM with 2-gram LM	87.8	Intractable	
1. Iterative EM with 2-gram LM	87.8	70.5	71.8
2. Bayesian with 2-gram LM	**88.6**	60.1	80.0
with 3-gram LM	-		**82.5**

Figure 1: Comparison of word substitution decipherment results using (1) Iterative EM, and (2) Bayesian method. For the *Transtac* corpus, decipherment performance is also shown for different training data sizes (9k versus 100k cipher tokens).

build an English word n-gram LM, which is used in the decipherment process.

Evaluation: We compute the accuracy of a particular decipherment as the percentage of cipher tokens that were correctly deciphered from the whole corpus. We run the two methods (Iterative EM[4] and Bayesian) and then compare them in terms of word substitution decipherment accuracies.

Results: Figure 1 compares the word substitution results from Iterative EM and Bayesian decipherment. Both methods achieve high accuracies, decoding 70-90% of the two word substitution ciphers. Overall, Bayesian decipherment (with sparse priors) performs better than Iterative EM and achieves the best results on this task. We also observe that both methods benefit from better LMs and more (cipher) training data. Figure 2 shows sample outputs from Bayesian decipherment.

3 Machine Translation as a Decipherment Task

We now turn to the problem of MT without parallel data. From a decipherment perspective, machine translation is a much more complex task than word substitution decipherment and poses several technical challenges: (1) scalability due to large corpora sizes and huge translation tables, (2) nondeterminism in translation mappings (a word can have multiple translations), (3) re-ordering of words

[4]For Iterative EM, we start with a channel of size 101x101 (K=100) and in every pass we iteratively increase the vocabulary sizes by 50, repeating the training procedure until the channel size becomes 351x351.

C:	*3894 9411 4357 8446 5433*
O:	a diploma that's good .
D:	a **fence** that's good .
C:	*8593 7932 3627 9166 3671*
O:	three families living here ?
D:	three **brothers** living here ?
C:	*6283 8827 7592 6959 5120 6137 9723 3671*
O:	okay and what did they tell you ?
D:	okay and what did they tell you ?
C:	*9723 3601 5834 5838 3805 4887 7961 9723 3174 4518 9067 4488 9551 7538 7239 9166 3671*
O:	you mean if we come to see you in the afternoon after five you'll be here ?
D:	**i** mean if we come to see you in the afternoon after **thirty** you'll be here ?
	...

Figure 2: Comparison of the original (O) English plaintext with output from Bayesian word substitution decipherment (D) for a few samples cipher (C) sentences from the *Transtac* corpus.

or phrases, (4) a single word can translate into a phrase, and (5) insertion/deletion of words.

Problem Formulation: We formulate the MT decipherment problem as—given a foreign text f (i.e., foreign word sequences $f_1 ... f_m$) and a monolingual English corpus, our goal is to decipher the foreign text and produce an English translation.

Probabilistic decipherment: Unlike parallel training, here we have to estimate the translation model $P_\theta(f|e)$ parameters using only monolingual data. During decipherment training, our objective is to estimate the model parameters θ in order to maximize the probability of the foreign corpus f. From Equation 4 we have:

$$\arg\max_\theta \prod_f \sum_e P(e) \cdot P_\theta(f|e)$$

For $P(e)$, we use a word n-gram LM trained on monolingual English data. We then estimate parameters of the translation model $P_\theta(f|e)$ during training. Next, we present two novel decipherment approaches for MT training without parallel data.

1. *EM Decipherment:* We propose a new translation model for MT decipherment which can be efficiently trained using the EM algorithm.

2. *Bayesian Decipherment:* We introduce a novel method for estimating IBM Model 3 parameters without parallel data, using Bayesian learning. Unlike EM, this method does not face any

memory issues and we use sampling to perform efficient inference during training.

3.1 EM Decipherment

For the translation model $P_\theta(f|e)$, we would like to use a well-known statistical model such as IBM Model 3 and subsequently train it using the EM algorithm. But without parallel training data, EM training for IBM Model 3 becomes intractable due to (1) scalability and efficiency issues because of large-sized fertility and distortion parameter tables, and (2) the resulting derivation lattices become too big to be stored in memory.

Instead, we propose a simpler generative story for MT without parallel data. Our model accounts for (word) substitutions, insertions, deletions and local re-ordering during the translation process but does not incorporate fertilities or global re-ordering. We describe the generative process here:

1. Generate an English string $e = e_1...e_l$, with probability $P(e)$.
2. Insert a NULL word at any position in the English string, with uniform probability.
3. For each English word token e_i (including NULLs), choose a foreign word translation f_i, with probability $P_\theta(f_i|e_i)$. The foreign word may be NULL.
4. Swap any pair of adjacent foreign words f_{i-1}, f_i, with probability $P_\theta(swap)$. We set this value to 0.1.
5. Output the foreign string $f = f_1...f_m$, skipping over NULLs.

We use the EM algorithm to estimate all the parameters θ in order to maximize likelihood of the foreign corpus. Finally, we use the Viterbi algorithm to decode the foreign sentence f and produce an English translation e that maximizes $P(e) \cdot P_{\theta_{trained}}(f|e)$.

Linguistic knowledge for decipherment: To help limit translation model size and deal with data sparsity problem, we use prior linguistic knowledge. We use identity mappings for numeric values (for example, "8" maps to "8"), and we split nouns into morpheme units prior to decipherment training (for example, "YEARS" → "YEAR" "+S").

Whole-segment Language Models: When using word n-gram models of English for decipherment, we find that some of the foreign sentences are decoded into sequences (such as "THANK YOU TALKING ABOUT ?") that are not good English. This stems from the fact that n-gram LMs have no global information about what constitutes a valid English segment. To learn this information automatically, we build a $P(e)$ model that only recognizes English whole-segments (entire sentences or expressions) observed in the monolingual training data. We then use this model (in place of word n-gram LMs) for decipherment training and decoding.

3.2 Bayesian Method

Brown et al. (1993) provide an efficient algorithm for training IBM Model 3 translation model when parallel sentence pairs are available. But we wish to perform IBM Model 3 training under non-parallel conditions, which is intractable using EM training. Instead, we take a Bayesian approach.

Following Equation 5, we represent the translation model as $P_\theta(f, a|e)$ in terms of hidden alignments a. Recall the generative story for IBM Model 3 translation which has the following formula:

$$\begin{aligned} P_\theta(f,a|e) &= \prod_{i=0}^{l} t_\theta(f_{a_j}|e_i) \cdot \prod_{i=1}^{l} n_\theta(\phi_i|e_i) \\ &\cdot \prod_{a_j \neq 0, j=1}^{m} d_\theta(a_j|i,l,m) \cdot \prod_{i=0}^{l} \phi_i! \\ &\cdot \frac{1}{\phi_0!} \cdot \binom{m-\phi_0}{\phi_0} \\ &\cdot p_{1_\theta}^{\phi_0} \cdot p_{0_\theta}^{m-2\phi_0} \end{aligned} \quad (8)$$

The alignment a is represented as a vector; $a_j = i$ implies that the foreign word f_j is produced by the English word e_i during translation.

Bayesian Formulation: Our goal is to learn the probability tables t (translation parameters) n (fertility parameters), d (distortion parameters), and p (English NULL word probabilities) without parallel data. In order to apply Bayesian inference for decipherment, we model each of these tables using a

Chinese Restaurant Process (CRP) formulation. For example, to model the translation probabilities, we use the formula:

$$t_\theta(f_j|e_i) = \frac{\alpha \cdot P_0(f_j|e_i) + C_{history}(e_i, f_j)}{\alpha + C_{history}(e_i)} \quad (9)$$

where, P_0 represents the base distribution (which is set to uniform) and $C_{history}$ represents the count of events occurring in the history (cache). Similarly, we use CRP formulations for the other probabilities (n, d and p). We use sparse Dirichlet priors for all these models (i.e., low values for α) and plug these probabilities into Equation 8 to get $P_\theta(f, a|e)$.

Sampling IBM Model 3: We use point-wise Gibbs sampling to estimate the IBM Model 3 parameters. The sampler is seeded with an initial English sample translation and a corresponding alignment for every foreign sentence. We define several sampling operators, which are applied in sequence one after the other to generate English samples for the entire foreign corpus. Some of the sampling operators are described below:

- TranslateWord(j): Sample a new English word translation for foreign word f_j, from all possibilities (including NULL).
- SwapSegment(i_1, i_2): Swap the alignment links for English words e_{i_1} and e_{i_2}.
- JoinWords(i_1, i_2): Eliminate the English word e_{i_1} and transfer its links to the word e_{i_2}.

During sampling, we apply each of these operators to generate a new derivation e, a for the foreign text f and compute its score as $P(e) \cdot P_\theta(f, a|e)$. These small-change operators are similar to the heuristic techniques used for greedy decoding by German et al. (2001). But unlike the greedy method, which can easily get stuck, our Bayesian approach guarantees that once the sampler converges we will be sampling from the true *posterior* distribution.

As with Bayesian decipherment for word substitution, we compute the probability of each new derivation incrementally, which makes sampling efficient. We also apply blocked sampling on top of point-wise sampling—we treat all occurrences of a particular foreign sentence as a single block and sample a single derivation for the entire block. We also parallelize the sampling procedure (as described in Section 2.2).[5]

Choosing the best translation: Once the sampling run finishes, we select the final sample and extract the corresponding English translations for every foreign sentence. This yields the final decipherment output.

3.3 MT Experiments and Results

Data: We work with the Spanish/English language pair and use the following corpora in our MT experiments:

- *Time corpus:* We mined English newswire text on the Web and collected 295k temporal expressions such as "LAST YEAR", "THE FOURTH QUARTER", "IN JAN 1968", etc. We first process the data and normalize numbers and names of months/weekdays—for example, "1968" is replaced with "NNNN", "JANUARY" with "[MONTH]", and so on. We then translate the English temporal phrases into Spanish using an automatic translation software (Google Translate) followed by manual annotation to correct mistakes made by the software. We create the following splits out of the resulting parallel corpus:

  TRAIN (English): 195k temporal expressions (7588 unique), 382k word tokens, 163 types.

  TEST (Spanish): 100k temporal expressions (2343 unique), 204k word tokens, 269 types.

- *OPUS movie subtitle corpus:* This is a large open source collection of parallel corpora available for multiple language pairs (Tiedemann, 2009). We downloaded the parallel Spanish/English subtitle corpus which consists of aligned Spanish/English sentences from a collection of movie subtitles. For our MT experiments, we select only Spanish/English sentences with frequency > 10 and create the following train/test splits:

[5] For Bayesian MT decipherment, we set a high prior value on the language model (10^4) and use sparse priors for the IBM 3 model parameters t, n, d, p $(0.01, 0.01, 0.01, 0.01)$. We use the output from EM decipherment as the initial sample and run the sampler for 2000 iterations, during which we apply annealing with a linear schedule ($2 \rightarrow 0.08$).

Method	Decipherment Accuracy	
	Time expressions	*OPUS subtitles*
1a. *Parallel training* (MOSES)		
with 2-gram LM	5.6 (85.6)	26.8 (63.6)
with 5-gram LM	4.7 (88.0)	
1b. *Parallel training* (IBM 3 without distortion)		
with 2-gram LM	10.1 (78.9)	29.9 (59.6)
with whole-segment LM	9.0 (79.2)	
2a. *Decipherment* (EM)		
with 2-gram LM	37.6 (44.6)	67.2 (15.3)
with whole-segment LM	28.7 (48.7)	65.1 (19.3)
2b. *Decipherment* (Bayesian IBM 3)		
with 2-gram LM	34.0 (30.2)	66.6 (15.1)

Figure 3: Comparison of Spanish/English MT performance on the *Time* and *OPUS* test corpora achieved by various MT systems trained under (1) **parallel**—(a) MOSES, (b) IBM 3 without distortion, and (2) **decipherment** settings—(a) EM, (b) Bayesian. The scores reported here are normalized edit distance values with BLEU scores shown in parentheses.

> **TRAIN** (English): 19770 sentences (1128 unique), 62k word tokens, 411 word types.
>
> **TEST** (Spanish): 13181 sentences (1127 unique), 39k word tokens, 562 word types.

Both Spanish/English sides of **TRAIN** are used for parallel MT training, whereas decipherment uses only monolingual English data for training LMs.

MT Systems: We build and compare different MT systems under two training scenarios:

1. *Parallel training* using: (a) **MOSES**, a phrase translation system (Koehn et al., 2007) widely used in MT literature, and (b) a simpler version of **IBM Model 3 (without distortion parameters)** which can be trained tractably using the strategy of Knight and Al-Onaizan (1998).
2. *Decipherment without parallel data* using: (a) **EM method** (from Section 3.1), and (b) **Bayesian method** (from Section 3.2).

Evaluation: All the MT systems are run on the Spanish test data and the quality of the resulting English translations are evaluated using two different measures—(1) Normalized edit distance score (Navarro, 2001),[6] and (2) BLEU (Papineni et al., 2002), a standard MT evaluation measure.

Results: Figure 3 compares the results of various MT systems (using parallel versus decipherment training) on the two test corpora in terms of edit distance scores (a lower score indicates closer match to the gold translation). The figure also shows the corresponding BLEU scores in parentheses for comparison (higher scores indicate better MT output).

We observe that even without parallel training data, our decipherment strategies achieve MT accuracies comparable to parallel-trained systems. On the *Time* corpus, the best decipherment (Method 2a in the figure) achieves an edit distance score of 28.7 (versus 4.7 for MOSES). Better LMs yield better MT results for both parallel and decipherment training—for example, using a segment-based English LM instead of a 2-gram LM yields a 24% reduction in edit distance and a 9% improvement in BLEU score for EM decipherment.

We also investigate how the performance of different MT systems vary with the size of the training data. Figure 4 plots the BLEU scores versus training sizes for different MT systems on the *Time* corpus. Clearly, using more training data yields better performance for all systems. However, higher improvements are observed when using parallel data in comparison to decipherment training which only uses monolingual data. We also notice that the scores do not improve much when going beyond 10,000 train-

[6] When computing edit distance, we account for substitutions, insertions, deletions as well as local-swap edit operations required to convert a given English string into the (gold) reference translation.

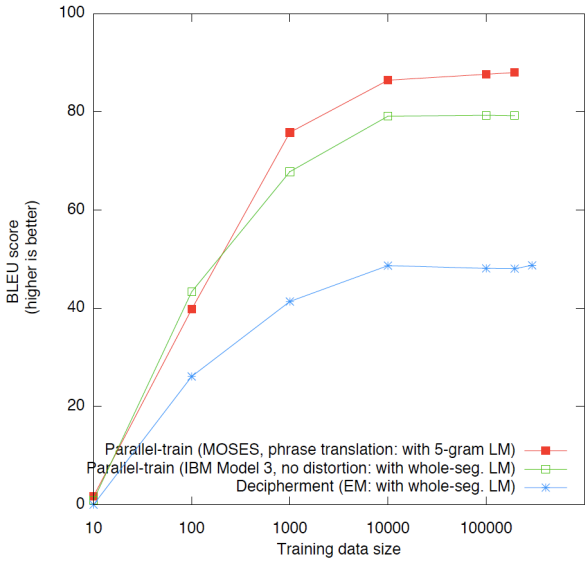


Figure 4: Comparison of *training data size* versus *MT accuracy* in terms of BLEU score under different training conditions: (1) **Parallel training**—(a) MOSES, (b) IBM Model 3 without distortion, and (2) **Decipherment** without parallel data using EM method (from Section 3.1).

ing instances for this domain.

It is interesting to quantify the value of parallel versus non-parallel data for any given MT task. In other words, "*how much non-parallel data is worth how much parallel data in order to achieve the same MT accuracy?*" Figure 4 provides a reasonable answer to this question for the Spanish/English MT task described here. We see that deciphering with 10k monolingual Spanish sentences yields the same performance as training with around 200-500 parallel English/Spanish sentence pairs. This is the first attempt at such a quantitative comparison for MT and our results are encouraging. We envision that further developments in unsupervised methods will help reduce this gap further.

4 Conclusion

Our work is the first attempt at doing MT without parallel data. We discussed several novel decipherment approaches for achieving this goal. Along the way, we developed efficient training methods that can deal with large-scale vocabularies and data sizes. For future work, it will be interesting to see if we can exploit both parallel and non-parallel data to improve on both.

Acknowledgments

This material is based in part upon work supported by the National Science Foundation (NSF) under Grant No. IIS-0904684 and the Defense Advanced Research Projects Agency (DARPA) through the Department of Interior/National Business Center under Contract No. NBCHD040058. Any opinion, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA), or the Department of the Interior/National Business Center.

References

Friedrich L. Bauer. 2006. *Decrypted Secrets: Methods and Maxims of Cryptology*. Springer-Verlag.

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A Gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*, pages 782–790.

Peter Brown, Vincent Della Pietra, Stephen Della Pietra, and Robert Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

David Chiang, Jonathan Graehl, Kevin Knight, Adam Pauls, and Sujith Ravi. 2010. Bayesian inference for finite-state transducers. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL/HLT)*, pages 447–455.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Jenny Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 363–370.

Pascal Fung and Kathleen McKeown. 1997. Finding terminology translations from non-parallel corpora. In *Proceedings of the Fifth Annual Workshop on Very Large Corpora*, pages 192–202.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741.

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

Sharon Goldwater and Thomas Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics - Human Language Technologies (ACL/HLT)*, pages 771–779.

Kevin Knight and Yaser Al-Onaizan. 1998. Translation with finite-state devices. In David Farwell, Laurie Gerber, and Eduard Hovy, editors, *Machine Translation and the Information Soup*, volume 1529 of *Lecture Notes in Computer Science*, pages 421–437. Springer Berlin / Heidelberg.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics*, pages 499–506.

Philipp Koehn and Kevin Knight. 2000. Estimating word translation probabilities from unrelated monolingual corpora using the EM algorithm. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 711–715.

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

Philip Koehn. 2009. *Statistical Machine Translation*. Cambridge University Press.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 152–159.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM Computing Surveys*, 33:31–88, March.

David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2009. Distributed algorithms for topic models. *Journal of Machine Learning Research*, 10:1801–1828.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318.

Reinhard Rapp. 1995. Identifying word translations in non-parallel texts. In *Proceedings of the Conference of the Association for Computational Linguistics*, pages 320–322.

Benjamin Snyder, Regina Barzilay, and Kevin Knight. 2010. A statistical model for lost language decipherment. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1048–1057.

Jörg Tiedemann. 2009. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, editors, *Recent Advances in Natural Language Processing*, volume V, pages 237–248. John Benjamins, Amsterdam/Philadelphia.

Warren Weaver. 1955. Translation (1949). Reproduced in W.N. Locke, A.D. Booth (eds.). In *Machine Translation of Languages*, pages 15–23. MIT Press.

Effective Use of Function Words for Rule Generalization in Forest-Based Translation

Xianchao Wu[†] **Takuya Matsuzaki**[†] **Jun'ichi Tsujii**[†‡*]

[†]Department of Computer Science, The University of Tokyo
[‡]School of Computer Science, University of Manchester
[*]National Centre for Text Mining (NaCTeM)
{wxc, matuzaki, tsujii}@is.s.u-tokyo.ac.jp

Abstract

In the present paper, we propose the effective usage of function words to generate generalized translation rules for forest-based translation. Given aligned forest-string pairs, we extract composed tree-to-string translation rules that account for multiple interpretations of both aligned and unaligned target function words. In order to constrain the exhaustive attachments of function words, we limit to bind them to the nearby syntactic chunks yielded by a target dependency parser. Therefore, the proposed approach can not only capture source-tree-to-target-chunk correspondences but can also use forest structures that compactly encode an exponential number of parse trees to properly generate target function words during decoding. Extensive experiments involving large-scale English-to-Japanese translation revealed a significant improvement of 1.8 points in BLEU score, as compared with a strong forest-to-string baseline system.

1 Introduction

Rule generalization remains a key challenge for current syntax-based statistical machine translation (SMT) systems. On the one hand, there is a tendency to integrate richer syntactic information into a translation rule in order to better express the translation phenomena. Thus, flat phrases (Koehn et al., 2003), hierarchical phrases (Chiang, 2005), and syntactic tree fragments (Galley et al., 2006; Mi and Huang, 2008; Wu et al., 2010) are gradually used in SMT. On the other hand, the use of syntactic phrases continues due to the requirement for phrase coverage in most syntax-based systems. For example, Mi et al. (2008) achieved a 3.1-point improvement in BLEU score (Papineni et al., 2002) by including bilingual syntactic phrases in their forest-based system. Compared with flat phrases, syntactic rules are good at capturing global reordering, which has been reported to be essential for translating between languages with substantial structural differences, such as English and Japanese, which is a subject-object-verb language (Xu et al., 2009).

Forest-based translation frameworks, which make use of packed parse forests on the source and/or target language side(s), are an increasingly promising approach to syntax-based SMT, being both algorithmically appealing (Mi et al., 2008) and empirically successful (Mi and Huang, 2008; Liu et al., 2009). However, forest-based translation systems, and, in general, most linguistically syntax-based SMT systems (Galley et al., 2004; Galley et al., 2006; Liu et al., 2006; Zhang et al., 2007; Mi et al., 2008; Liu et al., 2009; Chiang, 2010), are built upon word aligned parallel sentences and thus share a critical dependence on word alignments. For example, even a single spurious word alignment can invalidate a large number of otherwise extractable rules, and unaligned words can result in an exponentially large set of extractable rules for the interpretation of these unaligned words (Galley et al., 2006).

What makes word alignment so fragile? In order to investigate this problem, we manually analyzed the alignments of the first 100 parallel sentences in our English-Japanese training data (to be shown in Table 2). The alignments were generated by running GIZA++ (Och and Ney, 2003) and the *grow-diag-final-and* symmetrizing strategy (Koehn et al., 2007) on the training set. Of the 1,324 word alignment pairs, there were 309 error pairs, among

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 22–31,
Portland, Oregon, June 19-24, 2011.

which there were 237 target function words, which account for 76.7% of the error pairs[1]. This indicates that the alignments of the function words are more easily to be mistaken than content words. Moreover, we found that most Japanese function words tend to align to a few English words such as 'of' and 'the', which may appear anywhere in an English sentence. Following these problematic alignments, we are forced to make use of relatively large English tree fragments to construct translation rules that tend to be ill-formed and less generalized.

This is the motivation of the present approach of re-aligning the target function words to source tree fragments, so that the influence of incorrect alignments is reduced and the function words can be generated by tree fragments on the fly. However, the current dominant research only uses 1-best trees for syntactic realignment (Galley et al., 2006; May and Knight, 2007; Wang et al., 2010), which adversely affects the rule set quality due to parsing errors. Therefore, we realign target function words to a packed forest that compactly encodes exponentially many parses. Given aligned forest-string pairs, we extract composed tree-to-string translation rules that account for multiple interpretations of both aligned and unaligned target function words. In order to constrain the exhaustive attachments of function words, we further limit the function words to bind to their surrounding chunks yielded by a dependency parser. Using the composed rules of the present study in a baseline forest-to-string translation system results in a 1.8-point improvement in the BLEU score for large-scale English-to-Japanese translation.

2 Backgrounds

2.1 Japanese function words

In the present paper, we limit our discussion on Japanese particles and auxiliary verbs (Martin, 1975). Particles are suffixes or tokens in Japanese grammar that immediately follow modified content words or sentences. There are eight types of Japanese function words, which are classified depending on what function they serve: case markers, parallel markers, sentence ending particles, interjectory particles, adverbial particles, binding particles, conjunctive particles, and phrasal particles.

Japanese grammar also uses auxiliary verbs to give further semantic or syntactic information about the preceding main or full verb. Alike English, the extra meaning provided by a Japanese auxiliary verb alters the basic meaning of the main verb so that the main verb has one or more of the following functions: passive voice, progressive aspect, perfect aspect, modality, dummy, or emphasis.

2.2 HPSG forests

Following our precious work (Wu et al., 2010), we use head-drive phrase structure grammar (HPSG) forests generated by Enju[2] (Miyao and Tsujii, 2008), which is a state-of-the-art HPSG parser for English. HPSG (Pollard and Sag, 1994; Sag et al., 2003) is a lexicalist grammar framework. In HPSG, linguistic entities such as words and phrases are represented by a data structure called a *sign*. A sign gives a factored representation of the syntactic features of a word/phrase, as well as a representation of their semantic content. Phrases and words represented by signs are collected into larger phrases by the applications of *schemata*. The semantic representation of the new phrase is calculated at the same time. As such, an HPSG parse forest can be considered to be a forest of signs. Making use of these signs instead of part-of-speech (POS)/phrasal tags in PCFG results in a fine-grained rule set integrated with deep syntactic information.

For example, an aligned HPSG forest[3]-string pair is shown in Figure 1. For simplicity, we only draw the identifiers for the signs of the nodes in the HPSG forest. Note that the identifiers that start with 'c' denote non-terminal nodes (e.g., c0, c1), and the identifiers that start with 't' denote terminal nodes (e.g., t3, t1). In a complete HPSG forest given in (Wu et al., 2010), the terminal signs include features such as the POS tag, the tense, the auxiliary, the voice of a verb, etc.. The non-terminal signs include features such as the phrasal category, the name of the schema

[1]These numbers are language/corpus-dependent and are not necessarily to be taken as a general reflection of the overall quality of the word alignments for arbitrary language pairs.

[2]http://www-tsujii.is.s.u-tokyo.ac.jp/enju/index.html

[3]The forest includes three parse trees rooted at c0, c1, and c2. In the 1-best tree, 'by' modifies the passive verb 'verified'. Yet in the 2- and 3-best tree, 'by' modifies 'this result was verified'. Furthermore, 'verified' is an adjective in the 2-best tree and a passive verb in the 3-best tree.

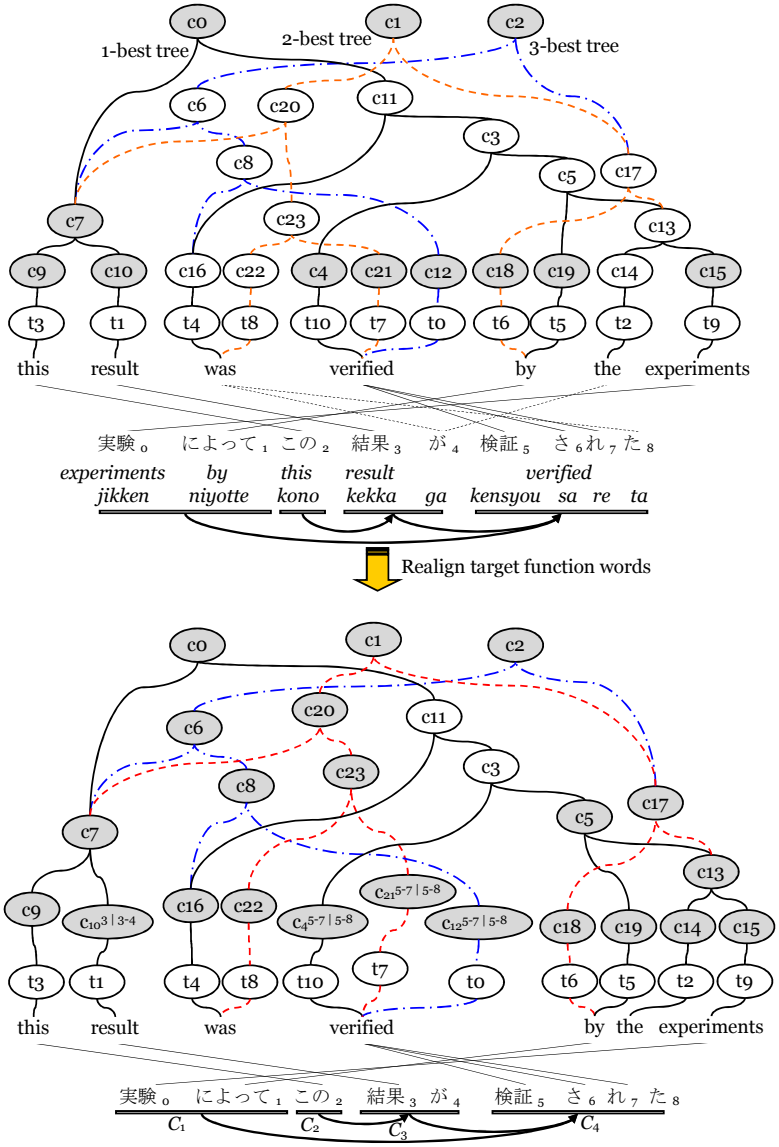


Figure 1: Illustration of an aligned HPSG forest-string pair for English-to-Japanese translation. The chunk-level dependency tree for the Japanese sentence is shown as well.

applied in the node, etc..

3 Composed Rule Extraction

In this section, we first describe an algorithm that attaches function words to a packed forest guided by target chunk information. That is, given a triple $\langle F_S, T, A \rangle$, namely an aligned ($A$) source forest ($F_S$) to target sentence ($T$) pair, we 1) tailor the alignment A by removing the alignments for target function words, 2) seek attachable nodes in the source forest F_S for each function word, and 3) construct a derivation forest by topologically traversing F_S. Then, we identify minimal and composed rules from the derivation forest and estimate the probabilities of rules and scores of derivations using the expectation-maximization (EM) (Dempster et al., 1977) algorithm.

3.1 Definitions

In the proposed algorithm, we make use of the following definitions, which are similar to those described in (Galley et al., 2004; Mi and Huang, 2008):

- $s(\cdot)$: the *span* of a (source) node v or a (target) chunk $\mathcal{C}$, which is an index set of the words that

v or $\mathcal{C}$ covers;

- $t(v)$: the *corresponding span* of v, which is an index set of aligned words on another side;
- $c(v)$: the *complement span* of v, which is the union of corresponding spans of nodes v' that share an identical parse tree with v but are neither antecedents nor descendants of v;
- $\mathcal{P}_A$: the *frontier set* of F_S, which contains nodes that are *consistent* with an alignment A (gray nodes in Figure 1), i.e., $t(v) \neq \varnothing$ and $closure(t(v)) \cap c(v) = \varnothing$.

The function $closure$ covers the gap(s) that may appear in the interval parameter. For example, $closure(t(\text{c3})) = closure(\{\text{0-1, 4-7}\}) = \{\text{0-7}\}$. Examples of the applications of these functions can be found in Table 1. Following (Galley et al., 2006), we distinguish between *minimal* and *composed* rules. The composed rules are generated by combining a sequence of minimal rules.

3.2 Free attachment of target function words

3.2.1 Motivation

We explain the motivation for the present research using an example that was extracted from our training data, as shown in Figure 1. In the alignment of this example, three lines (in dot lines) are used to align *was* and *the* with *ga* (subject particle), and *was* with *ta* (past tense auxiliary verb). Under this alignment, we are forced to extract rules with relatively large tree fragments. For example, by applying the GHKM algorithm (Galley et al., 2004), a rule rooted at c0 will take c7, t4, c4, c19, t2, and c15 as the leaves. The final tree fragment, with a height of 7, contains 13 nodes. In order to ensure that this rule is used during decoding, we must generate subtrees with a height of 7 for c0. Suppose that the input forest is binarized and that $|E|$ is the average number of hyperedges of each node, then we must generate $O(|E|^{2^6-1})$ subtrees[4] for c0 in the worst case. Thus, the existence of these rules prevents the generalization ability of the final rule set that is extracted.

In order to address this problem, we tailor the alignment by ignoring these three alignment pairs in dot lines. For example, by ignoring the ambiguous alignments on the Japanese function words, we enlarge the frontier set to include from 12 to 19 of the 24 non-terminal nodes. Consequently, the number of extractable minimal rules increases from 12 (with three reordering rules rooted at c0, c1, and c2) to 19 (with five reordering rules rooted at c0, c1, c2, c5, and c17). With more nodes included in the frontier set, we can extract more minimal and composed monotonic/reordering rules and avoid extracting the less generalized rules with extremely large tree fragments.

3.2.2 Why chunking?

In the proposed algorithm, we use a target chunk set to constrain the *attachment explosion problem* because we use a packed parse forest instead of a 1-best tree, as in the case of (Galley et al., 2006). Multiple interpretations of unaligned function words for an aligned tree-string pair result in a derivation forest. Now, we have a packed parse forest in which each tree corresponds to a derivation forest. Thus, pruning free attachments of function words is practically important in order to extract composed rules from this "(derivation) forest of (parse) forest".

In the English-to-Japanese translation test case of the present study, the target chunk set is yielded by a state-of-the-art Japanese dependency parser, Cabocha v0.53[5] (Kudo and Matsumoto, 2002). The output of Cabocha is a list of *chunks*. A chunk contains roughly one content word (usually the head) and affixed function words, such as case markers (e.g., *ga*) and verbal morphemes (e.g., *sa re ta*, which indicate past tense and passive voice). For example, the Japanese sentence in Figure 1 is separated into four chunks, and the dependencies among these chunks are identified by arrows. These arrows point out the head chunk that the current chunk modifies. Moreover, we also hope to gain a fine-grained alignment among these syntactic chunks and source tree fragments. Thereby, during decoding, we are binding the generation of function words with the generation of target chunks.

[4] For one (binarized) hyperedge e of a node, suppose there are x subtrees in the left tail node and y subtrees in the right tail node. Then the number of subtrees guided by e is $(x+1) \times (y+1)$. Thus, the recursive formula is $N_h = |E|(N_{h-1}+1)^2$, where h is the height of the hypergraph and N_h is the number of subtrees. When $h = 1$, we let $N_h = 0$.

[5] http://chasen.org/∼taku/software/cabocha/

Algorithm 1 Aligning function words to the forest

Input: HPSG forest F_S, target sentence T, word alignment $A = \{(i, j)\}$, target function word set $\{f_w\}$ appeared in T, and target chunk set $\{\mathcal{C}\}$
Output: a derivation forest DF

$A' \leftarrow A \setminus \{(i, s(f_w))\}$ ▷ $f_w \in \{f_w\}$
for each node $v \in \mathcal{P}_{A'}$ in topological order **do**
  $\mathcal{T}_v \leftarrow \varnothing$ ▷ store the corresponding spans of v
  for each function word $f_w \in \{f_w\}$ **do**
    if $f_w \in \mathcal{C}$ and $t(v) \cap (\mathcal{C}) \neq \varnothing$ and f_w are not attached to descendants of v **then**
      append $t(v) \cup \{s(f_w)\}$ to $\mathcal{T}_v$
    end if
  end for
  for each corresponding span $t(v) \in \mathcal{T}_v$ **do**
    $\mathcal{R} \leftarrow$ IDENTIFYMINRULES$(v, t(v), T)$ ▷ range over the hyperedges of v, and discount the factional count of each rule $r \in \mathcal{R}$ by $1/|\mathcal{T}_v|$
    create a node n in DF for each rule $r \in \mathcal{R}$
    create a shared parent node $\oplus$ when $|\mathcal{R}| > 1$
  end for
end for

3.2.3 The algorithm

Algorithm 1 outlines the proposed approach to constructing a derivation forest to include multiple interpretations of target function words. The derivation forest is a hypergraph as previously used in (Galley et al., 2006), to maintain the constraint that one unaligned target word be attached to some node v exactly once in one derivation tree. Starting from a triple $\langle F_S, T, A\rangle$, we first tailor the alignment A to A' by removing the alignments for target function words. Then, we traverse the nodes $v \in \mathcal{P}_{A'}$ in topological order. During the traversal, a function word f_w will be attached to v if 1) $t(v)$ overlaps with the span of the chunk to which f_w belongs, and 2) f_w has not been attached to the descendants of v.

We identify translation rules that take v as the root of their tree fragments. Each tree fragment is a *frontier tree* that takes a node in the frontier set $\mathcal{P}_{\mathcal{A}'}$ of F_S as the root node and non-lexicalized frontier nodes or lexicalized non-frontier nodes as the leaves. Also, a *minimal frontier tree* used in a minimal rule is limited to be a frontier tree such that all nodes other than the root and leaves are non-frontier nodes. We use Algorithm 1 described in (Mi and Huang, 2008) to collect minimal frontier trees rooted at v in F_S. That is, we range over each hyperedges headed at v and continue to expand downward until the current set of hyperedges forms a minimal frontier tree.

	$A \rightarrow (A')$			
node	$s(\cdot)$	$t(\cdot)$	$c(\cdot)$	consistent
c0	0-6	0-8(0-3,5-7)	∅	1
c1	0-6	0-8(0-3,5-7)	∅	1
c2	0-6	0-8(0-3,5-7)	∅	1
c3	3-6	0-1,4-7(0-1, 5-7)	2,8	0
c4	3	5-7	0,8(0-3)	1
c5*	4-6	0,4(0-1)	2-8(2-3,5-7)	0(1)
c6*	0-3	2-8(2-3,5-7)	0,4(0-1)	0(1)
c7	0-1	2-3	0-1,4-8(0-1,5-7)	1
c8*	2-3	4-8(5-7)	0-4(0-3)	0(1)
c9	0	2	0-1,3-8(0-1,3,5-7)	1
c10	1	3	0-2,4-8(0-2,5-7)	1
c11	2-6	0-1,4-8(0-1,5-7)	2-3	0
c12	3	5-7	0,8(0-3)	1
c13*	5-6	0,4(0)	1-8(1-3,5-7)	0(1)
c14	5	4(∅)	0-8(0-3,5-7)	0
c15	6	0	1-8(1-3,5-7)	1
c16	2	4,8(∅)	0-7(0-3,5-7)	0
c17*	4-6	0,4(0-1)	2-8(2-3,5-7)	0(1)
c18	4	1	0,2-8(0,2-3,5-7)	1
c19	4	1	0,2-8(0,2-3,5-7)	1
c20*	0-3	2-8(2-3,5-7)	0,4(0-1)	0(1)
c21	3	5-7	0,8(0-3)	1
c22	2	4,8(∅)	0-7(0-3,5-7)	0
c23*	2-3	4-8(5-7)	0-4(0-3)	0(1)

Table 1: Change of node attributes after alignment modification from A to A' of the example in Figure 1. Nodes with * superscripts are consistent with A' but not consistent with A.

In the derivation forest, we use $\oplus$ nodes to manage minimal/composed rules that share the same node and the same corresponding span. Figure 2 shows some minimal rule and $\oplus$ nodes derived from the example in Figure 1.

Even though we bind function words to their nearby chunks, these function words may still be attached to relative large tree fragments, so that richer syntactic information can be used to predict the function words. For example, in Figure 2, the tree fragments rooted at node c_0^{0-8} can predict *ga* and/or *ta*. The syntactic foundation behind is that, whether to use *ga* as a subject particle or to use *wo* as an object particle depends on both the left-hand-side noun phrase (*kekka*) and the right-hand-side verb (*kensyou sa re ta*). This type of node v' (such as c_0^{0-8}) should satisfy the following two heuristic conditions:

- v' is included in the frontier set $\mathcal{P}_{\mathcal{A}'}$ of F_S, and
- $t(v')$ covers the function word, or v' is the root node of F_S if the function word is the beginning or ending word in the target sentence T.

Starting from this derivation forest with minimal

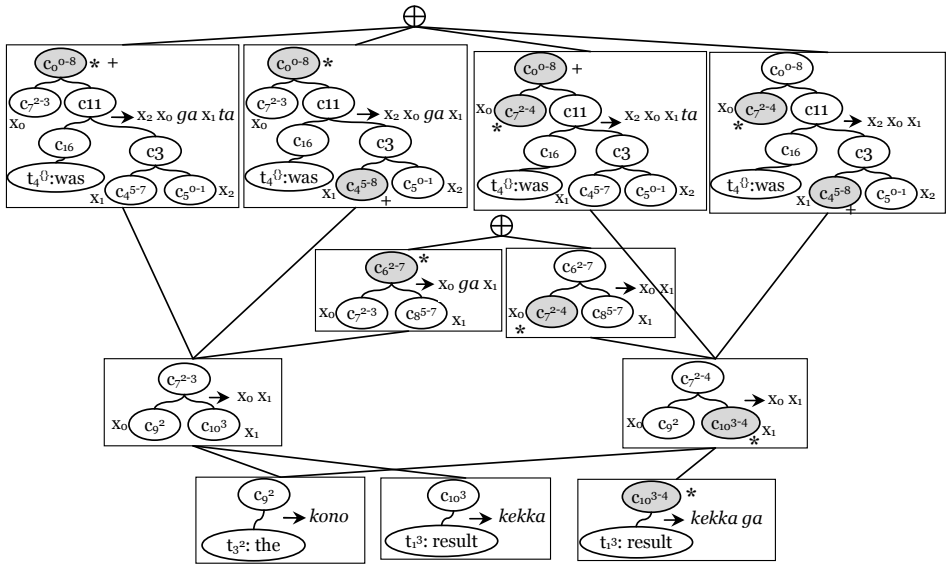


Figure 2: Illustration of a (partial) derivation forest. Gray nodes include some unaligned target function word(s). Nodes annotated by "*" include *ga*, and nodes annotated by "+" include *ta*.

rules as nodes, we can further combine two or more minimal rules to form composed rules nodes and can append these nodes to the derivation forest.

3.3 Estimating rule probabilities

We use the EM algorithm to jointly estimate 1) the translation probabilities and fractional counts of rules and 2) the scores of derivations in the derivation forests. As reported in (May and Knight, 2007), EM, as has been used in (Galley et al., 2006) to estimate rule probabilities in derivation forests, is an iterative procedure and prefers shorter derivations containing large rules over longer derivations containing small rules. In order to overcome this bias problem, we discount the fractional count of a rule by the product of the probabilities of parse hyperedges that are included in the tree fragment of the rule.

4 Experiments

4.1 Setup

We implemented the forest-to-string decoder described in (Mi et al., 2008) that makes use of forest-based translation rules (Mi and Huang, 2008) as the baseline system for translating English HPSG forests into Japanese sentences. We analyzed the performance of the proposed translation rule sets by using the same decoder.

	Train	Dev.	Test
# sentence pairs	994K	2K	2K
# En 1-best trees	987,401	1,982	1,984
# En forests	984,731	1,979	1,983
# En words	24.7M	50.3K	49.9K
# Jp words	28.2M	57.4K	57.1K
# Jp function words	8.0M	16.1K	16.1K

Table 2: Statistics of the JST corpus. Here, En = English and Jp = Japanese.

The JST Japanese-English paper abstract corpus[6] (Utiyama and Isahara, 2007), which consists of one million parallel sentences, was used for training, tuning, and testing. Table 2 shows the statistics of this corpus. Note that Japanese function words occupy more than a quarter of the Japanese words. Making use of Enju 2.3.1, we generated 987,401 1-best trees and 984,731 parse forests for the English sentences in the training set, with successful parse rates of 99.3% and 99.1%, respectively. Using the pruning criteria expressed in (Mi and Huang, 2008), we continue to prune a parse forest by setting p_e to be 8, 5, and 2, until there are no more than $e^{10} = 22,026$ trees in a forest. After pruning, there are an average of 82.3 trees in a parse forest.

[6] http://www.jst.go.jp

	C3-T	M&H-F	Min-F	C3-F
free fw	Y	N	Y	Y
alignment	A'	A	A'	A'
English side	tree	forest	forest	forest
# rule	86.30	96.52	144.91	228.59
# reorder rule	58.50	91.36	92.98	162.71
# tree types	21.62	93.55	72.98	120.08
# nodes/tree	14.2	42.1	26.3	18.6
extract time	30.2	52.2	58.6	130.7
EM time	9.4	-	11.2	29.0
# rules in dev.	0.77	1.22	1.37	2.18
# rules in test	0.77	1.23	1.37	2.15
DT(sec./sent.)	2.8	15.7	22.4	35.4
BLEU (%)	26.15	27.07	27.93	**28.89**

Table 3: Statistics and translation results for four types of tree-to-string rules. With the exception of '# nodes/tree', the numbers in the table are in millions and the time is in hours. Here, fw denotes function word, and DT denotes the decoding time, and the BLEU scores were computed on the test set.

We performed GIZA++ (Och and Ney, 2003) and the *grow-diag-final-and* symmetrizing strategy (Koehn et al., 2007) on the training set to obtain alignments. The SRI Language Modeling Toolkit (Stolcke, 2002) was employed to train a five-gram Japanese LM on the training set. We evaluated the translation quality using the BLEU-4 metric (Papineni et al., 2002).

Joshua v1.3 (Li et al., 2009), which is a freely available decoder for hierarchical phrase-based SMT (Chiang, 2005), is used as an external baseline system for comparison. We extracted 4.5M translation rules from the training set for the 4K English sentences in the development and test sets. We used the default configuration of Joshua, with the exception of the maximum number of items/rules, and the value of k (of the k-best outputs) is set to be 200.

4.2 Results

Table 3 lists the statistics of the following translation rule sets:

- C3-T: a composed rule set extracted from the derivation forests of 1-best HPSG trees that were constructed using the approach described in (Galley et al., 2006). The maximum number of internal nodes is set to be three when generating a composed rule. We free attach target function words to derivation forests;

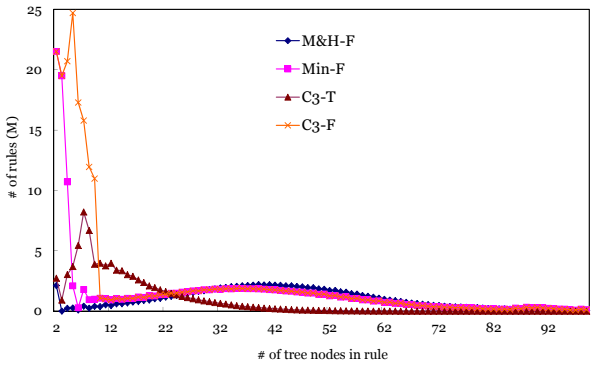


Figure 3: Distributions of the number of tree nodes in the translation rule sets. Note that the curves of Min-F and C3-F are duplicated when the number of tree nodes being larger than 9.

- M&H-F: a minimal rule set extracted from HPSG forests using the extracting algorithm of (Mi and Huang, 2008). Here, we make use of the original alignments. We use the two heuristic conditions described in Section 3.2.3 to attach unaligned words to some node(s) in the forest;

- Min-F: a minimal rule set extracted from the derivation forests of HPSG forests that were constructed using Algorithm 1 (Section 3).

- C3-F: a composed rule set extracted from the derivation forests of HPSG forests. Similar to C3-T, the maximum number of internal nodes during combination is three.

We investigate the generalization ability of these rule sets through the following aspects:

1. the number of rules, the number of reordering rules, and the distributions of the number of tree nodes (Figure 3), i.e., more rules with relatively small tree fragments are preferred;

2. the number of rules that are applicable to the development and test sets (Table 3); and

3. the final translation accuracies.

Table 3 and Figure 3 reflect that the generalization abilities of these four rule sets increase in the order of C3-T $<$ M&H-F $<$ Min-F $<$ C3-F. The advantage of using a packed forest for re-alignment is verified by comparing the statistics of the rules and

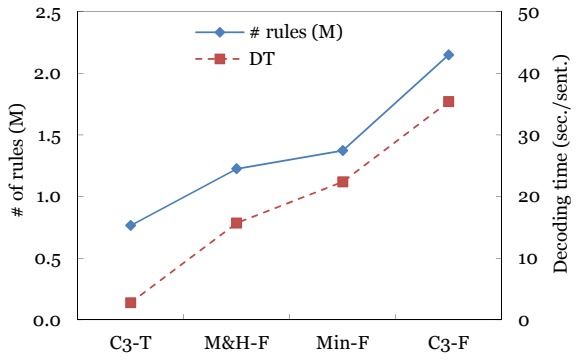


Figure 4: Comparison of decoding time and the number of rules used for translating the test set.

the final BLEU scores of C3-T with Min-F and C3-F. Using the composed rule set C3-F in our forest-based decoder, we achieved an optimal BLEU score of 28.89 (%). Taking M&H-F as the baseline translation rule set, we achieved a significant improvement ($p < 0.01$) of 1.81 points.

In terms of decoding time, even though we used Algorithm 3 described in (Huang and Chiang, 2005), which lazily generated the N-best translation candidates, the decoding time tended to be increased because more rules were available during cube-pruning. Figure 4 shows a comparison of decoding time (seconds per sentence) and the number of rules used for translating the test set. Easy to observe that, decoding time increases in a nearly linear way following the increase of the number of rules used during decoding.

Finally, compared with Joshua, which achieved a BLEU score of 24.79 (%) on the test set with a decoding speed of 8.8 seconds per sentence, our forest-based decoder achieved a significantly better ($p < 0.01$) BLEU score by using either of the four types of translation rules.

5 Related Research

Galley et al. (2006) first used derivation forests of aligned tree-string pairs to express multiple interpretations of unaligned target words. The EM algorithm was used to jointly estimate 1) the translation probabilities and fractional counts of rules and 2) the scores of derivations in the derivation forests. By dealing with the ambiguous word alignment instead of unaligned target words, syntax-based re-alignment models were proposed by (May and Knight, 2007; Wang et al., 2010) for tree-based translations.

Free attachment of the unaligned target word problem was ignored in (Mi and Huang, 2008), which was the first study on extracting tree-to-string rules from aligned forest-string pairs. This inspired the idea to re-align a packed forest and a target sentence. Specially, we observed that most incorrect or ambiguous word alignments are caused by function words rather than content words. Thus, we focus on the realignment of target function words to source tree fragments and use a dependency parser to limit the attachments of unaligned target words.

6 Conclusion

We have proposed an effective use of target function words for extracting generalized transducer rules for forest-based translation. We extend the unaligned word approach described in (Galley et al., 2006) from the 1-best tree to the packed parse forest. A simple yet effective modification is that, during rule extraction, we account for multiple interpretations of both aligned and unaligned target function words. That is, we chose to loose the ambiguous alignments for all of the target function words. The consideration behind is in order to generate target function words in a robust manner. In order to avoid generating too large a derivation forest for a packed forest, we further used chunk-level information yielded by a target dependency parser. Extensive experiments on large-scale English-to-Japanese translation resulted in a significant improvement in BLEU score of 1.8 points ($p < 0.01$), as compared with our implementation of a strong forest-to-string baseline system (Mi et al., 2008; Mi and Huang, 2008).

The present work only re-aligns target function words to source tree fragments. It will be valuable to investigate the feasibility to re-align all the target words to source tree fragments. Also, it is interesting to automatically learn a word set for re-aligning[7]. Given source parse forests and a target word set for re-aligning beforehand, we argue our approach is generic and applicable to any language pairs. Finally, we intend to extend the proposed approach to tree-to-tree translation frameworks by

[7]This idea comes from one reviewer, we express our thankfulness here.

re-aligning subtree pairs (Liu et al., 2009; Chiang, 2010) and consistency-to-dependency frameworks by re-aligning consistency-tree-to-dependency-tree pairs (Mi and Liu, 2010) in order to tackle the rule-sparseness problem.

Acknowledgments

The present study was supported in part by a Grant-in-Aid for Specially Promoted Research (MEXT, Japan), by the Japanese/Chinese Machine Translation Project through Special Coordination Funds for Promoting Science and Technology (MEXT, Japan), and by Microsoft Research Asia Machine Translation Theme.

Wu (wu.xianchao@lab.ntt.co.jp) has moved to NTT Communication Science Laboratories and Tsujii (junichi.tsujii@live.com) has moved to Microsoft Research Asia.

References

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*, pages 263–270, Ann Arbor, MI.

David Chiang. 2010. Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1443–1452, Uppsala, Sweden, July. Association for Computational Linguistics.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39:1–38.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT-NAACL*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of COLING-ACL*, pages 961–968, Sydney.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of IWPT*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL)*, Edomonton, Canada, May 27-June 1.

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

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proceedings of CoNLL-2002*, pages 63–69. Taipei, Taiwan.

Zhifei Li, Chris Callison-Burch, Chris Dyery, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren N. G. Thornton, Jonathan Weese, and Omar F. Zaidan. 2009. Demonstration of joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, pages 25–28, August.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment templates for statistical machine transaltion. In *Proceedings of COLING-ACL*, pages 609–616, Sydney, Australia.

Yang Liu, Yajuan Lü, and Qun Liu. 2009. Improving tree-to-tree translation with packed forests. In *Proceedings of ACL-IJCNLP*, pages 558–566, August.

Samuel E. Martin. 1975. *A Reference Grammar of Japanese*. New Haven, Conn.: Yale University Press.

Jonathan May and Kevin Knight. 2007. Syntactic re-alignment models for machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 360–368, Prague, Czech Republic, June. Association for Computational Linguistics.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of EMNLP*, pages 206–214, October.

Haitao Mi and Qun Liu. 2010. Constituency to dependency translation with forests. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1433–1442, Uppsala, Sweden, July. Association for Computational Linguistics.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL-08:HLT*, pages 192–199, Columbus, Ohio.

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature forest models for probabilistic hpsg parsing. *Computational Lingustics*, 34(1):35–80.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Ivan A. Sag, Thomas Wasow, and Emily M. Bender. 2003. *Syntactic Theory: A Formal Introduction*. Number 152 in CSLI Lecture Notes. CSLI Publications.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *Proceedings of International Conference on Spoken Language Processing*, pages 901–904.

Masao Utiyama and Hitoshi Isahara. 2007. A japanese-english patent parallel corpus. In *Proceedings of MT Summit XI*, pages 475–482, Copenhagen.

Wei Wang, Jonathan May, Kevin Knight, and Daniel Marcu. 2010. Re-structuring, re-labeling, and re-aligning for syntax-based machine translation. *Computational Linguistics*, 36(2):247–277.

Xianchao Wu, Takuya Matsuzaki, and Jun'ichi Tsujii. 2010. Fine-grained tree-to-string translation rule extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 325–334, Uppsala, Sweden, July. Association for Computational Linguistics.

Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. 2009. Using a dependency parser to improve smt for subject-object-verb languages. In *Proceedings of HLT-NAACL*, pages 245–253.

Min Zhang, Hongfei Jiang, Ai Ti Aw, Jun Sun, Sheng Li, and Chew Lim Tan. 2007. A tree-to-tree alignment-based model for statistical machine translation. In *Proceedings of MT Summit XI*, pages 535–542, Copenhagen, Denmark, September.

Combining Morpheme-based Machine Translation with Post-processing Morpheme Prediction

Ann Clifton and **Anoop Sarkar**
Simon Fraser University
Burnaby, British Columbia, Canada
{ann_clifton,anoop}@sfu.ca

Abstract

This paper extends the training and tuning regime for phrase-based statistical machine translation to obtain fluent translations *into* morphologically complex languages (we build an English to Finnish translation system). Our methods use unsupervised morphology induction. Unlike previous work we focus on morphologically productive phrase pairs – our decoder can combine morphemes across phrase boundaries. Morphemes in the target language may not have a corresponding morpheme or word in the source language. Therefore, we propose a novel combination of post-processing morphology prediction with morpheme-based translation. We show, using both automatic evaluation scores and linguistically motivated analyses of the output, that our methods outperform previously proposed ones and provide the best known results on the English-Finnish Europarl translation task. Our methods are mostly language independent, so they should improve translation into other target languages with complex morphology.

1 Translation and Morphology

Languages with rich morphological systems present significant hurdles for statistical machine translation (SMT), most notably data sparsity, source-target asymmetry, and problems with automatic evaluation.

In this work, we propose to address the problem of morphological complexity in an English-to-Finnish MT task within a phrase-based translation framework. We focus on unsupervised segmentation methods to derive the morphological information supplied to the MT model in order to provide coverage on very large datasets and for languages with few hand-annotated resources. In fact, in our experiments, unsupervised morphology always outperforms the use of a hand-built morphological analyzer. Rather than focusing on a few linguistically motivated aspects of Finnish morphological behaviour, we develop techniques for handling morphological complexity in general. We chose Finnish as our target language for this work, because it exemplifies many of the problems morphologically complex languages present for SMT. Among all the languages in the Europarl data-set, Finnish is the most difficult language to translate from and into, as was demonstrated in the MT Summit shared task (Koehn, 2005). Another reason is the current lack of knowledge about how to apply SMT successfully to agglutinative languages like Turkish or Finnish.

Our main contributions are: 1) the introduction of the notion of segmented translation where we explicitly allow phrase pairs that can end with a dangling morpheme, which can connect with other morphemes as part of the translation process, and 2) the use of a fully segmented translation model in combination with a post-processing morpheme prediction system, using unsupervised morphology induction. Both of these approaches beat the state of the art on the English-Finnish translation task. Morphology can express both content and function categories, and our experiments show that it is important to use morphology both within the translation model (for morphology with content) and outside it (for morphology contributing to fluency).

Automatic evaluation measures for MT, BLEU (Papineni et al., 2002), WER (Word Error Rate) and PER (Position Independent Word Error Rate) use the word as the basic unit rather than morphemes. In a word com-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 32–42,
Portland, Oregon, June 19-24, 2011.

prised of multiple morphemes, getting even a single morpheme wrong means the entire word is wrong. In addition to standard MT evaluation measures, we perform a detailed linguistic analysis of the output. Our proposed approaches are significantly better than the state of the art, achieving the highest reported BLEU scores on the English-Finnish Europarl version 3 data-set. Our linguistic analysis shows that our models have fewer morpho-syntactic errors compared to the word-based baseline.

2 Models

2.1 Baseline Models

We set up three baseline models for comparison in this work. The first is a basic word-based model (called Baseline in the results); we trained this on the original unsegmented version of the text. Our second baseline is a factored translation model (Koehn and Hoang, 2007) (called Factored), which used as factors the word, "stem"[1] and suffix. These are derived from the same unsupervised segmentation model used in other experiments. The results (Table 3) show that a factored model was unable to match the scores of a simple word-based baseline. We hypothesize that this may be an inherently difficult representational form for a language with the degree of morphological complexity found in Finnish. Because the morphology generation must be precomputed, for languages with a high degree of morphological complexity, the combinatorial explosion makes it unmanageable to capture the full range of morphological productivity. In addition, because the morphological variants are generated on a per-word basis within a given phrase, it excludes productive morphological combination across phrase boundaries and makes it impossible for the model to take into account any long-distance dependencies between morphemes. We conclude from this result that it may be more useful for an agglutinative language to use morphology beyond the confines of the phrasal unit, and condition its generation on more than just the local target stem. In order to compare the performance of unsupervised segmentation for translation, our third baseline is a segmented translation model based on a supervised segmentation model (called Sup), using the hand-built Omorfi morphological analyzer (Pirinen and Listenmaa, 2007), which provided slightly higher BLEU scores than the word-based baseline.

[1] see Section 2.2.

2.2 Segmented Translation

For segmented translation models, it cannot be taken for granted that greater linguistic accuracy in segmentation yields improved translation (Chang et al., 2008). Rather, the goal in segmentation for translation is instead to maximize the amount of lexical content-carrying morphology, while generalizing over the information not helpful for improving the translation model. We therefore trained several different segmentation models, considering factors of granularity, coverage, and source-target symmetry.

We performed unsupervised segmentation of the target data, using Morfessor (Creutz and Lagus, 2005) and Paramor (Monson, 2008), two top systems from the Morpho Challenge 2008 (their combined output was the Morpho Challenge winner). However, translation models based upon either Paramor alone or the combined systems output could not match the word-based baseline, so we concentrated on Morfessor. Morfessor uses minimum description length criteria to train a HMM-based segmentation model. When tested against a human-annotated gold standard of linguistic morpheme segmentations for Finnish, this algorithm outperforms competing unsupervised methods, achieving an F-score of 67.0% on a 3 million sentence corpus (Creutz and Lagus, 2006). Varying the perplexity threshold in Morfessor does not segment more word types, but rather over-segments the same word types. In order to get robust, common segmentations, we trained the segmenter on the 5000 most frequent words[2]; we then used this to segment the entire data set. In order to improve coverage, we then further segmented

[2] For the factored model baseline we also used the same setting *perplexity* = 30, 5,000 most frequent words, but with all but the last suffix collapsed and called the "stem".

	Training Set	Test Set
Total	64,106,047	21,938
Morph	30,837,615	5,191
Hanging Morph	10,906,406	296

Table 1: Morpheme occurences in the phrase table and in translation.

any word type that contained a match from the most frequent suffix set, looking for the longest matching suffix character string. We call this method Unsup L-match.

After the segmentation, word-internal morpheme boundary markers were inserted into the segmented text to be used to reconstruct the surface forms in the MT output. We then trained the Moses phrase-based system (Koehn et al., 2007) on the segmented and marked text. After decoding, it was a simple matter to join together all adjacent morphemes with word-internal boundary markers to reconstruct the surface forms. Figure 1(a) gives the full model overview for all the variants of the segmented translation model (supervised/unsupervised; with and without the Unsup L-match procedure).

Table 1 shows how morphemes are being used in the MT system. Of the phrases that included segmentations ('Morph' in Table 1), roughly a third were 'productive', i.e. had a hanging morpheme (with a form such as *stem+*) that could be joined to a suffix ('Hanging Morph' in Table 1). However, in phrases used while decoding the development and test data, roughly a quarter of the phrases that generated the translated output included segmentations, but of these, only a small fraction (6%) had a hanging morpheme; and while there are many possible reasons to account for this we were unable to find a single convincing cause.

2.3 Morphology Generation

Morphology generation as a post-processing step allows major vocabulary reduction in the translation model, and allows the use of morphologically targeted features for modeling inflection. A possible disadvantage of this approach is that in this model there is no opportunity to consider the morphology in translation since it is removed prior to training the translation model. Morphology generation models can use a variety of bilingual and contextual information to capture dependencies between morphemes, often more long-distance than what is possible using n-gram language models over morphemes in the segmented model.

Similar to previous work (Minkov et al., 2007; Toutanova et al., 2008), we model morphology generation as a sequence learning problem. Unlike previous work, we use unsupervised morphology induction and use automatically generated suffix classes as tags. The first phase of our morphology prediction model is to train a MT system that produces morphologically simplified word forms in the target language. The output word forms are complex stems (a stem and some suffixes) but still missing some important suffix morphemes. In the second phase, the output of the MT decoder is then tagged with a sequence of abstract suffix tags. In particular, the output of the MT decoder is a sequence of complex stems denoted by $\boldsymbol{x}$ and the output is a sequence of suffix class tags denoted by $\boldsymbol{y}$. We use a list of parts from ($\boldsymbol{x}$,$\boldsymbol{y}$) and map to a d-dimensional feature vector $\Phi(\boldsymbol{x}, \boldsymbol{y})$, with each dimension being a real number. We infer the best sequence of tags using:

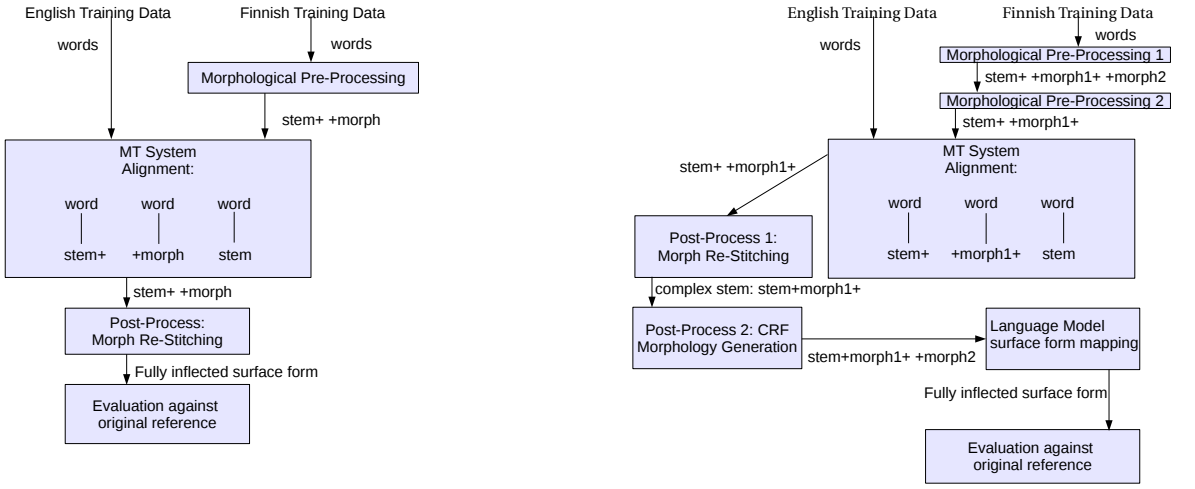
$$F(\boldsymbol{x}) = \underset{\boldsymbol{y}}{\operatorname{argmax}}\, p(\boldsymbol{y} \mid \boldsymbol{x}, \mathbf{w})$$

where $F(\boldsymbol{x})$ returns the highest scoring output $\boldsymbol{y}^*$. A *conditional random field* (CRF) (Lafferty et al., 2001) defines the conditional probability as a linear score for each candidate $\boldsymbol{y}$ and a *global* normalization term:

$$\log p(\boldsymbol{y} \mid \boldsymbol{x}, \mathbf{w}) = \Phi(\boldsymbol{x}, \boldsymbol{y}) \cdot \mathbf{w} - \log Z$$

where $Z = \sum_{\boldsymbol{y'} \in GEN(\boldsymbol{x})} \exp(\Phi(\boldsymbol{x}, \boldsymbol{y'}) \cdot \mathbf{w})$. We use stochastic gradient descent (using *crfsgd*[3]) to train the weight vector $\mathbf{w}$. So far, this is all off-the-shelf sequence learning. However, the output $\boldsymbol{y}^*$ from the CRF decoder is still only a sequence of abstract suffix tags. The third and final phase in our morphology prediction model

[3] *http://leon.bottou.org/projects/sgd*



(a) Segmented Translation Model (b) Post-Processing Model Translation & Generation

Figure 1: Training and testing pipelines for the SMT models.

is to take the abstract suffix tag sequence $\boldsymbol{y}^*$ and then map it into fully inflected word forms, and rank those outputs using a morphemic language model. The abstract suffix tags are extracted from the unsupervised morpheme learning process, and are carefully designed to enable CRF training and decoding. We call this model CRF-LM for short. Figure 1(b) shows the full pipeline and Figure 2 shows a worked example of all the steps involved.

We use the morphologically segmented training data (obtained using the segmented corpus described in Section 2.2[4]) and remove selected suffixes to create a morphologically simplified version of the training data. The MT model is trained on the morphologically simplified training data. The output from the MT system is then used as input to the CRF model. The CRF model was trained on a ∼210,000 Finnish sentences, consisting of ∼1.5 million tokens; the 2,000 sentence Europarl test set consisted of 41,434 stem tokens. The labels in the output sequence $\mathbf{y}$ were obtained by selecting the most productive 150 stems, and then collapsing certain vowels into equivalence classes corresponding to Finnish vowel harmony patterns. Thus variants -kö and -ko become vowel-generic enclitic particle -kO, and variants -ssä and -ssa become the vowel-generic inessive case marker -ssA, etc. This is the only language-specific component of our translation model. However, we expect this approach to work for other agglutinative languages as well. For fusional languages like Spanish, another mapping from suffix to abstract tags might be needed. These suffix transformations to their equivalence classes prevent morphophonemic variants of the same morpheme from competing against each other in the prediction model. This resulted in 44 possible label outputs per stem which was a reasonable sized tag-set for CRF training. The CRF was trained on monolingual features of the segmented text for suffix prediction, where t is the current token:

Word Stem	$s_{t-n}, .., s_t, .., s_{t+n} (n = 4)$
Morph Prediction	y_{t-2}, y_{t-1}, y_t

With this simple feature set, we were able to use features over longer distances, resulting in a total of 1,110,075 model features. After CRF based recovery of the suffix tag sequence, we use a bigram language model trained on a full segmented version on the training data to recover the original vowels. We used bigrams only, because the suffix vowel harmony alternation depends only upon the preceding phonemes in the word from which it was segmented.

[4]Note that unlike Section 2.2 we do not use Unsup L-match because when evaluating the CRF model on the suffix prediction task it obtained 95.61% without using Unsup L-match and 82.99% when using Unsup L-match.

original training data:
koskevaa mietintöä käsitellään
segmentation:
koske+ +va+ +a mietintö+ +ä käsi+ +te+ +llä+ +ä+ +n
(train bigram language model with mapping A = { a, ä })
map final suffix to abstract tag-set:
koske+ +va+ +A mietintö+ +A käsi+ +te+ +llä+ +ä+ +n
(train CRF model to predict the final suffix)
peeling of final suffix:
koske+ +va+ mietintö+ käsi+ +te+ +llä+ +ä+
(train SMT model on this transformation of training data)

(a) Training

decoder output:
koske+ +va+ mietintö+ käsi+ +te+ +llä+ +ä+
decoder output stitched up:
koskeva+ mietintö+ käsitellää+
CRF model prediction:
$\boldsymbol{x}$ = 'koskeva+ mietintö+ käsitellää+', $\boldsymbol{y}$ = '+A +A +n'
koskeva+ +A mietintö+ +A käsitellää+ +n
unstitch morphemes:
koske+ +va+ +A mietintö+ +A käsi+ +te+ +llä+ +ä+ +n
language model disambiguation:
koske+ +va+ +a mietintö+ +ä käsi+ +te+ +llä+ +ä+ +n
final stitching:
koskevaa mietintöä käsitellään
(the output is then compared to the reference translation)

(b) Decoding

Figure 2: Worked example of all steps in the post-processing morphology prediction model.

3 Experimental Results

For all of the models built in this paper, we used the Europarl version 3 corpus (Koehn, 2005) English-Finnish training data set, as well as the standard development and test data sets. Our parallel training data consists of ∼1 million sentences of 40 words or less, while the development and test sets were each 2,000 sentences long. In all the experiments conducted in this paper, we used the Moses[5] phrase-based translation system (Koehn et al., 2007), 2008 version. We trained all of the Moses systems herein using the standard features: language model, reordering model, translation model, and word penalty; in addition to these, the factored experiments called for additional translation and generation features for the added factors as noted above. We used in all experiments the following settings: a hypothesis stack size 100, distortion limit 6, phrase translations limit 20, and maximum phrase length 20. For the language models, we used SRILM 5-gram language models (Stolcke, 2002) for all factors. For our word-based Baseline system, we trained a word-based model using the same Moses system with identical settings. For evaluation against segmented translation systems in segmented forms before word reconstruction, we also segmented the baseline system's word-based output. All the BLEU scores reported are for lowercase evaluation.

We did an initial evaluation of the segmented output translation for each system using the notion of m-BLEU score (Luong et al., 2010) where the BLEU score is computed by comparing the segmented output with a segmented reference translation. Table 2 shows the m-BLEU scores for various systems. We also show the m-BLEU score without unigrams, since over-segmentation could lead to artificially high m-BLEU scores. In fact, if we compare the relative improvement of our m-BLEU scores for the Unsup L-match system we see a relative improvement of 39.75% over the baseline. Luong et. al. (2010) report an m-BLEU score of 55.64% but obtain a relative improvement of 0.6% over their baseline m-BLEU score. We find that when using a good segmentation model, segmentation of the morphologically complex target language improves model performance over an unsegmented baseline (the confidence scores come from bootstrap resampling). Table 3 shows the evaluation scores for all the baselines and the methods introduced in this paper using standard word-based lowercase BLEU, WER and PER. We do

Segmentation	m-BLEU	No Uni
Baseline	14.84±0.69	9.89
Sup	18.41±0.69	13.49
Unsup L-match	**20.74±0.68**	15.89

Table 2: Segmented Model Scores. Sup refers to the supervised segmentation baseline model. m-BLEU indicates that the segmented output was evaluated against a segmented version of the reference (this measure does not have the same correlation with human judgement as BLEU). No Uni indicates the segmented BLEU score without unigrams.

[5] http://www.statmt.org/moses/

Model	BLEU	WER	TER
Baseline	14.68	74.96	72.42
Factored	14.22	76.68	74.15
(Luong et.al, 2010)	14.82	-	-
Sup	14.90	74.56	71.84
Unsup L-match	**15.09***	74.46	71.78
CRF-LM	14.87	**73.71**	**71.15**

Table 3: Test Scores: lowercase BLEU, WER and TER. The $*$ indicates a statistically significant improvement of BLEU score over the Baseline model. The boldface scores are the best performing scores per evaluation measure.

better than (Luong et al., 2010), the previous best score for this task. We also show a better relative improvement over our baseline when compared to (Luong et al., 2010): a relative improvement of 4.86% for Unsup L-match compared to our baseline word-based model, compared to their 1.65% improvement over their baseline word-based model. Our best performing method used unsupervised morphology with L-match (see Section 2.2) and the improvement is significant: bootstrap resampling provides a confidence margin of ± 0.77 and a t-test (Collins et al., 2005) showed significance with $p = 0.001$.

3.1 Morphological Fluency Analysis

To see how well the models were doing at getting morphology right, we examined several patterns of morphological behavior. While we wish to explore minimally supervised morphological MT models, and use as little language specific information as possible, we do want to use linguistic analysis on the output of our system to see how well the models capture essential morphological information in the target language. So, we ran the word-based baseline system, the segmented model (Unsup L-match), and the prediction model (CRF-LM) outputs, along with the reference translation through the supervised morphological analyzer Omorfi (Pirinen and Listenmaa, 2007). Using this analysis, we looked at a variety of linguistic constructions that might reveal patterns in morphological behavior. These were: (a) explicitly marked noun forms, (b) noun-adjective case agreement, (c) subject-verb person/number agreement, (d) transitive object case marking, (e) postpositions, and (f) possession. In each of these categories, we looked for construction matches on a per-sentence level between the models' output and the reference translation.

Table 4 shows the models' performance on the constructions we examined. In all of the categories, the CRF-LM model achieves the best precision score, as we explain below, while the Unsup L-match model most frequently gets the highest recall score.

A general pattern in the most prevalent of these constructions is that the baseline tends to prefer the least marked form for noun cases (corresponding to the nominative) more than the reference or the CRF-LM model. The baseline leaves nouns in the (unmarked) nominative far more than the reference, while the CRF-LM model comes much closer, so it seems to fare better at explicitly marking forms, rather than defaulting to the more frequent unmarked form.

Finnish adjectives must be marked with the same case as their head noun, while verbs must agree in person and number with their subject. We saw that in both these categories, the CRF-LM model outperforms for precision, while the segmented model gets the best recall.

In addition, Finnish generally marks direct objects of verbs with the accusative or the partitive case; we observed more accusative/partitive-marked nouns following verbs in the CRF-LM output than in the baseline, as illustrated by example (1) in Fig. 3. While neither translation picks the same verb as in the reference for the input 'clarify,' the CRF-LM-output paraphrases it by using a grammatical construction of the transitive verb followed by a noun phrase inflected with the accusative case, correctly capturing the transitive construction. The baseline translation instead follows 'give' with a direct object in the nominative case.

To help clarify the constructions in question, we have used Google Translate[6] to provide back-

[6] *http://translate.google.com/*

Construction	Freq.	Baseline			Unsup L-match			CRF-LM		
		P	R	F	P	R	F	P	R	F
Noun Marking	5.5145	51.74	78.48	62.37	53.11	**83.63**	64.96	**54.99**	80.21	**65.25**
Trans Obj	1.0022	32.35	27.50	29.73	33.47	29.64	31.44	**35.83**	**30.71**	**33.07**
Noun-Adj Agr	0.6508	72.75	67.16	69.84	69.62	**71.00**	**70.30**	**73.29**	62.58	67.51
Subj-Verb Agr	0.4250	56.61	40.67	47.33	55.90	**48.17**	**51.48**	**57.79**	40.17	47.40
Postpositions	0.1138	43.31	29.89	35.37	39.31	**36.96**	**38.10**	**47.16**	31.52	37.79
Possession	0.0287	66.67	**70.00**	68.29	75.68	**70.00**	**72.73**	**78.79**	60.00	68.12

Table 4: Model Accuracy: Morphological Constructions. Freq. refers to the construction's average number of occurrences per sentence, also averaged over the various translations. P, R and F stand for precision, recall and F-score. The constructions are listed in descending order of their frequency in the texts. The highlighted value in each column is the most accurate with respect to the reference value.

translations of our MT output into English; to contextualize these back-translations, we have provided Google's back-translation of the reference.

The use of postpositions shows another difference between the models. Finnish postpositions require the preceding noun to be in the genitive or sometimes partitive case, which occurs correctly more frequently in the CRF-LM than the baseline. In example (2) in Fig. 3, all three translations correspond to the English text, 'with the basque nationalists.' However, the CRF-LM output is more grammatical than the baseline, because not only do the adjective and noun agree for case, but the noun 'baskien' to which the postposition 'kanssa' belongs is marked with the correct genitive case. However, this well-formedness is not rewarded by BLEU, because 'baskien' does not match the reference.

In addition, while Finnish may express possession using case marking alone, it has another construction for possession; this can disambiguate an otherwise ambiguous clause. This alternate construction uses a pronoun in the genitive case followed by a possessive-marked noun; we see that the CRF-LM model correctly marks this construction more frequently than the baseline. As example (3) in Fig. 3 shows, while neither model correctly translates 'matkan' ('trip'), the baseline's output attributes the inessive 'yhteydess' ('connection') as belonging to 'tulokset' ('results'), and misses marking the possession linking it to 'Commissioner Fischler'.

Our manual evaluation shows that the CRF-LM model is producing output translations that are more morphologically fluent than the word-based baseline and the segmented translation Unsup L-match system, even though the word choices lead to a lower BLEU score overall when compared to Unsup L-match.

4 Related Work

The work on morphology in MT can be grouped into three categories, factored models, segmented translation, and morphology generation.

Factored models (Koehn and Hoang, 2007) factor the phrase translation probabilities over additional information annotated to each word, allowing for text to be represented on multiple levels of analysis. We discussed the drawbacks of factored models for our task in Section 2.1. While (Koehn and Hoang, 2007; Yang and Kirchhoff, 2006; Avramidis and Koehn, 2008) obtain improvements using factored models for translation into English, German, Spanish, and Czech, these models may be less useful for capturing long-distance dependencies in languages with much more complex morphological systems such as Finnish. In our experiments factored models did worse than the baseline.

Segmented translation performs morphological analysis on the morphologically complex text for use in the translation model (Brown et al., 1993; Goldwater and McClosky, 2005; de Gispert and Mariño, 2008). This method unpacks complex forms into simpler, more frequently occurring components, and may also increase the symmetry of the lexically realized content be-

(1) Input: 'the charter we are to approve today both strengthens and gives visible shape to the common fundamental rights and values our community is to be based upon.'

   a. Reference: perusoikeuskirja , jonka tänään aiomme hyväksyä , sekä vahvistaa että **selventää** (selventää/VERB/ACT/INF/SG/LAT-clarify) **niitä** (ne/PRONOUN/PL/PAR-them) yhteisiä perusoikeuksia ja -arvoja , joiden on oltava yhteisömme perusta.
   Back-translation: 'Charter of Fundamental Rights, which today we are going to accept that clarify and strengthen the common fundamental rights and values, which must be community based.'
   b. Baseline: perusoikeuskirja me hyväksymme tänään molemmat vahvistaa ja **antaa** (antaa/VERB/INF/SG/LAT-give) **näkyvä** (näkyä/VERB/ACT/PCP/SG/NOM-visible) muokata yhteistä perusoikeuksia ja arvoja on perustuttava.
   Back-translation: 'Charter today, we accept both confirm and modify to make a visible and common values, fundamental rights must be based.'
   c. CRF-LM: perusoikeuskirja on hyväksytty tänään , sekä vahvistaa ja **antaa** (antaa/VERB/ACT/INF/SG/LAT-give) **konkreettisen** (konkreettinen/ADJECTIVE/SG/GEN,ACC-concrete) **muodon** (muoto/NOUN/SG/GEN,ACC-shape) yhteisiä perusoikeuksia ja perusarvoja , yhteisön on perustuttava.
   Back-translation: 'Charter has been approved today, and to strengthen and give concrete shape to the common basic rights and fundamental values, the Community must be based.'

(2) Input: 'with the basque nationalists'

   a. Reference: baskimaan kansallismielisten kanssa
   basque-SG/NOM+land-SG/GEN,ACC nationalists-PL/GEN with-POST
   b. Baseline: baskimaan kansallismieliset kanssa
   basque-SG/NOM-+land-SG/GEN,ACC kansallismielinen-PL/NOM,ACC-nationalists POST-with
   c. CRF-LM: kansallismielisten baskien kanssa
   nationalists-PL/GEN basques-PL/GEN with-POST

(3) Input: 'and in this respect we should value the latest measures from commissioner fischler , the results of **his trip to morocco on the 26th of last month** and the high level meetings that took place, including the one with the king himself'

   a. Reference: ja tässä mielessä osaamme myös arvostaa komission jäsen fischlerin viimeisimpiä toimia , jotka ovat **hänen** (hänen/GEN-his) **marokkoon 26 lokakuuta tekemns** (tekemänsä/POSS-his) **matkan** (matkan/GEN-tour) ja korkean tason kokousten jopa itsensä kuninkaan kanssa tulosta
   Back-translation: 'and in this sense we can also appreciate the Commissioner Fischler's latest actions, which are **his to Morocco 26 October trip** to high-level meetings and even the king himself with the result
   b. Baseline: ja tässä yhteydessä olisi arvoa viimeisin toimia komission jäsen fischler , tulokset monitulkintaisia **marokon yhteydessä** (yhteydess/INE-connection) , ja viime kuussa pidettiin korkean tason kokouksissa , mukaan luettuna kuninkaan kanssa
   Back-translation: 'and in this context would be the value of the last act, Commissioner Fischler, the results of **the Moroccan context**, ambiguous, and last month held high level meetings, including with the king'
   c. CRF-LM: ja tässä yhteydessä meidän olisi lisäarvoa viimeistä toimenpiteitä kuin komission jäsen fischler , että **hänen** (hänen/GEN-his) **kokemuksensa** (kokemuksensa/POSS-experience) **marokolle** (marokolle-Moroccan) viime kuun 26 ja korkean tason tapaamiset järjestettiin, kuninkaan kanssa
   Back-translation: 'and in this context, we should value the last measures as the Commissioner Fischler, that **his experience in Morocco** has on the 26th and high-level meetings took place, including with the king.'

Figure 3: Morphological fluency analysis (see Section 3.1).

tween source and target. In a somewhat orthogonal approach to ours, (Ma et al., 2007) use alignment of a parallel text to pack together adjacent segments in the alignment output, which are then fed back to the word aligner to bootstrap an improved alignment, which is then used in the translation model. We compared our results against (Luong et al., 2010) in Table 3 since their results are directly comparable to ours. They use a segmented phrase table and language model along with the word-based versions in the decoder and in tuning a Finnish target. Their approach requires segmented phrases to match word boundaries, eliminating morphologically productive phrases. In their work a segmented language model can score a translation, but cannot insert morphology that does not show source-side reflexes. In order to perform a similar experiment that still allowed for morphologically productive phrases, we tried training a segmented translation model, the output of which we stitched up in tuning so as to tune to a word-based reference. The goal of this experiment was to control the segmented model's tendency to overfit by rewarding it for using correct whole-word forms. However, we found

that this approach was less successful than using the segmented reference in tuning, and could not meet the baseline (13.97% BLEU best tuning score, versus 14.93% BLEU for the baseline best tuning score). Previous work in segmented translation has often used linguistically motivated morphological analysis selectively applied based on a language-specific heuristic. A typical approach is to select a highly inflecting class of words and segment them for particular morphology (de Gispert and Mariño, 2008; Ramanathan et al., 2009). Popoviç and Ney (2004) perform segmentation to reduce morphological complexity of the source to translate into an isolating target, reducing the translation error rate for the English target. For Czech-to-English, Goldwater and McClosky (2005) lemmatized the source text and inserted a set of 'pseudowords' expected to have lexical reflexes in English.

Minkov et. al. (2007) and Toutanova et. al. (2008) use a Maximum Entropy Markov Model for morphology generation. The main drawback to this approach is that it removes morphological information from the translation model (which only uses stems); this can be a problem for languages in which morphology expresses lexical content. de Gispert (2008) uses a language-specific targeted morphological classifier for Spanish verbs to avoid this issue. Talbot and Osborne (2006) use clustering to group morphological variants of words for word alignments and for smoothing phrase translation tables. Habash (2007) provides various methods to incorporate morphological variants of words in the phrase table in order to help recognize out of vocabulary words in the source language.

5 Conclusion and Future Work

We found that using a segmented translation model based on unsupervised morphology induction and a model that combined morpheme segments in the translation model with a post-processing morphology prediction model gave us better BLEU scores than a word-based baseline. Using our proposed approach we obtain better scores than the state of the art on the English-Finnish translation task (Luong et al., 2010): from 14.82% BLEU to 15.09%, while using a simpler model. We show that using morphological segmentation in the translation model can improve output translation scores. We also demonstrate that for Finnish (and possibly other agglutinative languages), phrase-based MT benefits from allowing the translation model access to morphological segmentation yielding productive morphological phrases. Taking advantage of linguistic analysis of the output we show that using a post-processing morphology generation model can improve translation fluency on a sub-word level, in a manner that is not captured by the BLEU word-based evaluation measure.

In order to help with replication of the results in this paper, we have run the various morphological analysis steps and created the necessary training, tuning and test data files needed in order to train, tune and test any phrase-based machine translation system with our data. The files can be downloaded from *natlang.cs.sfu.ca*.

In future work we hope to explore the utility of phrases with productive morpheme boundaries and explore why they are not used more pervasively in the decoder. Evaluation measures for morphologically complex languages and tuning to those measures are also important future work directions. Also, we would like to explore a non-pipelined approach to morphological pre- and post-processing so that a globally trained model could be used to remove the target side morphemes that would improve the translation model and then predict those morphemes in the target language.

Acknowledgements

This research was partially supported by NSERC, Canada (RGPIN: 264905) and a Google Faculty Award. We would like to thank Christian Monson, Franz Och, Fred Popowich, Howard Johnson, Majid Razmara, Baskaran Sankaran and the anonymous reviewers for their valuable comments on this work. We would particularly like to thank the developers of the open-source Moses machine translation toolkit and the Omorfi morphological analyzer for Finnish which we used for our experiments.

References

Eleftherios Avramidis and Philipp Koehn. 2008. Enriching morphologically poor languages for statistical machine translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, page 763?770, Columbus, Ohio, USA. Association for Computational Linguistics.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Pi-Chuan Chang, Michel Galley, and Christopher D. Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 224–232, Columbus, Ohio, June. Association for Computational Linguistics.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of 43rd Annual Meeting of the Association for Computational Linguistics (ACL05)*. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2005. Inducing the morphological lexicon of a natural language from unannotated text. In *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR'05)*, pages 106–113, Espoo, Finland.

Mathias Creutz and Krista Lagus. 2006. Morfessor in the morpho challenge. In *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*.

Adriá de Gispert and José Mariño. 2008. On the impact of morphology in English to Spanish statistical MT. *Speech Communication*, 50(11-12).

Sharon Goldwater and David McClosky. 2005. Improving statistical MT through morphological analysis. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 676–683, Vancouver, B.C., Canada. Association for Computational Linguistics.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 868–876, Prague, Czech Republic. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL '07: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–108, Prague, Czech Republic. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X*, pages 79–86, Phuket, Thailand. Association for Computational Linguistics.

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

Minh-Thang Luong, Preslav Nakov, and Min-Yen Kan. 2010. A hybrid morpheme-word representation for machine translation of morphologically rich languages. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 148–157, Cambridge, Massachusetts. Association for Computational Linguistics.

Yanjun Ma, Nicolas Stroppa, and Andy Way. 2007. Bootstrapping word alignment via word packing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 304–311, Prague, Czech Republic. Association for Computational Linguistics.

Einat Minkov, Kristina Toutanova, and Hisami Suzuki. 2007. Generating complex morphology for machine translation. In *In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL07)*, pages 128–135, Prague, Czech Republic. Association for Computational Linguistics.

Christian Monson. 2008. Paramor and morpho challenge 2008. In *Lecture Notes in Computer Science: Workshop of the Cross-Language Evaluation Forum (CLEF 2008), Revised Selected Papers*.

Habash Nizar. 2007. Four techniques for online handling of out-of-vocabulary words in arabic-english statistical machine translation. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*, Columbus, Ohio. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics ACL*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Tommi Pirinen and Inari Listenmaa. 2007. Omorfi morphological analzer. http://gna.org/projects/omorfi.

Maja Popoviç and Hermann Ney. 2004. Towards the use of word stems and suffixes for statistical machine translation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*, pages 1585–1588, Lisbon, Portugal. European Language Resources Association (ELRA).

Ananthakrishnan Ramanathan, Hansraj Choudhary, Avishek Ghosh, and Pushpak Bhattacharyya. 2009. Case markers and morphology: Addressing the crux of the fluency problem in English-Hindi SMT. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 800–808, Suntec, Singapore. Association for Computational Linguistics.

Andreas Stolcke. 2002. Srilm – an extensible language modeling toolkit. *7th International Conference on Spoken Language Processing*, 3:901–904.

David Talbot and Miles Osborne. 2006. Modelling lexical redundancy for machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 969–976, Sydney, Australia, July. Association for Computational Linguistics.

Kristina Toutanova, Hisami Suzuki, and Achim Ruopp. 2008. Applying morphology generation models to machine translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 514–522, Columbus, Ohio, USA. Association for Computational Linguistics.

Mei Yang and Katrin Kirchhoff. 2006. Phrase-based backoff models for machine translation of highly inflected languages. In *Proceedings of the European Chapter of the Association for Computational Linguistics*, pages 41–48, Trento, Italy. Association for Computational Linguistics.

Evaluating the Impact of Coder Errors on Active Learning

Ines Rehbein
Computational Linguistics
Saarland University
rehbein@coli.uni-sb.de

Josef Ruppenhofer
Computational Linguistics
Saarland University
josefr@coli.uni-sb.de

Abstract

Active Learning (AL) has been proposed as a technique to reduce the amount of annotated data needed in the context of supervised classification. While various simulation studies for a number of NLP tasks have shown that AL works well on goldstandard data, there is some doubt whether the approach can be successful when applied to noisy, real-world data sets. This paper presents a thorough evaluation of the impact of annotation noise on AL and shows that systematic noise resulting from biased coder decisions can seriously harm the AL process. We present a method to filter out inconsistent annotations during AL and show that this makes AL far more robust when applied to noisy data.

1 Introduction

Supervised machine learning techniques are still the mainstay for many NLP tasks. There is, however, a well-known bottleneck for these approaches: the amount of high-quality data needed for training, mostly obtained by human annotation. Active Learning (AL) has been proposed as a promising approach to reduce the amount of time and cost for human annotation. The idea behind active learning is quite intuitive: instead of annotating a large number of randomly picked instances we carefully select a small number of instances that are maximally informative for the machine learning classifier. Thus a smaller set of data points is able to boost classifier performance and to yield an accuracy comparable to the one obtained when training the same system on a larger set of randomly chosen data.

Active learning has been applied to several NLP tasks like part-of-speech tagging (Ringger et al., 2007), chunking (Ngai and Yarowsky, 2000), syntactic parsing (Osborne and Baldridge, 2004; Hwa, 2004), Named Entity Recognition (Shen et al., 2004; Laws and Schütze, 2008; Tomanek and Hahn, 2009), Word Sense Disambiguation (Chen et al., 2006; Zhu and Hovy, 2007; Chan and Ng, 2007), text classification (Tong and Koller, 1998) or statistical machine translation (Haffari and Sarkar, 2009), and has been shown to reduce the amount of annotated data needed to achieve a certain classifier performance, sometimes by as much as half. Most of these studies, however, have only simulated the active learning process using goldstandard data. This setting is crucially different from a real world scenario where we have to deal with erroneous data and inconsistent annotation decisions made by the human annotators. While simulations are an indispensable instrument to test different parameters and settings, it has been shown that when applying AL to highly ambiguous tasks like e.g. Word Sense Disambiguation (WSD) with fine-grained sense distinctions, AL can actually harm the learning process (Dang, 2004; Rehbein et al., 2010). Dang suggests that the lack of a positive effect of AL might be due to inconsistencies in the human annotations and that AL cannot efficiently be applied to tasks which need double blind annotation with adjudication to insure a sufficient data quality. Even if we take a more optimistic view and assume that AL might still be useful even for tasks featuring a high degree of ambiguity, it remains crucial to address the problem of annotation noise and its impact on AL.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 43–51,
Portland, Oregon, June 19-24, 2011.

In this paper we present a thorough evaluation of the impact of annotation noise on AL. We simulate different types of coder errors and assess the effect on the learning process. We propose a method to detect inconsistencies and remove them from the training data, and show that our method does alleviate the problem of annotation noise in our experiments.

The paper is structured as follows. Section 2 reports on recent research on the impact of annotation noise in the context of supervised classification. Section 3 describes the experimental setup of our simulation study and presents results. In Section 4 we present our filtering approach and show its impact on AL performance. Section 5 concludes and outlines future work.

2 Related Work

We are interested in the question whether or not AL can be successfully applied to a supervised classification task where we have to deal with a considerable amount of inconsistencies and noise in the data, which is the case for many NLP tasks (e.g. sentiment analysis, the detection of metaphors, WSD with fine-grained word senses, to name but a few). Therefore we do not consider part-of-speech tagging or syntactic parsing, where coders are expected to agree on most annotation decisions. Instead, we focus on work on AL for WSD, where inter-coder agreement (at least for fine-grained annotation schemes) usually is much lower than for the former tasks.

2.1 Annotation Noise

Studies on active learning for WSD have been limited to running simulations of AL using gold standard data and a coarse-grained annotation scheme (Chen et al., 2006; Chan and Ng, 2007; Zhu and Hovy, 2007). Two exceptions are Dang (2004) and Rehbein et al. (2010) who both were not able to replicate the positive findings obtained for AL for WSD on coarse-grained sense distinctions. A possible reason for this failure is the amount of annotation noise in the training data which might mislead the classifier during the AL process. Recent work on the impact of annotation noise on a machine learning task (Reidsma and Carletta, 2008) has shown that random noise can be tolerated in supervised learning, while systematic errors (as caused by biased annotators) can seriously impair the performance of a supervised classifier even if the observed accuracy of the classifier on a test set coming from the same population as the training data is as high as 0.8.

Related work (Beigman Klebanov et al., 2008; Beigman Klebanov and Beigman, 2009) has been studying annotation noise in a multi-annotator setting, distinguishing between *hard* cases (unreliably annotated due to genuine ambiguity) and *easy* cases (reliably annotated data). The authors argue that even for those data points where the annotators agreed on one particular class, a proportion of the agreement might be merely due to chance. Following this assumption, the authors propose a measure to estimate the amount of annotation noise in the data after removing all hard cases. Klebanov et al. (2008; 2009) show that, according to their model, high inter-annotator agreement (κ) achieved in an annotation scenario with two annotators is no guarantee for a high-quality data set. Their model, however, assumes that a) all instances where annotators disagreed are in fact hard cases, and b) that for the hard cases the annotators decisions are obtained by coin-flips. In our experience, some amount of disagreement can also be observed for easy cases, caused by attention slips or by a deviant interpretation of some class(es) by one of the annotators, and the annotation decision of an individual annotator cannot so much be described as random choice (coin-flip) but as systematically biased selection, causing the types of errors which have been shown to be problematic for supervised classification (Reidsma and Carletta, 2008).

Further problems arise in the AL scenario where the instances to be annotated are selected as a function of the sampling method and the annotation judgements made before. Therefore, Beigman and Klebanov Beigman (2009)'s approach of identifying unreliably annotated instances by disagreement is not applicable to AL, as most instances are annotated only once.

2.2 Annotation Noise and Active Learning

For AL to be succesful, we need to remove systematic noise in the training data. The challenge we face is that we only have a small set of seed data and no information about the reliability of the annotations

assigned by the human coders.

Zhu et al. (2008) present a method for detecting outliers in the pool of unannotated data to prevent these instances from becoming part of the training data. This approach is different from ours, where we focus on detecting annotation noise in the manually labelled training data produced by the human coders.

Schein and Ungar (2007) provide a systematic investigation of 8 different sampling methods for AL and their ability to handle different types of noise in the data. The types of noise investigated are a) prediction residual error (the portion of squared error that is independent of training set size), and b) different levels of confusion among the categories. Type a) models the presence of unknown features that influence the true probabilities of an outcome: a form of noise that will increase residual error. Type b) models categories in the data set which are intrinsically hard to disambiguate, while others are not. Therefore, type b) errors are of greater interest to us, as it is safe to assume that intrinsically ambiguous categories will lead to biased coder decisions and result in the systematic annotation noise we are interested in.

Schein and Ungar observe that none of the 8 sampling methods investigated in their experiment achieved a significant improvement over the random sampling baseline on type b) errors. In fact, entropy sampling and margin sampling even showed a decrease in performance compared to random sampling. For AL to work well on noisy data, we need to identify and remove this type of annotation noise during the AL process. To the best of our knowledge, there is no work on detecting and removing annotation noise by human coders during AL.

3 Experimental Setup

To make sure that the data we use in our simulation is as close to real-world data as possible, we do not create an artificial data set as done in (Schein and Ungar, 2007; Reidsma and Carletta, 2008) but use real data from a WSD task for the German verb *drohen* (threaten).[1] *Drohen* has three different word senses which can be disambiguated by humans with a high accuracy.[2] This point is crucial to our setup. To control the amount of noise in the data, we need to be sure that the initial data set is noise-free.

For classification we use a maximum entropy classifier.[3] Our sampling method is uncertainty sampling (Lewis and Gale, 1994), a standard sampling heuristic for AL where new instances are selected based on the confidence of the classifier for predicting the appropriate label. As a measure of uncertainty we use Shannon entropy (1) (Zhang and Chen, 2002) and the *margin* metric (2) (Schein and Ungar, 2007). The first measure considers the model's predictions q for each class c and selects those instances from the pool where the Shannon entropy is highest.

$$-\sum_{c} q_c \, log \, q_c \tag{1}$$

The second measure looks at the difference between the largest two values in the prediciton vector q, namely the two predicted classes c, c' which are, according to our model, the most likely ones for instance x_n, and selects those instances where the difference (*margin*) between the two predicted probabilities is the smallest. We discuss some details of this metric in Section 4.

$$M_n = |P(c|x_n) - P(c'|x_n)| \tag{2}$$

The features we use for WSD are a combination of context features (word token with window size 11 and POS context with window size 7), syntactic features based on the output of a dependency parser[4] and semantic features based on GermaNet hyperonyms. These settings were tuned to the target verb by (Rehbein et al., 2009). All results reported below are averages over a 5-fold cross validation.

3.1 Simulating Coder Errors in AL

Before starting the AL trials we automatically separate the 2,500 sentences into test set (498 sentences) and pool (2,002 sentences),[5] retaining the overall distribution of word senses in the data set. We insert a varying amount of noise into the pool data,

[1] The data has been provided by the SALSA project: `http://www.coli.uni-saarland.de/projects/salsa`

[2] In a pilot study where two human coders assigned labels to a set of 100 sentences, the coders agreed on 99% of the data.

[3] `http://maxent.sourceforge.net`

[4] The MaltParser: `http://maltparser.org`

[5] The split has been made automatically, the unusual numbers are caused by rounding errors.

	test	pool		
% errors	**0%**	**0%**	**ALrand 30%**	**ALbias 30%**
drohen1-salsa	126	506	524	514
Comittment	129	520	522	327
Run_risk	243	976	956	1161
Total	498	2002	2002	2002

Table 1: Distribution of word senses in pool and test sets

starting from 0% up to 30% of noise, increasing by 2% in each trial.

We assess the impact of annotation noise on active learning in three different settings. In the first setting, we randomly select new instances from the pool (random sampling; *rand*). In the second setting, we randomly replace n percent of all labels (from 0 to 30) in the pool by another label before starting the active learning trial, but retain the distribution of the different labels in the pool data (active learning with random errors); (Table 1, *ALrand*, 30%). In the third setting we simulate biased decisions by a human annotator. For a certain fraction (0 to 30%) of instances of a particular non-majority class, we substitute the majority class label for the gold label, thereby producing a more skewed distribution than in the original pool (active learning with biased errors); (Table 1, *ALbias*, 30%).

For all three settings (*rand, ALrand, ALbias*) and each degree of noise (0-30%), we run active learning simulations on the already annotated data, simulating the annotation process by selecting one new, prelabelled instance per trial from the pool and, instead of handing them over to a human coder, assigning the known (possibly erroneous) label to the instance and adding it to the training set. We use the same split (test, pool) for all three settings and all degrees of noise, with identical test sets for all trials.

3.2 Results

Figure 1 shows active learning curves for the different settings and varying degrees of noise. The horizontal black line slightly below 0.5 accuracy shows the majority baseline (the performance obtained when always assigning the majority class). For all degrees of randomly inserted noise, active learning (*ALrand*) outperforms random sampling (*rand*) at an early stage in the learning process. Looking at the biased errors (*ALbias*), we see a different picture. With a low degree of noise, the curves for *ALrand* and *ALbias* are very similar. When inserting more noise, performance for *ALbias* decreases, and with around 20% of biased errors in the pool AL performs worse than our random sampling baseline. In the random noise setting (*ALrand*), even after inserting 30% of errors AL clearly outperforms random sampling. Increasing the size of the seed data reduces the effect slightly, but does not prevent it (not shown here due to space limitations). This confirms the findings that under certain circumstances AL performs worse than random sampling (Dang, 2004; Schein and Ungar, 2007; Rehbein et al., 2010). We could also confirm Schein and Ungar (2007)'s observation that margin sampling is less sensitive to certain types of noise than entropy sampling (Table 2). Because of space limitations we only show curves for margin sampling. For entropy sampling, the general trend is the same, with results being slightly lower than for margin sampling.

4 Detecting Annotation Noise

Uncertainty sampling using the margin metric selects instances for which the difference between classifier predictions for the two most probable classes c, c' is very small (Section 3, Equation 2). When selecting unlabelled instances from the pool, this metric picks examples which represent regions of uncertainty between classes which have yet to be learned by the classifier and thus will advance the learning process. Our human coder, however, is not the perfect oracle assumed in most AL simulations, and might also assign incorrect labels. The filter approach has two objectives: a) to detect incorrect labels assigned by human coders, and b) to prevent the *hard* cases (following the terminology of Klebanov et al. (2008)) from becoming part of the training data.

We proceed as follows. Our approach makes use of the limited set of seed data S and uses heuristics to detect unreliably annotated instances. We assume that the instances in S have been validated thoroughly. We train an ensemble of classifiers E on subsets of S, and use E to decide whether or not a newly annotated instance should be added to the seed.

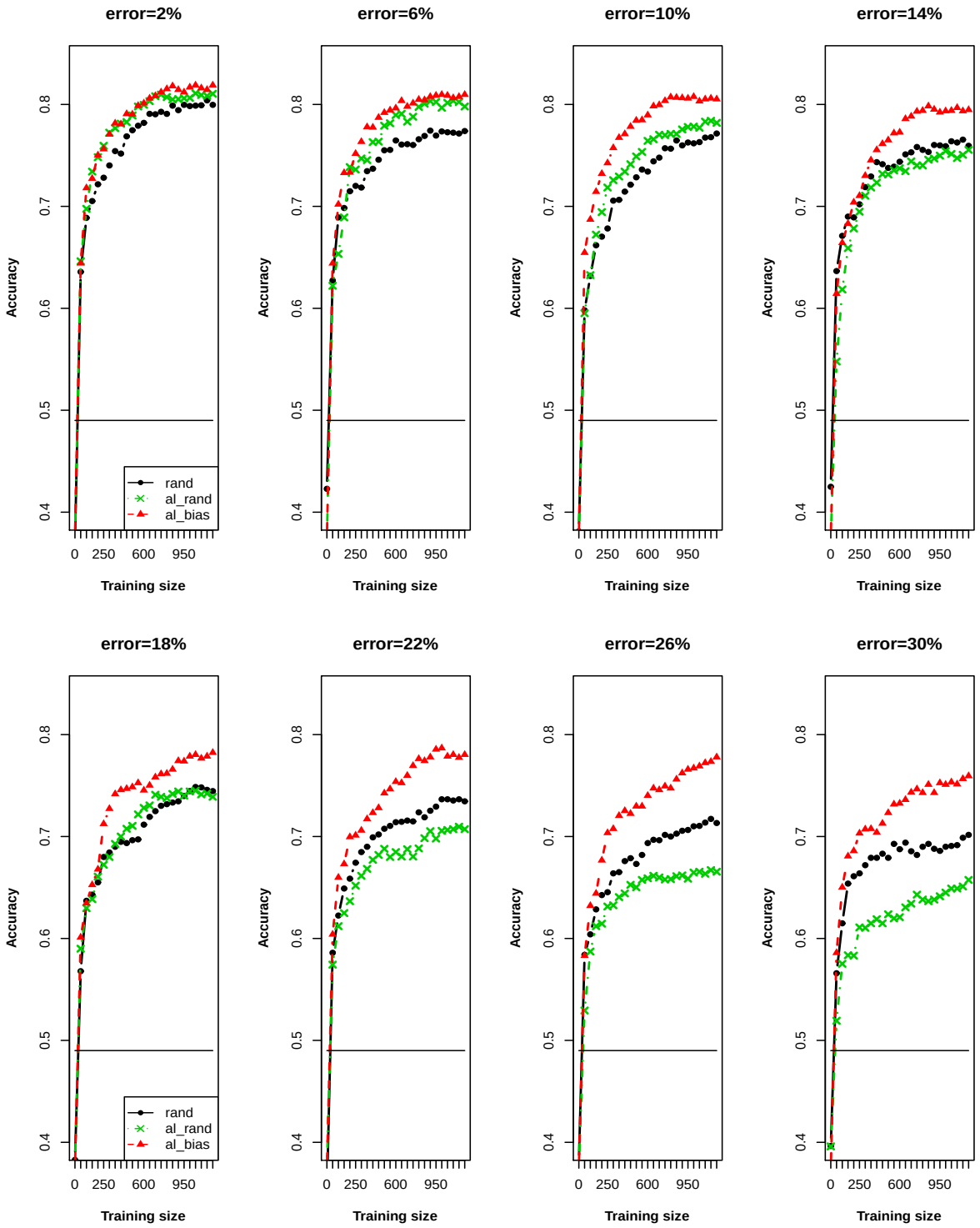


Figure 1: Active learning curves for varying degrees of noise, starting from 0% up to 30% for a training size up to 1200 instances (solid circle (black): random sampling; filled triangle point-up (red): AL with random errors; cross (green): AL with biased errors)

	filter	% error	0	4	8	12	16	20	24	28	30
	-	**rand**	0.763	0.752	0.736	0.741	0.726	0.708	0.707	0.677	0.678
entropy	-	**ALrand**	0.806	0.786	0.779	0.743	0.752	0.762	0.731	0.724	0.729
entropy	y	**ALrand**	0.792	0.786	0.777	0.760	0.771	0.748	0.730	0.729	0.727
margin	-	**ALrand**	0.795	0.795	0.782	0.771	0.758	0.755	0.737	0.719	0.708
margin	y	**ALrand**	0.800	0.785	0.773	0.777	0.765	0.766	0.734	0.735	0.718
entropy	-	**ALbias**	0.806	0.793	0.759	0.748	0.702	0.651	0.625	0.630	0.622
entropy	y	**ALbias**	0.802	0.781	0.777	0.735	0.702	0.678	0.687	0.624	0.616
margin	-	**ALbias**	0.795	0.789	0.770	0.753	0.706	0.684	0.656	0.634	0.624
margin	y	**ALbias**	0.787	0.781	0.787	0.768	0.739	0.700	0.671	0.653	0.651

Table 2: Accuracy for the different sampling methods without and with filtering after adding 500 instances to the seed data

There are a number of problems with this approach. First, there is the risk of overfitting S. Second, we know that classifier accuracy in the early phase of AL is low. Therefore, using classifier predictions at this stage to accept or reject new instances could result in poor choices that might harm the learning proceess. To avoid this and to generalise over S to prevent overfitting, we do not directly train our ensemble on instances from S. Instead, we create new feature vectors F_{gen} on the basis of the feature vectors F_{seed} in S. For each class in S, we extract all attribute-value pairs from the feature vectors for this particular class. For each class, we randomly select features (with replacement) from F_{seed} and combine them into a new feature vector F_{gen}, retaining the distribution of the different classes in the data. As a result, we obtain a more general set of feature vectors F_{gen} with characteristic features being distributed more evenly over the different feature vectors.

In the next step we train $n = 5$ maximum entropy classifiers on subsets of F_{gen}, excluding the instances last annotated by the oracle. Each subset is half the size of the current S. We use the ensemble to predict the labels for the new instances and, based on the predictions, accept or reject these, following the two heuristics below (also see Figure 2).

1. If all n ensemble classifiers agree on one label but disagree with the oracle $\Rightarrow$ reject.

2. If the sum of the margins predicted by the ensemble classifiers is below a particular theshold $t_{margin} \Rightarrow$ reject.

The threshold t_{margin} was set to 0.01, based on a qualitative data analysis.

AL with Filtering:

Input: annotated seed data S,
unannotated pool P
AL loop:

- train classifier C on S
- let C predict labels for data in P
- select new instances from P according to sampling method, hand over to oracle for annotation

Repeat: after every c new instances annotated by the oracle

- for each class in S, extract sets of features F_{seed}
- create new, more general feature vectors F_{gen} from this set (with replacement)
- train an ensemble E of n classifiers on different subsets of F_{gen}

  Filtering Heuristics:

  - **if** all n classifier in E agree on label but disagree with oracle:
    $\Rightarrow$ *remove instance from seed*
  - **if** margin is less than threshold t_{margin}:
    $\Rightarrow$ *remove instance from seed*

Until done

Figure 2: Heuristics for filtering unreliable data points (parameters used: initial seed size: 9 sentences, $c = 10$, $n = 5$, $t_{margin} = 0.01$)

In each iteration of the AL process, one new instance is selected using margin sampling. The instance is presented to the oracle who assigns a label. Then the instance is added to the seed data, thus influencing the selection of the next data point to be annotated. After 10 new instances have been added, we apply the filter technique which finally decides whether the newly added instances will remain in the seed data or will be removed.

Figure 3 shows learning curves for the filter approach. With increasing amount of errors in the pool, a clear pattern emerges. For both sampling methods (ALrand, ALbias), the filtering step clearly improves results. Even for the noisier data sets with up to 26% of errors, ALbias with filtering performs at least as well as random sampling.

4.1 Error Analysis

Next we want to find out what kind of errors the system could detect. We want to know whether the approach is able to detect the errors previously inserted into the data, and whether it manages to identify hard cases representing true ambiguities.

To answer these questions we look at one fold of the ALbias data with 10% of noise. In 1,200 AL iterations the system rejected 116 instances (Table 3). The major part of the rejections was due to the majority vote of the ensemble classifiers (first heuristic, H1) which rejects all instances where the ensemble classifiers agree with each other but disagree with the human judgement. Out of the 105 instances rejected by H1, 41 were labelled incorrectly. This means that we were able to detect around half of the incorrect labels inserted in the pool.

11 instances were filtered out by the margin threshold (H2). None of these contained an incorrect label. On first glance H2 seems to be more lenient than H1, considering the number of rejected sentences. This, however, could also be an effect of the order in which we apply the filters.

errors inserted in pool	173
err. instances selected by AL	93
instances rejected by H1+H2	116
instances rejected by H1	105
true errors rejected by H1	41
instances rejected by H2	11
true errors rejected by H2	0

Table 3: Error analysis of the instances rejected by the filtering approach

The different word senses are evenly distributed over the rejected instances (H1: Commitment 30, drohen1-salsa 38, Run_risk 36; H2: Commitment 3, drohen1-salsa 4, Run_risk 4). This shows that there is less uncertainty about the majority word sense, Run_risk.

It is hard to decide whether the correctly labelled instances rejected by the filtering method would have helped or hurt the learning process. Simply adding them to the seed data after the conclusion of AL would not answer this question, as it would merely tell us whether they improve classification accuracy further, but we still would not know what impact these instances would have had on the selection of instances during the AL process.

5 Conclusions

This paper shows that certain types of annotation noise cause serious problems for active learning approaches. We showed how biased coder decisions can result in an accuracy for AL approaches which is below the one for random sampling. In this case, it is necessary to apply an additional filtering step to remove the noisy data from the training set. We presented an approach based on a resampling of the features in the seed data and guided by an ensemble of classifiers trained on the resampled feature vectors. We showed that our approach is able to detect a certain amount of noise in the data.

Future work should focus on finding optimal parameter settings to make the filtering method more robust even for noisier data sets. We also plan to improve the filtering heuristics and to explore further ways of detecting human coder errors. Finally, we plan to test our method in a real-world annotation scenario.

6 Acknowledgments

This work was funded by the German Research Foundation DFG (grant PI 154/9-3). We would like to thank the anonymous reviewers for their helpful comments and suggestions.

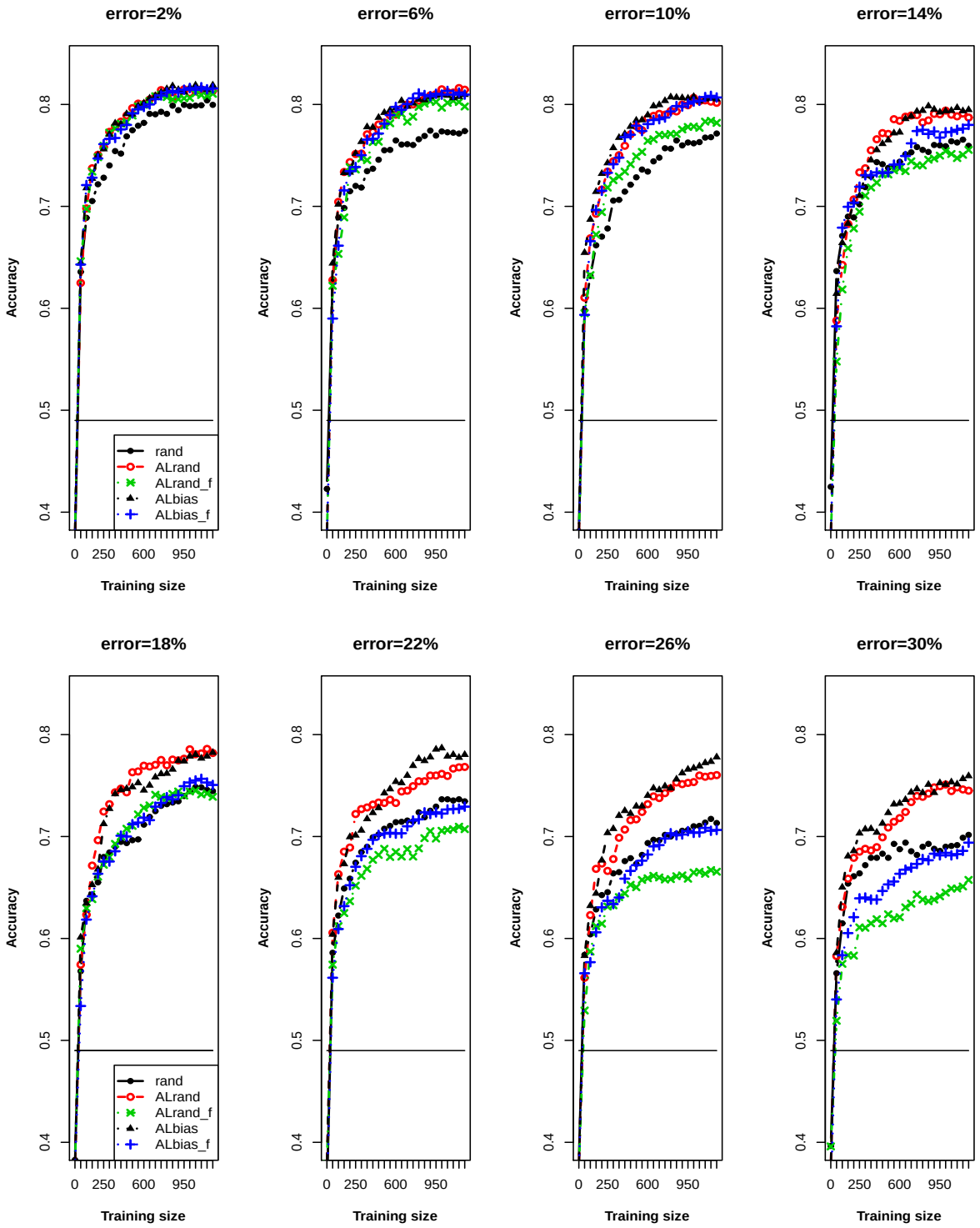


Figure 3: Active learning curves for varying degrees of noise, starting from 0% up to 30% for a training size up to 1200 instances (solid circle (black): random sampling; open circle (red): ALrand; cross (green): ALrand with filtering; filled triangle point-up (black): ALbias; plus (blue): ALbias with filtering)

References

Beata Beigman Klebanov and Eyal Beigman. 2009. From annotator agreement to noise models. *Computational Linguistics*, 35:495–503, December.

Beata Beigman Klebanov, Eyal Beigman, and Daniel Diermeier. 2008. Analyzing disagreements. In *Proceedings of the Workshop on Human Judgements in Computational Linguistics*, HumanJudge '08, pages 2–7, Morristown, NJ, USA. Association for Computational Linguistics.

Yee Seng Chan and Hwee Tou Ng. 2007. Domain adaptation with active learning for word sense disambiguation. In *Proceedings of ACL-2007*.

Jinying Chen, Andrew Schein, Lyle Ungar, and Martha Palmer. 2006. An empirical study of the behavior of active learning for word sense disambiguation. In *Proceedings of NAACL-2006*, New York, NY.

Hoa Trang Dang. 2004. *Investigations into the role of lexical semantics in word sense disambiguation*. PhD dissertation, University of Pennsylvania, Pennsylvania, PA.

Gholamreza Haffari and Anoop Sarkar. 2009. Active learning for multilingual statistical machine translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, pages 181–189. Association for Computational Linguistics.

Rebecca Hwa. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30(3):253–276.

Florian Laws and H. Schütze. 2008. Stopping criteria for active learning of named entity recognition. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, Manchester, UK, August.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of ACM-SIGIR*, Dublin, Ireland.

Grace Ngai and David Yarowsky. 2000. Rule writing or annotation: cost-efficient resource usage for base noun phrase chunking. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 117–125, Stroudsburg, PA, USA. Association for Computational Linguistics.

Miles Osborne and Jason Baldridge. 2004. Ensemble-based active learning for parse selection. In *Proceedings of HLT-NAACL 2004*.

Ines Rehbein, Josef Ruppenhofer, and Jonas Sunde. 2009. Majo - a toolkit for supervised word sense disambiguation and active learning. In *Proceedings of the 8th Workshop on Treebanks and Linguistic Theories (TLT-8)*, Milano, Italy.

Ines Rehbein, Josef Ruppenhofer, and Alexis Palmer. 2010. Bringing active learning to life. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, Beijing, China.

Dennis Reidsma and Jean Carletta. 2008. Reliability measurement without limits. *Computational Linguistics*, 34:319–326.

Eric Ringger, Peter McClanahan, Robbie Haertel, George Busby, Marc Carmen, James Carroll, Kevin Seppi, and Deryle Lonsdale. 2007. Active learning for part-of-speech tagging: Accelerating corpus annotation. In *Proceedings of the Linguistic Annotation Workshop*, Prague.

Andrew I. Schein and Lyle H. Ungar. 2007. Active learning for logistic regression: an evaluation. *Machine Learning*, 68:235–265.

Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. 2004. Multi-criteria-based active learning for named entity recognition. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, Stroudsburg, PA, USA. Association for Computational Linguistics.

Katrin Tomanek and Udo Hahn. 2009. Reducing class imbalance during active learning for named entity annotation. In *Proceedings of the 5th International Conference on Knowledge Capture*, Redondo Beach, CA.

Simon Tong and Daphne Koller. 1998. Support vector machine active learning with applications to text classification. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-00)*, pages 287–295.

Cha Zhang and Tsuhan Chen. 2002. An active learning framework for content-based information retrieval. *IEEE Transactions on Multimedia*, 4(2):260–268.

Jingbo Zhu and Edward Hovy. 2007. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Prague, Czech Republic.

Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K. Tsou. 2008. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, Manchester, UK.

A Fast and Accurate Method for Approximate String Search

Ziqi Wang*
School of EECS
Peking University
Beijing 100871, China
wangziqi@pku.edu.cn

Gu Xu
Microsoft Research Asia
Building 2, No.5 Danling Street,
Beijing 100080, China
guxu@microsoft.com

Hang Li
Microsoft Research Asia
Building 2, No.5 Danling Street,
Beijing 100080, China
hangli@microsoft.com

Ming Zhang
School of EECS
Peking University
Beijing 100871, China
mzhang@net.pku.edu.cn

Abstract

This paper proposes a new method for approximate string search, specifically candidate generation in spelling error correction, which is a task as follows. Given a misspelled word, the system finds words in a dictionary, which are most "similar" to the misspelled word. The paper proposes a probabilistic approach to the task, which is both accurate and efficient. The approach includes the use of a log linear model, a method for training the model, and an algorithm for finding the top k candidates. The log linear model is defined as a conditional probability distribution of a corrected word and a rule set for the correction conditioned on the misspelled word. The learning method employs the criterion in candidate generation as loss function. The retrieval algorithm is efficient and is guaranteed to find the optimal k candidates. Experimental results on large scale data show that the proposed approach improves upon existing methods in terms of accuracy in different settings.

1 Introduction

This paper addresses the following problem, referred to as approximate string search. Given a query string, a dictionary of strings (vocabulary), and a set of operators, the system returns the top k strings in the dictionary that can be transformed from the query string by applying several operators in the operator set. Here each operator is a rule that can replace a substring in the query string with another substring. The top k results are defined in terms of an evaluation measure employed in a specific application. The requirement is that the task must be conducted very efficiently.

Approximate string search is useful in many applications including spelling error correction, similar terminology retrieval, duplicate detection, etc. Although certain progress has been made for addressing the problem, further investigation on the task is still necessary, particularly from the viewpoint of *enhancing both accuracy and efficiency*.

Without loss of generality, in this paper we address candidate generation in spelling error correction. Candidate generation is to find the most possible corrections of a misspelled word. In such a problem, strings are words, and the operators represent insertion, deletion, and substitution of characters with or without surrounding characters, for example, "a"→"e" and "lly"→"ly". Note that candidate generation is concerned with a single word; after candidate generation, the words surrounding it in the text can be further leveraged to make the final *candidate selection*, e.g., Li et al. (2006), Golding and Roth (1999).

In spelling error correction, Brill and Moore (2000) proposed employing a generative model for candidate generation and a hierarchy of trie structures for fast candidate retrieval. Our approach is a discriminative approach and is aimed at improving Brill and Moore's method. Okazaki *et al.* (2008) proposed using a logistic regression model for approximate dictionary matching. Their method is also a discriminative approach, but it is largely different from our approach in the following points. It formalizes the problem as binary classification and

* Contribution during internship at Microsoft Research Asia.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 52–61,
Portland, Oregon, June 19-24, 2011.

assumes that there is only one rule applicable each time in candidate generation. Efficiency is also not a major concern for them, because it is for offline text mining.

There are two fundamental problems in research on approximate string search: (1) how to build a model that can archive both high accuracy and efficiency, and (2) how to develop a data structure and algorithm that can facilitate efficient retrieval of the top k candidates.

In this paper, we propose a probabilistic approach to the task. Our approach is novel and unique in the following aspects. It employs (a) a log-linear (discriminative) model for candidate generation, (b) an effective algorithm for model learning, and (c) an efficient algorithm for candidate retrieval.

The log linear model is defined as a conditional probability distribution of a corrected word and a rule set for the correction given the misspelled word. The learning method employs, in the training process, a criterion that represents the goal of making both accurate and efficient prediction (candidate generation). As a result, the model is optimally trained toward its objective. The retrieval algorithm uses special data structures and efficiently performs the top k candidates finding. It is guaranteed to find the best k candidates without enumerating all the possible ones.

We empirically evaluated the proposed method in spelling error correction of web search queries. The experimental results have verified that the accuracy of the top candidates given by our method is significantly higher than those given by the baseline methods. Our method is more accurate than the baseline methods in different settings such as large rule sets and large vocabulary sizes. The efficiency of our method is also very high in different experimental settings.

2 Related Work

Approximate string search has been studied by many researchers. Previous work mainly focused on efficiency rather than model. Usually, it is assumed that the model (similarity distance) is fixed and the goal is to efficiently find all the strings in the collection whose similarity distances are within a threshold. Most existing methods employ n-gram based algorithms (Behm et al., 2009; Li et al., 2007; Yang et al., 2008) or filtering algorithms (Mihov and Schulz, 2004; Li et al., 2008). Instead of finding all the candidates in a fixed range, methods for finding the top k candidates have also been developed. For example, the method by Vernica and Li (2009) utilized n-gram based inverted lists as index structure and a similarity function based on n-gram overlaps and word frequencies. Yang et al. (2010) presented a general framework for top k retrieval based on n-grams. In contrast, our work in this paper aims to learn a ranking function which can achieve both high accuracy and efficiency.

Spelling error correction normally consists of candidate generation and candidate final selection. The former task is an example of approximate string search. Note that candidate generation is only concerned with a single word. For single-word candidate generation, rule-based approach is commonly used. The use of edit distance is a typical example, which exploits operations of character deletion, insertion and substitution. Some methods generate candidates within a fixed range of edit distance or different ranges for strings with different lengths (Li et al., 2006; Whitelaw et al., 2009). Other methods make use of weighted edit distance to enhance the representation power of edit distance (Ristad and Yianilos, 1998; Oncina and Sebban, 2005; McCallum et al., 2005; Ahmad and Kondrak, 2005).

Conventional edit distance does not take in consideration context information. For example, people tend to misspell "c" to "s" or "k" depending on contexts, and a straightforward application of edit distance cannot deal with the problem. To address the challenge, some researchers proposed using a large number of substitution rules containing context information (at character level). For example, Brill and Moore (2000) developed a generative model including contextual substitution rules; and Toutanova and Moore (2002) further improved the model by adding pronunciation factors into the model. Schaback and Li (2007) proposed a multi-level feature-based framework for spelling error correction including a modification of Brill and Moore's model (2000). Okazaki et al. (2008) utilized substring substitution rules and incorporated the rules into a L_1-regularized logistic regression model. Okazaki et al.'s model is largely different

from the model proposed in this paper, although both of them are discriminative models. Their model is a binary classification model and it is assumed that only a single rule is applied in candidate generation.

Since users' behavior of misspelling and correction can be frequently observed in web search log data, it has been proposed to mine spelling-error and correction pairs by using search log data. The mined pairs can be directly used in spelling error correction. Methods of selecting spelling and correction pairs with maximum entropy model (Chen et al., 2007) or similarity functions (Islam and Inkpen, 2009; Jones et al., 2006) have been developed. The mined pairs *can only be used in candidate generation of high frequency typos*, however. In this paper, we work on candidate generation *at the character level*, which can be applied to spelling error correction for both high and low frequency words.

3 Model for Candidate Generation

As an example of approximate string search, we consider candidate generation in spelling correction. Suppose that there is a vocabulary $\mathcal{V}$ and a misspelled word, the objective of candidate generation is to select the best corrections from the vocabulary $\mathcal{V}$. We care about both accuracy and efficiency of the process. The problem is very challenging when the size of vocabulary is large, because there are a large number of potential candidates to be verified.

In this paper, we propose a probabilistic approach to candidate generation, which can achieve both high accuracy and efficiency, and is particularly powerful when the scale is large.

In our approach, it is assumed that a large number of misspelled words and their best corrections are given as training data. A probabilistic model is then trained by using the training data, which can assign ranking scores to candidates. The best candidates for correction of a misspelled word are thus defined as those candidates having the highest probabilistic scores with respect to the training data and the operators.

Hereafter, we will describe the probabilistic model for candidate generation, as well as training and exploitation of the model.

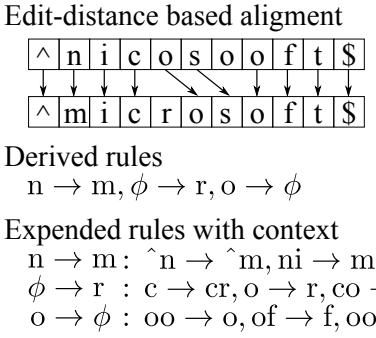


Figure 1: Example of rule extraction from word pair

3.1 Model

The operators (rules) represent insertion, deletion, and substitution of characters in a word with or without surrounding context (characters), which are similar to those defined in (Brill and Moore, 2000; Okazaki et al., 2008). An operator is formally represented a rule $\alpha \rightarrow \beta$ that replaces a substring α in a misspelled word with β, where $\alpha, \beta \in \{s | s = t, s = \hat{}t$, or $s = t\$\}$ and $t \in \Sigma^*$ is the set of all possible strings over the alphabet. Obviously, $\mathcal{V} \subset \Sigma^*$. We actually derive all the possible rules from the training data using a similar approach to (Brill and Moore, 2000) as shown in Fig. 1. First we conduct the letter alignment based on the minimum edit-distance, and then derive the rules from the alignment. Furthermore we expand the derived rules with surrounding words. Without loss of generality, we only consider using $+2, +1, 0, -1, -2$ characters as contexts in this paper.

If we can apply a set of rules to transform the misspelled word w_m to a correct word w_c in the vocabulary, then we call the rule set a "transformation" for the word pair w_m and w_c. Note that for a given word pair, it is likely that there are multiple possible transformations for it. For example, both "n"→"m" and "ni"→"mi" can transform "nicrosoft" to "microsoft".

Without loss of generality, we set the maximum number of rules applicable to a word pair to be a fixed number. As a result, the number of possible transformations for a word pair is finite, and usually limited. This is equivalent to the assumption that the number of spelling errors in a word is small.

Given word pair (w_m, w_c), let $R(w_m, w_c)$ denote one transformation (a set of rules) that can rewrite

w_m to w_c. We consider that there is a probabilistic mapping between the misspelled word w_m and correct word w_c plus transformation $R(w_m, w_c)$. We define the conditional probability distribution of w_c and $R(w_m, w_c)$ given w_m as the following log linear model:

$$P(w_c, R(w_m, w_c)|w_m) \quad (1)$$
$$= \frac{\exp\left(\sum_{r \in R(w_m, w_c)} \lambda_r\right)}{\sum_{(w'_c, R(w_m, w'_c)) \in \mathcal{Z}(w_m)} \exp\left(\sum_{o \in R(w_m, w'_c)} \lambda_o\right)}$$

where r or o denotes a rule in rule set R, λ_r or λ_o denotes a weight, and the normalization is carried over $\mathcal{Z}(w_m)$, all pairs of word w'_c in $\mathcal{V}$ and transformation $R(w_m, w'_c)$, such that w_m can be transformed to w'_c by $R(w_m, w'_c)$. The log linear model actually uses binary features indicating whether or not a rule is applied.

In general, the weights in Equ. (1) can be any real numbers. To improve efficiency in retrieval, we further assume that all the weights are non-positive, i.e., $\forall \lambda_r \leq 0$. It introduces monotonicity in rule application and implies that applying additional rules cannot lead to generation of better candidates. For example, both "office" and "officer" are correct candidates of "ofice". We view "office" a better candidate (with higher probability) than "officer", as it needs one less rule. The assumption is reasonable because the chance of making more errors should be lower than that of making less errors. Our experimental results have shown that the change in accuracy by making the assumption is negligible, but the gain in efficiency is very large.

3.2 Training of Model

Training data is given as a set of pairs $\mathcal{T} = \left\{(w_m^i, w_c^i)\right\}_{i=1}^N$, where w_m^i is a misspelled word and $w_c^i \in \mathcal{V}$ is a correction of w_m^i. The objective of training would be to maximize the conditional probability $P(w_c^i, R(w_m^i, w_c^i)|w_m^i)$ over the training data.

This is not a trivial problem, however, because the "true" transformation $R^*(w_m^i, w_c^i)$ for each word pair w_m^i and w_c^i is not given in the training data. It is often the case that there are multiple transformations applicable, and it is not realistic to assume that such information can be provided by humans or automatically derived. (It is relatively easy to automatically find the pairs w_m^i and w_c^i as explained in Section 5.1).

In this paper, we assume that the transformation that actually generates the correction among all the possible transformations is the one that can give the maximum conditional probability; the exactly same criterion is also used for fast prediction. Therefore we have the following objective function

$$\lambda^* = \arg\max_{\lambda} L(\lambda) \quad (2)$$
$$= \arg\max_{\lambda} \sum_i \max_{R(w_m^i, w_c^i)} \log P(w_c^i, R(w_m^i, w_c^i)|w_m^i)$$

where λ denotes the weight parameters and the $\max$ is taken over the set of transformations that can transform w_m^i to w_c^i.

We employ gradient ascent in the optimization in Equ. (2). At each step, we first find the best transformation for each word pair based on the current parameters $\lambda^{(t)}$

$$R^*(w_m^i, w_c^i) \quad (3)$$
$$= \arg\max_{R(w_m^i, w_c^i)} \log P_{\lambda^{(t)}}(w_c^i, R(w_m^i, w_c^i)|w_m^i)$$

Next, we calculate the gradients,

$$\frac{\partial L}{\partial \lambda_r} = \frac{\sum_i \log P_{\lambda^{(t)}}(w_c^i, R^*(w_m^i, w_c^i)|w_m^i)}{\partial \lambda_r} \quad (4)$$

In this paper, we employ the bounded L-BFGS (Behm et al., 2009) algorithm for the optimization task, which works well even when the number of weights λ is large.

3.3 Candidate Generation

In candidate generation, given a misspelled word w_m, we find the k candidates from the vocabulary, that can be transformed from w_m and have the largest probabilities assigned by the learned model.

We only need to utilize the following ranking function to rank a candidate w_c given a misspelled word w_m, by taking into account Equs. (1) and (2)

$$\text{rank}(w_c|w_m) = \max_{R(w_m, w_c)} \left(\sum_{r \in R(w_m, w_c)} \lambda_r \right) \quad (5)$$

For each possible transformation, we simply take summation of the weights of the rules used in the transformation. We then choose the sum as a ranking score, which is equivalent to ranking candidates based on their largest conditional probabilities.

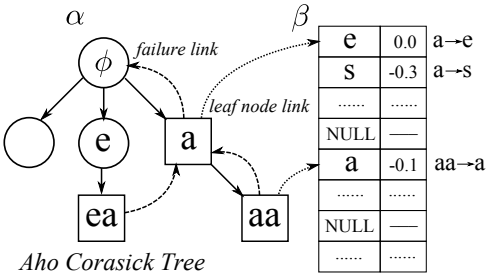


Figure 2: Rule Index based on Aho Corasick Tree.

4 Efficient Retrieval Algorithm

In this section, we introduce how to efficiently perform top k candidate generation. Our retrieval algorithm is guaranteed to find the optimal k candidates with some "pruning" techniques. We first introduce the data structures and then the retrieval algorithm.

4.1 Data Structures

We exploit two data structures for candidate generation. One is a trie for storing and matching words in the vocabulary, referred to as vocabulary trie, and the other based on what we call an Aho-Corasick tree (AC tree) (Aho and Corasick, 1975), which is used for storing and applying correction rules, referred to as rule index. The vocabulary trie is the same as that used in existing work and it will be traversed when searching the top k candidates.

Our rule index is unique because it indexes all the rules based on an AC tree. The AC tree is a trie with "failure links", on which the Aho-Corasick string matching algorithm can be executed. Aho-Corasick algorithm is a well known dictionary-matching algorithm which can quickly locate all the words in a dictionary within an input string. Time complexity of the algorithm is of linear order in length of input string plus number of matched entries.

We index all the α's in the rules on the AC tree. Each α corresponds to a leaf node, and the β's of the α are stored in an associated list in decreasing order of rule weights λ, as illustrated in Fig. 2. [1]

[1]One may further improve the index structure by using a trie rather than a ranking list to store βs associated with the same α. However the improvement would not be significant because the number of βs associated with each α is usually very small.

4.2 Algorithm

One could employ a naive algorithm that applies all the possible combinations of rules (α's) to the current word w_m, verifies whether the resulting words (candidates) are in the vocabulary, uses the function in Equ. (5) to calculate the ranking scores of the candidates, and find the top k candidates. This algorithm is clearly inefficient.

Our algorithm first employs the Aho-Corasick algorithm to locate all the applicable α's within the input word w_m, from the rule index. The corresponding β's are retrieved as well. Then all the applicable rules are identified and indexed by the applied positions of word w_m.

Our algorithm next traverses the vocabulary trie and searches the top k candidates with some pruning techniques. The algorithm starts from the root node of the vocabulary trie. At each step, it has multiple search branches. It tries to match at the next position of w_m, or apply a rule at the current position of w_m. The following two pruning criteria are employed to significantly accelerate the search process.

1) If the current sum of weights of applied rules is smaller than the smallest weight in the top k list, the search branch is pruned. This criterion is derived from the non-negative constraint on rule weights λ. It is easy to verify that the sum of weights will not become larger if one continues to search the branch because all the weights are non-positive.
2) If two search branches merge at the same node in the vocabulary trie as well as the same position on w_m, the search branches with smaller sum of weights will be pruned. It is based on the dynamic programming technique because we take $\max$ in the ranking function in Equ. 5.

It is not difficult to prove that our algorithm is guaranteed to find the best k candidates in terms of the ranking scores, because we only prune those candidates that cannot give better scores than the ones in the current top k list. Due to the limitation of space, we omit the proof of the theorem that if the weights of rules λ are non-positive and the ranking function is defined as in Equ. 5, then the top k candidates obtained with the pruning criteria are the same as the top k candidates obtained without pruning.

5 Experimental Results

We have experimentally evaluated our approach in spelling error correction of queries in web search. The problem is more challenging than usual due to the following reasons. (1) The vocabulary of queries in web search is extremely large due to the scale, diversity, and dynamics of the Internet. (2) Efficiency is critically important, because the response time of top k candidate retrieval for web search must be kept very low. Our approach for candidate generation is in fact motivated by the application.

5.1 Word Pair Mining

In web search, a search session is comprised of a sequence of queries from the same user within a time period. It is easy to observe from search session data that there are many spelling errors and their corrections occurring in the same sessions. We employed heuristics to automatically mine training pairs from search session data at a commercial search engine.

First, we segmented the query sequence from each user into sessions. If two queries were issued more than 5 minutes apart, then we put a session boundary between them. We used short sessions here because we found that search users usually correct their misspelled queries very quickly after they find the misspellings. Then the following heuristics were employed to identify pairs of misspelled words and their corrections from two consecutive queries within a session:

1) Two queries have the same number of words.
2) There is only one word difference between two queries.
3) For the two distinct words, the word in the first query is considered as misspelled and the second one as its correction.

Finally, we aggregated the identified training pairs across sessions and users and discarded the pairs with low frequencies. Table 1 shows some examples of the mined word pairs.

Misspelled	Correct	Misspelled	Correct
aacoustic	acoustic	chevorle	chevrolet
liyerature	literature	tournemen	tournament
shinngle	shingle	newpape	newspaper
finlad	finland	ccomponet	component
reteive	retrieve	olimpick	olympic

Table 1: Examples of Word Pairs

5.2 Experiments on Accuracy

Two representative methods were used as baselines: the generative model proposed by (Brill and Moore, 2000) referred to as *generative* and the logistic regression model proposed by (Okazaki et al., 2008) referred to as *logistic*. Note that Okazaki et al. (2008)'s model is not particularly for spelling error correction, but it can be employed in the task. When using their method for ranking, we used outputs of the logistic regression model as rank scores.

We compared our method with the two baselines in terms of top k accuracy, which is ratio of the true corrections among the top k candidates generated by a method. All the methods shared the same settings: 973,902 words in the vocabulary, 10,597 rules for correction, and up to two rules used in one transformation. We made use of 100,000 word pairs mined from query sessions for training, and 10,000 word pairs for testing.

The experimental results are shown in Fig. 3. We can see that our method always performs the best when compared with the baselines and the improvements are statistically significant ($p < 0.01$). The *logistic* method works better than *generative*, when k is small, but its performance becomes saturated, when k is large. Usually a discriminative model works better than a generative model, and that seems to be what happens with small k's. However, *logistic* cannot work so well for large k's, because it only allows the use of one rule each time. We observe that there are many word pairs in the data that need to be transformed with multiple rules.

Next, we conducted experiments to investigate how the top k accuracy changes with different sizes of vocabularies, maximum numbers of applicable rules and sizes of rule set for the three methods. The experimental results are shown in Fig. 4, Fig. 5 and Fig. 6.

For the experiment in Fig. 4, we enlarged the vocabulary size from 973,902 (smallVocab) to 2,206,948 (largeVocab) and kept the other settings the same as in the previous experiment. Because more candidates can be generated with a larger vocabulary, the performances of all the methods de-

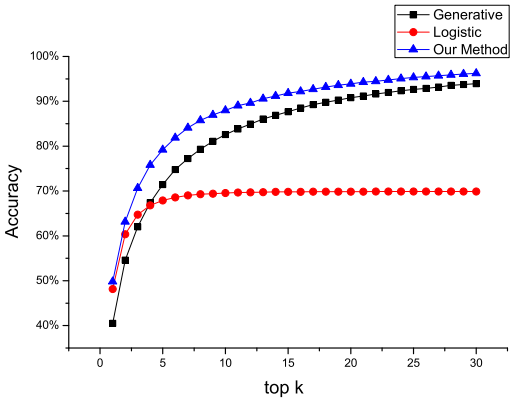


Figure 3: Accuracy Comparison between Our Method and Baselines

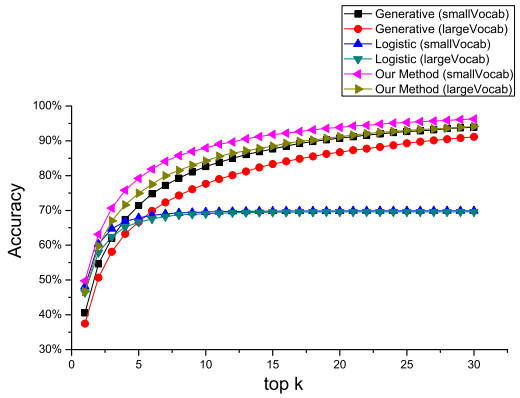


Figure 4: Accuracy Comparisons between Baselines and Our Method with Different Vocabulary Sizes

cline. However, the drop of accuracy by our method is much smaller than that by *generative*, which means our method is more powerful when the vocabulary is large, e.g., for web search. For the experiment in Fig. 5, we changed the maximum number of rules that can be applied to a transformation from 2 to 3. Because *logistic* can only use one rule at a time, it is not included in this experiment. When there are more applicable rules, more candidates can be generated and thus ranking of them becomes more challenging. The accuracies of both methods drop, but our method is constantly better than *generative*. Moreover, the decrease in accuracy by our method is clearly less than that by *generative*. For the experiment in Fig. 6, we enlarged the number of rules from 10,497 (smallRuleNum) to 24,054 (largeRuleNum). The performance of our method and those of the two baselines did not change so much, and our method still visibly outperform the baselines when more rules are exploited.

5.3 Experiments on Efficiency

We have also experimentally evaluated the efficiency of our approach. Because most existing work uses a predefined ranking function, it is not fair to make a comparison with them. Moreover, Okazaki et al.' method does not consider efficiency, and Brill and Moore's method is based a complicated retrieve algorithm which is very hard to implement. Instead of making comparison with the existing methods in terms of efficiency, we evaluated the efficiency of our method by looking at how efficient it becomes with its data structure and pruning technique.

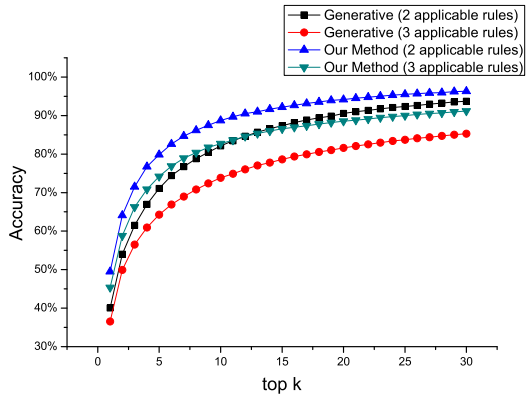


Figure 5: Accuracy Comparison between Generative and Our Method with Different Maximum Numbers of Applicable Rules

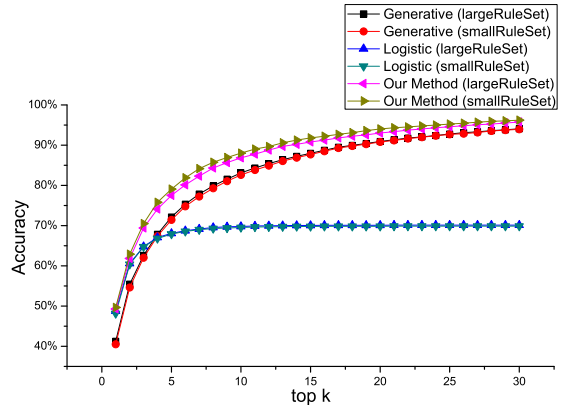


Figure 6: Accuracy Comparison between Baselines and Our Method with Different Numbers of Rules

First, we tested the efficiency of using Aho-Corasick algorithm (the rule index). Because the

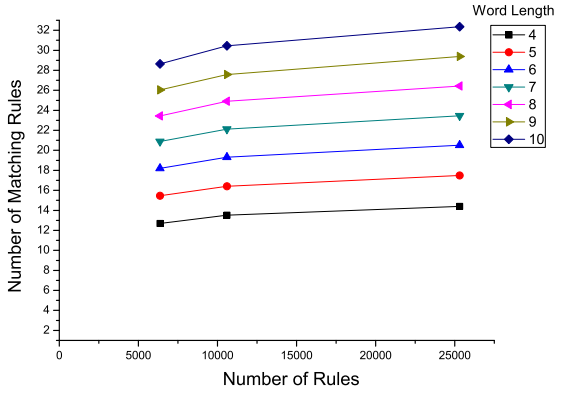


Figure 7: Number of Matching Rules v.s. Number of Rules

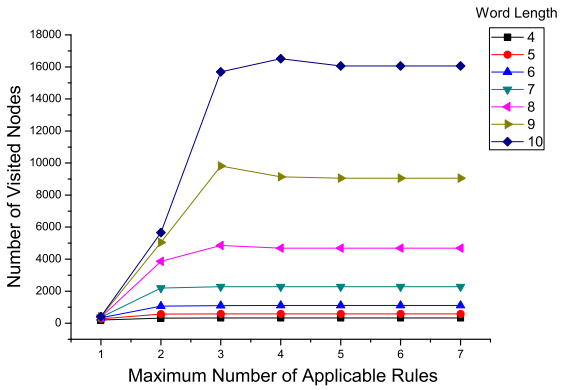


Figure 8: Efficiency Evaluation with Different Maximum Numbers of Applicable Rules

time complexity of Aho-Corasick algorithm is determined by the lengths of query strings and the number of matches, we examined how the number of matches on query strings with different lengths changes when the number of rules increases. The experimental results are shown in Fig. 7. We can see that the number of matches is not largely affected by the number of rules in the rule index. It implies that the time for searching applicable rules is close to a constant and does not change much with different numbers of rules.

Next, since the running time of our method is proportional to the number of visited nodes on the vocabulary trie, we evaluated the efficiency of our method in terms of number of visited nodes. The result reported here is that when k is 10.

Specifically, we tested how the number of visited nodes changes according to three factors: maximum number of applicable rules in a transformation, vocabulary size and rule set size. The experimental results are shown in Fig. 8, Fig. 9 and Fig. 10 respectively. From Fig. 8, with increasing maximum number of applicable rules in a transformation, number of visited nodes increases first and then stabilizes, especially when the words are long. Note that pruning becomes even more effective because number of visited nodes without pruning grows much faster. It demonstrates that our method is very efficient when compared to the non-pruning method. Admittedly, the efficiency of our method also deteriorates somewhat. This would not cause a noticeable issue in real applications, however. In the previous section,

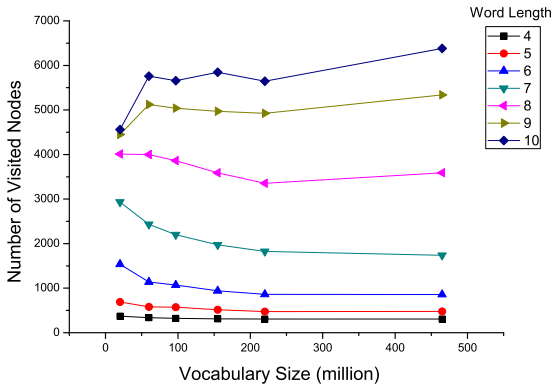


Figure 9: Efficiency Evaluation with Different Sizes of Vocabulary

we have seen that using up to two rules in a transformation can bring a very high accuracy. From Fig. 8 and Fig. 9, we can conclude that the numbers of visited nodes are stable and thus the efficiency of our method keeps high with larger vocabulary size and number of rules. It indicates that our pruning strategy is very effective. From all the figures, we can see that our method is always efficient especially when the words are relatively short.

5.4 Experiments on Model Constraints

In Section 3.1, we introduce the non-positive constraints on the parameters, i.e., $\forall \lambda_r \leq 0$, to enable the pruning technique for efficient top k retrieval. We experimentally verified the impact of the constraints to both the accuracy and efficiency. For ease of reference, we name the model with the non-positive constraints as *bounded*, and the origi-

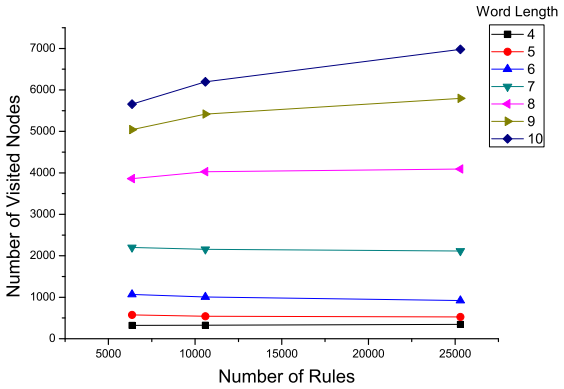


Figure 10: Efficiency Evaluation with Different Number of Rules

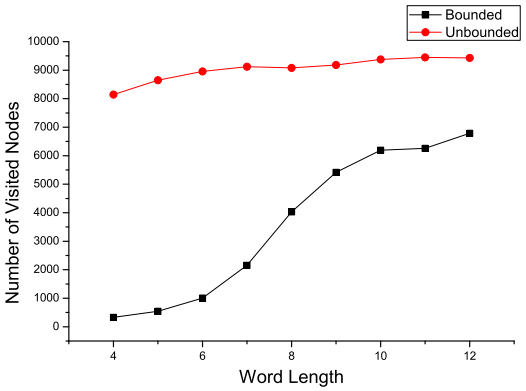


Figure 12: Efficiency Comparison between Bounded and Unbounded Models

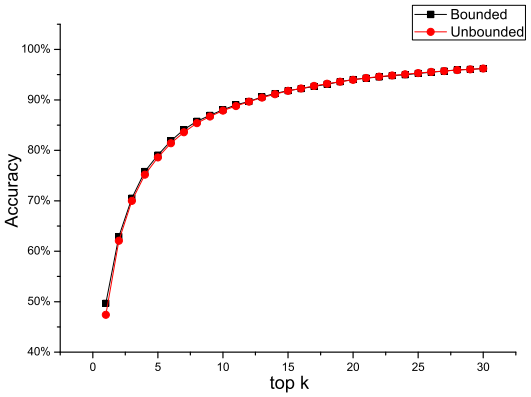


Figure 11: Accuracy Comparison between Bounded and Unbounded Models

nal model as *unbounded*. The experimental results are shown in Fig. 11 and Fig. 12. All the experiments were conducted based on the typical setting of our experiments: 973,902 words in the vocabulary, 10,597 rules, and up to two rules in one transformation. In Fig. 11, we can see that the difference between *bounded* and *unbounded* in terms of accuracy is negligible, and we can draw a conclusion that adding the constraints does not hurt the accuracy. From Fig. 12, it is easy to note that *bounded* is much faster than *unbounded* because our pruning strategy can be applied to *bounded*.

6 Conclusion

In this paper, we have proposed a new method for approximate string search, including spelling error correction, which is both accurate and efficient. Our method is novel and unique in its model, learning algorithm, and retrieval algorithm. Experimental results on a large data set show that our method improves upon existing methods in terms of accuracy, and particularly our method can perform better when the dictionary is large and when there are many rules. Experimental results have also verified the high efficiency of our method. As future work, we plan to add contextual features into the model and apply our method to other data sets in other tasks.

References

Farooq Ahmad and Grzegorz Kondrak. 2005. Learning a spelling error model from search query logs. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 955–962, Morristown, NJ, USA. Association for Computational Linguistics.

Alfred V. Aho and Margaret J. Corasick. 1975. Efficient string matching: an aid to bibliographic search. *Commun. ACM*, 18:333–340, June.

Alexander Behm, Shengyue Ji, Chen Li, and Jiaheng Lu. 2009. Space-constrained gram-based indexing for efficient approximate string search. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 604–615, Washington, DC, USA. IEEE Computer Society.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 286–293, Morristown, NJ, USA. Association for Computational Linguistics.

Qing Chen, Mu Li, and Ming Zhou. 2007. Improving query spelling correction using web search re-

sults. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 181–189.

Andrew R. Golding and Dan Roth. 1999. A winnow-based approach to context-sensitive spelling correction. *Mach. Learn.*, 34:107–130, February.

Aminul Islam and Diana Inkpen. 2009. Real-word spelling correction using google web it 3-grams. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1241–1249, Morristown, NJ, USA. Association for Computational Linguistics.

Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 387–396, New York, NY, USA. ACM.

Mu Li, Yang Zhang, Muhua Zhu, and Ming Zhou. 2006. Exploring distributional similarity based models for query spelling correction. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 1025–1032, Morristown, NJ, USA. Association for Computational Linguistics.

Chen Li, Bin Wang, and Xiaochun Yang. 2007. Vgram: improving performance of approximate queries on string collections using variable-length grams. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 303–314. VLDB Endowment.

Chen Li, Jiaheng Lu, and Yiming Lu. 2008. Efficient merging and filtering algorithms for approximate string searches. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 257–266, Washington, DC, USA. IEEE Computer Society.

Andrew McCallum, Kedar Bellare, and Fernando Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *Conference on Uncertainty in AI (UAI)*.

Stoyan Mihov and Klaus U. Schulz. 2004. Fast approximate search in large dictionaries. *Comput. Linguist.*, 30:451–477, December.

Naoaki Okazaki, Yoshimasa Tsuruoka, Sophia Ananiadou, and Jun'ichi Tsujii. 2008. A discriminative candidate generator for string transformations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 447–456, Morristown, NJ, USA. Association for Computational Linguistics.

Jose Oncina and Marc Sebban. 2005. Learning unbiased stochastic edit distance in the form of a memoryless finite-state transducer. In *International Joint Conference on Machine Learning (2005). Workshop: Grammatical Inference Applications: Successes and Future Challenges*.

Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string-edit distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20:522–532, May.

Johannes Schaback and Fang Li. 2007. Multi-level feature extraction for spelling correction. In *IJCAI-2007 Workshop on Analytics for Noisy Unstructured Text Data*, pages 79–86, Hyderabad, India.

Kristina Toutanova and Robert C. Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 144–151, Morristown, NJ, USA. Association for Computational Linguistics.

Rares Vernica and Chen Li. 2009. Efficient top-k algorithms for fuzzy search in string collections. In *Proceedings of the First International Workshop on Keyword Search on Structured Data*, KEYS '09, pages 9–14, New York, NY, USA. ACM.

Casey Whitelaw, Ben Hutchinson, Grace Y. Chung, and Gerard Ellis. 2009. Using the web for language independent spellchecking and autocorrection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 890–899, Morristown, NJ, USA. Association for Computational Linguistics.

Xiaochun Yang, Bin Wang, and Chen Li. 2008. Cost-based variable-length-gram selection for string collections to support approximate queries efficiently. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 353–364, New York, NY, USA. ACM.

Zhenglu Yang, Jianjun Yu, and Masaru Kitsuregawa. 2010. Fast algorithms for top-k approximate string matching. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI '10, pages 1467–1473.

Domain Adaptation by Constraining Inter-Domain Variability of Latent Feature Representation

Ivan Titov
Saarland University
Saarbruecken, Germany
titov@mmci.uni-saarland.de

Abstract

We consider a semi-supervised setting for domain adaptation where only unlabeled data is available for the target domain. One way to tackle this problem is to train a generative model with latent variables on the mixture of data from the source and target domains. Such a model would cluster features in both domains and ensure that at least some of the latent variables are predictive of the label on the source domain. The danger is that these predictive clusters will consist of features specific to the source domain only and, consequently, a classifier relying on such clusters would perform badly on the target domain. We introduce a constraint enforcing that marginal distributions of each cluster (i.e., each latent variable) do not vary significantly across domains. We show that this constraint is effective on the sentiment classification task (Pang et al., 2002), resulting in scores similar to the ones obtained by the structural correspondence methods (Blitzer et al., 2007) without the need to engineer auxiliary tasks.

1 Introduction

Supervised learning methods have become a standard tool in natural language processing, and large training sets have been annotated for a wide variety of tasks. However, most learning algorithms operate under assumption that the learning data originates from the same distribution as the test data, though in practice this assumption is often violated. This difference in the data distributions normally results in a significant drop in accuracy. To address this problem a number of *domain-adaptation* methods has recently been proposed (see e.g., (Daumé and Marcu, 2006; Blitzer et al., 2006; Bickel et al., 2007)). In addition to the labeled data from the source domain, they also exploit small amounts of labeled data and/or unlabeled data from the target domain to estimate a more predictive model for the target domain.

In this paper we focus on a more challenging and arguably more realistic version of the domain-adaptation problem where only unlabeled data is available for the target domain. One of the most promising research directions on domain adaptation for this setting is based on the idea of inducing a *shared feature representation* (Blitzer et al., 2006), that is mapping from the initial feature representation to a new representation such that (1) examples from both domains 'look similar' and (2) an accurate classifier can be trained in this new representation. Blitzer et al. (2006) use auxiliary tasks based on unlabeled data for both domains (called pivot features) and a dimensionality reduction technique to induce such shared representation. The success of their domain-adaptation method (Structural Correspondence Learning, SCL) crucially depends on the choice of the auxiliary tasks, and defining them can be a non-trivial engineering problem for many NLP tasks (Plank, 2009). In this paper, we investigate methods which do not use auxiliary tasks to induce a shared feature representation.

We use generative latent variable models (LVMs) learned on all the available data: unlabeled data for both domains and on the labeled data for the source domain. Our LVMs use vectors of latent features

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 62–71,
Portland, Oregon, June 19-24, 2011.

to represent examples. The latent variables encode regularities observed on unlabeled data from both domains, and they are learned to be predictive of the labels on the source domain. Such LVMs can be regarded as composed of two parts: a mapping from initial (normally, word-based) representation to a new shared distributed representation, and also a classifier in this representation. The danger of this semi-supervised approach in the domain-adaptation setting is that some of the latent variables will correspond to clusters of features specific only to the source domain, and consequently, the classifier relying on this latent variable will be badly affected when tested on the target domain. Intuitively, one would want the model to induce only those features which generalize between domains. We encode this intuition by introducing a term in the learning objective which regularizes inter-domain difference in marginal distributions of each latent variable.

Another, though conceptually similar, argument for our method is coming from theoretical results which postulate that the drop in accuracy of an adapted classifier is dependent on the *discrepancy distance* between the source and target domains (Blitzer et al., 2008; Mansour et al., 2009; Ben-David et al., 2010). Roughly, the discrepancy distance is small when linear classifiers cannot distinguish between examples from different domains. A necessary condition for this is that the feature expectations do not vary significantly across domains. Therefore, our approach can be regarded as minimizing a coarse approximation of the discrepancy distance.

The introduced term regularizes model expectations and it can be viewed as a form of a generalized expectation (GE) criterion (Mann and McCallum, 2010). Unlike the standard GE criterion, where a model designer defines the prior for a model expectation, our criterion postulates that the model expectations should be similar across domains.

In our experiments, we use a form of Harmonium Model (Smolensky, 1986) with a single layer of binary latent variables. Though exact inference with this class of models is infeasible we use an efficient approximation (Bengio and Delalleau, 2007), which can be regarded either as a mean-field approximation to the reconstruction error or a deterministic version of the Contrastive Divergence sampling method (Hinton, 2002). Though such an estimator is biased, in practice, it yields accurate models. We explain how the introduced regularizer can be integrated into the stochastic gradient descent learning algorithm for our model.

We evaluate our approach on adapting sentiment classifiers on 4 domains: books, DVDs, electronics and kitchen appliances (Blitzer et al., 2007). The loss due to transfer to a new domain is very significant for this task: in our experiments it was approaching 9%, in average, for the non-adapted model. Our regularized model achieves 35% average relative error reduction with respect to the non-adapted classifier, whereas the non-regularized version demonstrates a considerably smaller reduction of 26%. Both the achieved error reduction and the absolute score match the results reported in (Blitzer et al., 2007) for the best version[1] of the SCL method (SCL-MI, 36%), suggesting that our approach is a viable alternative to SCL.

The rest of the paper is structured as follows. In Section 2 we introduce a model which uses vectors of latent variables to model statistical dependencies between the elementary features. In Section 3 we discuss its applicability in the domain-adaptation setting, and introduce constraints on inter-domain variability as a way to address the discovered limitations. Section 4 describes approximate learning and inference algorithms used in our experiments. In Section 5 we provide an empirical evaluation of the proposed method. We conclude in Section 6 with further examination of the related work.

2 The Latent Variable Model

The adaptation method advocated in this paper is applicable to any joint probabilistic model which uses *distributed representations*, i.e. vectors of latent variables, to abstract away from hand-crafted features. These models, for example, include Restricted Boltzmann Machines (Smolensky, 1986; Hinton, 2002) and Sigmoid Belief Networks (SBNs) (Saul et al., 1996) for classification and regression tasks, Factorial HMMs (Ghahramani and Jordan, 1997) for sequence labeling problems, Incremental SBNs for parsing problems (Titov and Henderson, 2007a),

[1] Among the versions which do not exploit labeled data from the target domain.

as well as different types of Deep Belief Networks (Hinton and Salakhutdinov, 2006). The power of these methods is in their ability to automatically construct new features from elementary ones provided by the model designer. This feature induction capability is especially desirable for problems where engineering features is a labor-intensive process (e.g., multilingual syntactic parsing (Titov and Henderson, 2007b)), or for multitask learning problems where the nature of interactions between the tasks is not fully understood (Collobert and Weston, 2008; Gesmundo et al., 2009).

In this paper we consider classification tasks, namely prediction of sentiment polarity of a user review (Pang et al., 2002), and model the joint distribution of the binary sentiment label $y \in \{0,1\}$ and the multiset of text features $\boldsymbol{x}, x_i \in \mathcal{X}$. The hidden variable vector $\boldsymbol{z}$ ($z_i \in \{0,1\}$, $i = 1,\dots,m$) encodes statistical dependencies between components of $\boldsymbol{x}$ and also dependencies between the label y and the features $\boldsymbol{x}$. Intuitively, the model can be regarded as a logistic regression classifier with latent features.

The model assumes that the features and the latent variable vector are generated jointly from a globally-normalized model and then the label y is generated from a conditional distribution dependent on $\boldsymbol{z}$. Both of these distributions, $P(\boldsymbol{x},\boldsymbol{z})$ and $P(y|\boldsymbol{z})$, are parameterized as log-linear models and, consequently, our model can be seen as a combination of an undirected Harmonium model (Smolensky, 1986) and a directed SBN model (Saul et al., 1996). The formal definition is as follows:

(1) Draw $(\boldsymbol{x},\boldsymbol{z}) \sim P(\boldsymbol{x},\boldsymbol{z}|\boldsymbol{v})$,
(2) Draw label $y \sim \sigma(w_0 + \sum_{i=1}^{m} w_i z_i)$,

where $\boldsymbol{v}$ and $\boldsymbol{w}$ are parameters, σ is the logistic sigmoid function, $\sigma(t) = 1/(1+e^{-t})$, and the joint distribution of $(\boldsymbol{x},\boldsymbol{z})$ is given by the Gibbs distribution:

$$P(\boldsymbol{x},\boldsymbol{z}|\boldsymbol{v}) \propto \exp(\sum_{j=1}^{|x|} v_{x_j 0} + \sum_{i=1}^{n} v_{0i} z_i + \sum_{j,i=1}^{|x|,n} v_{x_j i} z_i).$$

Figure 1 presents the corresponding graphical model. Note that the arcs between $\boldsymbol{x}$ and $\boldsymbol{z}$ are undirected, whereas arcs between y and $\boldsymbol{z}$ are directed.

The parameters of this model $\theta = (\boldsymbol{v},\boldsymbol{w})$ can be estimated by maximizing joint likelihood $L(\theta)$ of labeled data for the source domain $\{\boldsymbol{x}^{(l)}, y^{(l)}\}_{l \in S_L}$

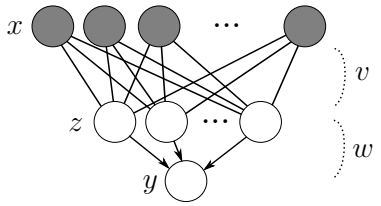


Figure 1: The latent variable model: $\boldsymbol{x}$, $\boldsymbol{z}$, y are random variables, dependencies between $\boldsymbol{x}$ and $\boldsymbol{z}$ are parameterized by matrix $\boldsymbol{v}$, and dependencies between $\boldsymbol{z}$ and y - by vector $\boldsymbol{w}$.

and unlabeled data for the source and target domain $\{\boldsymbol{x}^{(l)}\}_{l \in S_U \cup T_U}$, where S_U and T_U stand for the unlabeled datasets for the source and target domains, respectively. However, given that, first, amount of unlabeled data $|S_U \cup T_U|$ normally vastly exceeds the amount of labeled data $|S_L|$ and, second, the number of features for each example $|\boldsymbol{x}^{(l)}|$ is usually large, the label y will have only a minor effect on the mapping from the initial features $\boldsymbol{x}$ to the latent representation $\boldsymbol{z}$ (i.e. on the parameters $\boldsymbol{v}$). Consequently, the latent representation induced in this way is likely to be inappropriate for the classification task in question. Therefore, we follow (McCallum et al., 2006) and use a multi-conditional objective, a specific form of hybrid learning, to emphasize the importance of labels y:

$$L(\theta,\alpha) = \alpha \sum_{l \in S_L} \log P(y^{(l)}|x^{(l)},\theta) + \sum_{l \in S_U \cup T_U \cup S_L} \log P(x^{(l)}|\theta),$$

where α is a weight, $\alpha > 1$.

Direct maximization of the objective is problematic, as it would require summation over all the 2^m latent vectors $\boldsymbol{z}$. Instead we use a mean-field approximation. Similarly, an efficient approximate inference algorithm is used to compute $\arg\max_y P(y|x,\theta)$ at testing time. The approximations are described in Section 4.

3 Constraints on Inter-Domain Variability

As we discussed in the introduction, our goal is to provide a method for domain adaptation based on semi-supervised learning of models with distributed representations. In this section, we first discuss the shortcomings of domain adaptation with the above-described semi-supervised approach and motivate constraints on inter-domain variability of

the induced shared representation. Then we propose a specific form of this constraint based on the Kullback-Leibler (KL) divergence.

3.1 Motivation for the Constraints

Each latent variable z_i encodes a cluster or a combination of elementary features x_j. At least some of these clusters, when induced by maximizing the likelihood $L(\theta, \alpha)$ with sufficiently large α, will be useful for the classification task on the source domain. However, when the domains are substantially different, these predictive clusters are likely to be specific only to the source domain. For example, consider moving from reviews of electronics to book reviews: the cluster of features related to equipment reliability and warranty service will not generalize to books. The corresponding latent variable will always be inactive on the books domain (or always active, if negative correlation is induced during learning). Equivalently, the marginal distribution of this variable will be very different for both domains. Note that the classifier, defined by the vector $\boldsymbol{w}$, is only trained on the labeled source examples $\{\boldsymbol{x}^{(l)}, y^{(l)}\}_{l \in S_L}$ and therefore it will rely on such latent variables, even though they do not generalize to the target domain. Clearly, the accuracy of such classifier will drop when it is applied to target domain examples. To tackle this issue, we introduce a regularizing term which penalizes differences in the marginal distributions between the domains.

In fact, we do not need to consider the behavior of the classifier to understand the rationale behind the introduction of the regularizer. Intuitively, when adapting between domains, we are interested in representations $\boldsymbol{z}$ which explain domain-independent regularities rather than in modeling inter-domain differences. The regularizer favors models which focus on the former type of phenomena rather than the latter.

Another motivation for the form of regularization we propose originates from theoretical analysis of the domain adaptation problems (Ben-David et al., 2010; Mansour et al., 2009; Blitzer et al., 2007). Under the assumption that there exists a domain-independent scoring function, these analyses show that the drop in accuracy is upper-bounded by the quantity called discrepancy distance. The discrepancy distance is dependent on the feature representation $\boldsymbol{z}$, and the input distributions for both domains $P_S(\boldsymbol{z})$ and $P_T(\boldsymbol{z})$, and is defined as

$$d_{\boldsymbol{z}}(S,T) = \max_{f,f'} |E_{P_S}[f(\boldsymbol{z}) \neq f'(\boldsymbol{z})] - E_{P_T}[f(\boldsymbol{z}) \neq f'(\boldsymbol{z})]|,$$

where f and f' are arbitrary linear classifiers in the feature representation $\boldsymbol{z}$. The quantity $E_P[f(\boldsymbol{z}) \neq f'(\boldsymbol{z})]$ measures the probability mass assigned to examples where f and f' disagree. Then the discrepancy distance is the maximal change in the size of this disagreement set due to transfer between the domains. For a more restricted class of classifiers which rely only on any single feature[2] z_i, the distance is equal to the maximum over the change in the distributions $P(z_i)$. Consequently, for arbitrary linear classifiers we have:

$$d_{\boldsymbol{z}}(S,T) \geq \max_{i=1,\ldots,m} |E_{P_S}[z_i = 1] - E_{P_T}[z_i = 1]|.$$

It follows that low inter-domain variability of the marginal distributions of latent variables is a necessary condition for low discrepancy distance. Minimizing the difference in the marginal distributions can be regarded as a coarse approximation to the minimization of the distance. However, we have to concede that the above argument is fairly informal, as the generalization bounds do not directly apply to our case: (1) our feature representation is learned from the same data as the classifier, (2) we cannot guarantee that the existence of a domain-independent scoring function is preserved under the learned transformation $\boldsymbol{x} \rightarrow \boldsymbol{z}$ and (3) in our setting we have access not only to samples from $P(\boldsymbol{z}|\boldsymbol{x}, \theta)$ but also to the distribution itself.

3.2 The Expectation Criterion

Though the above argument suggests a specific form of the regularizing term, we believe that the penalizer should not be very sensitive to small differences in the marginal distributions, as useful variables (clusters) are likely to have somewhat different marginal distributions in different domains, but it should severely penalize extreme differences.

To achieve this goal we instead propose to use the symmetrized Kullback-Leibler (KL) divergence between the marginal distributions as the penalty. The

[2] We consider only binary features here.

derivative of the symmetrized KL divergence is large when one of the marginal distributions is concentrated at 0 or 1 with another distribution still having high entropy, and therefore such configurations are severely penalized.[3] Formally, the regularizer $G(\theta)$ is defined as

$$G(\theta) = \sum_{i=1}^{m} D(P_S(z_i|\theta)||P_T(z_i|\theta)) + D(P_T(z_i|\theta)||P_S(z_i|\theta)), \quad (1)$$

where $P_S(z_i)$ and $P_T(z_i)$ stand for the training sample estimates of the marginal distributions of latent features, for instance:

$$P_T(z_i = 1|\theta) = \frac{1}{|T_U|} \sum_{l \in T_U} P(z_i = 1|\boldsymbol{x}^{(l)}, \theta).$$

We augment the multi-conditional log-likelihood $L(\theta, \alpha)$ with the weighted regularization term $G(\theta)$ to get the composite objective function:

$$L_R(\theta, \alpha, \beta) = L(\theta, \alpha) - \beta G(\theta), \quad \beta > 0.$$

Note that this regularization term can be regarded as a form of the generalized expectation (GE) criteria (Mann and McCallum, 2010), where GE criteria are normally defined as KL divergences between a prior expectation of some feature and the expectation of this feature given by the model, where the prior expectation is provided by the model designer as a form of weak supervision. In our case, both expectations are provided by the model but on different domains.

Note that the proposed regularizer can be trivially extended to support the multi-domain case (Mansour et al., 2008) by considering symmetrized KL divergences for every pair of domains or regularizing the distributions for every domain towards their average.

More powerful regularization terms can also be motivated by minimization of the discrepancy distance but their optimization is likely to be expensive, whereas $L_R(\theta, \alpha, \beta)$ can be optimized efficiently.

[3] An alternative is to use the Jensen-Shannon (JS) divergence, however, our preliminary experiments seem to suggest that the symmetrized KL divergence is preferable. Though the two divergences are virtually equivalent when the distributions are very similar (their ratio tends to a constant as the distributions go closer), the symmetrized KL divergence stronger penalizes extreme differences and this is important for our purposes.

4 Learning and Inference

In this section we describe an approximate learning algorithm based on the mean-field approximation. Though we believe that our approach is independent of the specific learning algorithm, we provide the description for completeness. We also describe a simple approximate algorithm for computing $P(y|\boldsymbol{x}, \theta)$ at test time.

The stochastic gradient descent algorithm iterates over examples and updates the weight vector based on the contribution of every considered example to the objective function $L_R(\theta, \alpha, \beta)$. To compute these updates we need to approximate gradients of $\nabla_\theta \log P(y^{(l)}|\boldsymbol{x}^{(l)}, \theta)$ ($l \in S_L$), $\nabla_\theta \log P(\boldsymbol{x}^{(l)}|\theta)$ ($l \in S_L \cup S_U \cup T_U$) as well as to estimate the contribution of a given example to the gradient of the regularizer $\nabla_\theta G(\theta)$. In the next sections we will describe how each of these terms can be estimated.

4.1 Conditional Likelihood Term

We start by explaining the mean-field approximation of $\log P(y|\boldsymbol{x}, \theta)$. First, we compute the means $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_m)$:

$$\mu_i = P(z_i = 1|\boldsymbol{x}, \boldsymbol{v}) = \quad \sigma(v_{0i} + \textstyle\sum_{j=1}^{|\boldsymbol{x}|} v_{x_j i}).$$

Now we can substitute them instead of $\boldsymbol{z}$ to approximate the conditional probability of the label:

$$\begin{aligned} P(y = 1|\boldsymbol{x}, \theta) \quad &= \textstyle\sum_{\boldsymbol{z}} P(y|\boldsymbol{z}, \boldsymbol{w}) P(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{v}) \\ &\propto \sigma(w_0 + \textstyle\sum_{i=1}^{m} w_i \mu_i). \end{aligned}$$

We use this estimate both at testing time and also to compute gradients $\nabla_\theta \log P(y^{(l)}|\boldsymbol{x}^{(l)}, \theta)$ during learning. The gradients can be computed efficiently using a form of back-propagation. Note that with this approximation, we do not need to normalize over the feature space, which makes the model very efficient at classification time.

This approximation is equivalent to the computation of the two-layer perceptron with the soft-max activation function (Bishop, 1995). However, the above derivation provides a probabilistic interpretation of the hidden layer.

4.2 Unlabeled Likelihood Term

In this section, we describe how the unlabeled likelihood term is optimized in our stochastic learning

algorithm. First, we note that, given the directed nature of the arcs between $\boldsymbol{z}$ and y, the weights $\boldsymbol{w}$ do not affect the probability of input $\boldsymbol{x}$, that is $P(\boldsymbol{x}|\theta) = P(\boldsymbol{x}|\boldsymbol{v})$.

Instead of directly approximating the gradient $\nabla_{\boldsymbol{v}} \log P(\boldsymbol{x}^{(l)}|\boldsymbol{v})$, we use a deterministic version of the Contrastive Divergence (CD) algorithm, equivalent to the mean-field approximation of the reconstruction error used in training autoassociaters (Bengio and Delalleau, 2007). The CD-based estimators are biased estimators but are guaranteed to converge. Intuitively, maximizing the likelihood of unlabeled data is closely related to minimizing the reconstruction error, that is training a model to discover such mapping parameters $\boldsymbol{u}$ that $\boldsymbol{z}$ encodes all the necessary information to accurately reproduce $\boldsymbol{x}^{(l)}$ from $\boldsymbol{z}$ for every training example $\boldsymbol{x}^{(l)}$. Formally, the mean-field approximation to the negated reconstruction error is defined as

$$\hat{L}(\boldsymbol{x}^{(l)}, \boldsymbol{v}) = \log P(\boldsymbol{x}^{(l)}|\boldsymbol{\mu}, \boldsymbol{v}),$$

where the means, $\mu_i = P(z_i = 1|\boldsymbol{x}^{(l)}, \boldsymbol{v})$, are computed as in the preceding section. Note that when computing the gradient of $\nabla_{\boldsymbol{v}}\hat{L}$, we need to take into account both the forward and backward mappings: the computation of the means $\boldsymbol{\mu}$ from $\boldsymbol{x}^{(\boldsymbol{l})}$ and the computation of the log-probability of $\boldsymbol{x}^{(\boldsymbol{l})}$ given the means $\boldsymbol{\mu}$:

$$\frac{d\hat{L}}{dv_{ki}} = \frac{\partial \hat{L}}{\partial v_{ki}} + \frac{\partial \hat{L}}{\partial \mu_i}\frac{d\mu_i}{dv_{ki}}.$$

4.3 Regularization Term

The criterion $G(\theta)$ is also independent of the classifier parameters $\boldsymbol{w}$, i.e. $G(\theta) = G(\boldsymbol{v})$, and our goal is to compute the contribution of a considered example l to the gradient $\nabla_{\boldsymbol{v}} G(\boldsymbol{v})$.

The regularizer $G(\boldsymbol{v})$ is defined as in equation (1) and it is a function of the sample-based domain-specific marginal distributions of latent variables P_S and P_T:

$$P_T(z_i = 1|\theta) = \frac{1}{|T_U|}\sum_{l \in T_U} \mu_i^{(l)},$$

where the means $\mu_i^{(l)} = P(z_i = 1|x^{(l)}, \boldsymbol{v})$; P_S can be re-written analogously. $G(\boldsymbol{v})$ is dependent on the parameters $\boldsymbol{v}$ only via the mean activations of the latent variables $\boldsymbol{\mu}^{(l)}$, and contribution of each example l can be computed by straightforward differentiation:

$$\frac{dG^{(l)}(\boldsymbol{v})}{dv_{ki}} = (\log\frac{p}{p'} - \log\frac{1-p}{1-p'} - \frac{p'}{p} + \frac{1-p'}{1-p})\frac{d\mu_i^{(l)}}{dv_{ki}},$$

where $p = P_S(z_i = 1|\theta)$ and $p' = P_T(z_i = 1|\theta)$ if l is from the source domain, and, inversely, $p = P_T(z_i = 1|\theta)$ and $p' = P_S(z_i = 1|\theta)$, otherwise.

One problem with the above expression is that the exact computation of P_S and P_T requires recomputation of the means $\boldsymbol{\mu}^{(l)}$ for all the examples after each update of the parameters, resulting in $O(|S_L \cup S_U \cup T_U|^2)$ complexity of each iteration of stochastic gradient descent. Instead, we shuffle examples and use amortization; we approximate P_S at update t by:

$$\hat{P}_S^{(t)}(z_i = 1) = \begin{cases} (1-\gamma)\hat{P}_S^{(t-1)}(z_i = 1) + \gamma\mu_i^{(l)}, & l \in S_L \cup S_U \\ \hat{P}_S^{(t-1)}(z_i = 1), & \text{otherwise,} \end{cases}$$

where l is an example considered at update t. The approximation $\hat{P}_T$ is computed analogously.

5 Empirical Evaluation

In this section we empirically evaluate our approach on the sentiment classification task. We start with the description of the experimental set-up and the baselines, then we present the results and discuss the utility of the constraint on inter-domain variability.

5.1 Experimental setting

To evaluate our approach, we consider the same dataset as the one used to evaluate the SCL method (Blitzer et al., 2007). The dataset is composed of labeled and unlabeled reviews of four different product types: books, DVDs, electronics and kitchen appliances. For each domain, the dataset contains 1,000 labeled positive reviews and 1,000 labeled negative reviews, as well as several thousands of unlabeled examples (4,919 reviews per domain in average: ranging from 3,685 for DVDs to 5,945 for kitchen appliances). As in Blitzer et al. (2007), we randomly split each labelled portion into 1,600 examples for training and 400 examples for testing.

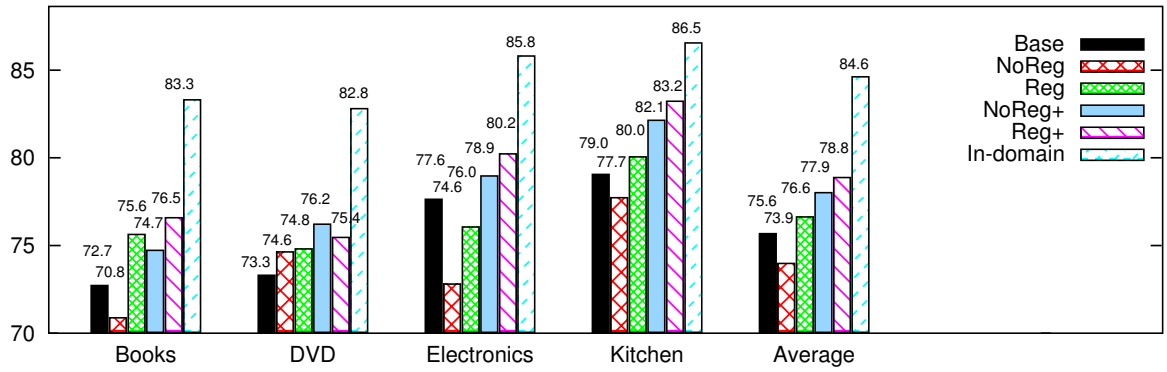


Figure 2: Averages accuracies when transferring to books, DVD, electronics and kitchen appliances domains, and average accuracy over all 12 domain pairs.

We evaluate the performance of our domain-adaptation approach on every ordered pair of domains. For every pair, the semi-supervised methods use labeled data from the source domain and unlabeled data from both domains. We compare them with two supervised methods: a supervised model (*Base*) which is trained on the source domain data only, and another supervised model (*In-domain*) which is learned on the labeled data from the target domain. The Base model can be regarded as a natural baseline model, whereas the In-domain model is essentially an upper-bound for any domain-adaptation method. All the methods, supervised and semi-supervised, are based on the model described in Section 2.

Instead of using the full set of bigram and unigram counts as features (Blitzer et al., 2007), we use a frequency cut-off of 30 to remove infrequent ngrams. This does not seem to have an adverse effect on the accuracy but makes learning very efficient: the average training time for the semi-supervised methods was about 20 minutes on a standard PC.

We coarsely tuned the parameters of the learning methods using a form of cross-validation. Both the parameter of the multi-conditional objective α (see Section 2) and the weighting for the constraint β (see Section 3.2) were set to 5. We used 25 iterations of stochastic gradient descent. The initial learning rate and the weight decay (the inverse squared variance of the Gaussian prior) were set to 0.01, and both parameters were reduced by the factor of 2 every iteration the objective function estimate went down. The size of the latent representation was equal to 10. The stochastic weight updates were amortized with the momentum (γ) of 0.99.

We trained the model both without regularization of the domain variability (*NoReg*, $\beta = 0$), and with the regularizing term (*Reg*). For the SCL method to produce an accurate classifier for the target domain it is necessary to train a classifier using both the induced shared representation and the initial non-transformed representation. In our case, due to joint learning and non-convexity of the learning problem, this approach would be problematic.[4] Instead, we combine predictions of the semi-supervised models *Reg* and *NoReg* with the baseline out-of-domain model (*Base*) using the product-of-experts combination (Hinton, 2002), the corresponding methods are called *Reg+* and *NoReg+*, respectively.

In all our models, we augmented the vector z with an additional component set to 0 for examples in the source domain and to 1 for the target domain examples. In this way, we essentially subtracted a unigram domain-specific model from our latent variable model in the hope that this will further reduce the domain dependence of the rest of the model parameters. In preliminary experiments, this modification was beneficial for all the models including the non-constrained one (*NoReg*).

5.2 Results and Discussion

The results of all the methods are presented in Figure 2. The 4 leftmost groups of results correspond to a single target domain, and therefore each of

[4]The latent variables are not likely to learn any useful mapping in the presence of observable features. Special training regimes may be used to attempt to circumvent this problem.

them is an average over experiments on 3 domain-pairs, for instance, the group Books represents an average over adaptation experiments DVDs→books, electronics→books, kitchen→books. The rightmost group of the results corresponds to the average over all 12 experiments. First, observe that the total drop in the accuracy when moving to the target domain is 8.9%: from 84.6% demonstrated by the In-domain classifier to 75.6% shown by the non-adapted Base classifier. For convenience, we also present the errors due to transfer in a separate Table 1: our best method (*Reg+*) achieves 35% relative reduction of this loss, decreasing the gap to 5.7%.

Now, let us turn to the question of the utility of the constraints. First, observe that the non-regularized version of the model (*NoReg*) often fails to outperform the baseline and achieves the scores considerably worse than the results of the regularized version (2.6% absolute difference). We believe that this happens because the clusters induced when optimizing the non-regularized learning objective are often domain-specific. The regularized model demonstrates substantially better results slightly beating the baseline in most cases. Still, to achieve a larger decrease of the domain-adaptation error, it was necessary to use the combined models, *Reg+* and *NoReg+*. Here, again, the regularized model substantially outperforms the non-regularized one (35% against 26% relative error reduction for *Reg+* and *NoReg+*, respectively).

In Table 1, we also compare the results of our method with the results of the best version of the SCL method (SCL-MI) reported in Blitzer et al. (2007). The average error reductions for our method *Reg+* and for the SCL method are virtually equal. However, formally, these two numbers are not directly comparable. First, the random splits are different, though this is unlikely to result in any significant difference, as the split proportions are the same and the test sets are sufficiently large. Second, the absolute scores achieved in Blitzer et al. (2007) are slightly worse than those demonstrated in our experiments both for supervised and semi-supervised methods. In absolute terms, our *Reg+* method outperforms the SCL method by more than 1%: 75.6% against 74.5%, in average. This is probably due to the difference in the used learning methods: optimization of the Huber loss vs. our latent variable model.[5] This comparison suggests that our domain-adaptation method is a viable alternative to SCL.

D	Base	NoReg	Reg	NoReg+	Reg+	SCL-MI
B	10.6	12.4	7.7	8.6	6.7	**5.8**
D	9.5	8.2	8.0	6.6	7.3	**6.1**
E	8.2	13.0	9.7	6.8	**5.5**	**5.5**
K	7.5	8.8	6.5	4.4	**3.3**	5.6
Av	8.9	10.6	8.0	6.6	**5.7**	5.8

Table 1: Drop in the accuracy score due to the transfer for the 4 domains: (B)ooks, (D)VD, (E)electronics and (K)itchen appliances, and in average over the domains.

Also, it is important to point out that the SCL method uses auxiliary tasks to induce the shared feature representation, these tasks are constructed on the basis of unlabeled data. The auxiliary tasks and the original problem should be closely related, namely they should have the same (or similar) set of predictive features. Defining such tasks can be a challenging engineering problem. On the sentiment classification task in order to construct them two steps need to be performed: (1) a set of words correlated with the sentiment label is selected, and, then (2) prediction of each such word is regarded a distinct auxiliary problem. For many other domains (e.g., parsing (Plank, 2009)) the construction of an effective set of auxiliary tasks is still an open problem.

6 Related Work

There is a growing body of work on domain adaptation. In this paper, we focus on the class of methods which induce a shared feature representation. Another popular class of domain-adaptation techniques assume that the input distributions $P(x)$ for the source and the target domain share support, that is every example x which has a non-zero probability on the target domain must have also a non-zero probability on the source domain, and vice-versa. Such methods tackle domain adaptation by instance re-weighting (Bickel et al., 2007; Jiang and Zhai, 2007), or, similarly, by feature re-weighting (Satpal and Sarawagi, 2007). In NLP, most features

[5]The drop in accuracy for the SCL method in Table 1 is is computed with respect to the less accurate supervised in-domain classifier considered in Blitzer et al. (2007), otherwise, the computed drop would be larger.

are word-based and lexicons are very different for different domains, therefore such assumptions are likely to be overly restrictive.

Various semi-supervised techniques for domain-adaptation have also been considered, one example being self-training (McClosky et al., 2006). However, their behavior in the domain-adaptation setting is not well-understood. Semi-supervised learning with distributed representations and its application to domain adaptation has previously been considered in (Huang and Yates, 2009), but no attempt has been made to address problems specific to the domain-adaptation setting. Similar approaches has also been considered in the context of topic models (Xue et al., 2008), however the preference towards induction of domain-independent topics was not explicitly encoded in the learning objective or model priors.

A closely related method to ours is that of (Druck and McCallum, 2010) which performs semi-supervised learning with posterior regularization (Ganchev et al., 2010). Our approach differs from theirs in many respects. First, they do not focus on the domain-adaptation setting and do not attempt to define constraints to prevent the model from learning domain-specific information. Second, their expectation constraints are estimated from labeled data, whereas we are trying to match expectations computed on unlabeled data for two domains.

This approach bears some similarity to the adaptation methods standard for the setting where labelled data is available for both domains (Chelba and Acero, 2004; Daumé and Marcu, 2006). However, instead of ensuring that the classifier parameters are similar across domains, we favor models resulting in similar marginal distributions of latent variables.

7 Discussion and Conclusions

In this paper we presented a domain-adaptation method based on semi-supervised learning with distributed representations coupled with constraints favoring domain-independence of modeled phenomena. Our approach results in competitive domain-adaptation performance on the sentiment classification task, rivalling that of the state-of-the-art SCL method (Blitzer et al., 2007). Both of these methods induce a shared feature representation but unlike SCL our method does not require construction of any auxiliary tasks in order to induce this representation. The primary area of the future work is to apply our method to structured prediction problems in NLP, such as syntactic parsing or semantic role labeling, where construction of auxiliary tasks proved problematic. Another direction is to favor domain-invariability not only of the expectations of individual variables but rather those of constraint functions involving latent variables, features and labels.

Acknowledgements

The author acknowledges the support of the Cluster of Excellence on Multimodal Computing and Interaction at Saarland University and thanks the anonymous reviewers for their helpful comments and suggestions.

References

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine Learning*, 79:151–175.

Yoshua Bengio and Olivier Delalleau. 2007. Justifying and generalizing contrastive divergence. Technical Report TR 1311, Department IRO, University of Montreal, November.

S. Bickel, M. Brüeckner, and T. Scheffer. 2007. Discriminative learning for differing training and test distributions. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 81–88.

Christopher M. Bishop. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of EMNLP*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proc. 45th Meeting of Association for Computational Linguistics (ACL)*, Prague, Czech Republic.

John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. 2008. Learning bounds for domain adaptation. In *Proc. Advances In Neural Information Processing Systems (NIPS '07)*.

Ciprian Chelba and Alex Acero. 2004. Adaptation of maximum entropy capitalizer: Little data can help a lot. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 285–292.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning, ICML*.

Hal Daumé and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence*, 26:101–126.

Gregory Druck and Andrew McCallum. 2010. High-performance semi-supervised learning using discriminatively constrained generative models. In *Proc. of the International Conference on Machine Learning (ICML)*, Haifa, Israel.

Kuzman Ganchev, Joao Graca, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research (JMLR)*, pages 2001–2049.

Andrea Gesmundo, James Henderson, Paola Merlo, and Ivan Titov. 2009. Latent variable model of synchronous syntactic-semantic parsing for multiple languages. In *CoNLL 2009 Shared Task*.

Zoubin Ghahramani and Michael I. Jordan. 1997. Factorial hidden Markov models. *Machine Learning*, 29:245–273.

G. E. Hinton and R. R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507.

Geoffrey E. Hinton. 2002. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14:1771–1800.

Fei Huang and Alexander Yates. 2009. Distributional representations for handling sparsity in supervised sequence labeling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proc. of the Annual Meeting of the ACL*, pages 264–271, Prague, Czech Republic, June. Association for Computational Linguistics.

Gideon S. Mann and Andrew McCallum. 2010. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of Machine Learning Research*, 11:955–984.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. 2008. Domain adaptation with multiple sources. In *Advances in Neural Information Processing Systems*.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. 2009. Domain adaptation: Learning bounds and algorithms. In *Proceedings of The 22nd Annual Conference on Learning Theory (COLT 2009)*, Montreal, Canada.

Andrew McCallum, Chris Pal, Greg Druck, and Xuerui Wang. 2006. Multi-conditional learning: Generative/discriminative training for clustering and classification. In *AAAI*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *Proc. of the Annual Meeting of the ACL and the International Conference on Computational Linguistics*, Sydney, Australia.

B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Barbara Plank. 2009. Structural correspondence learning for parse disambiguation. In *Proceedings of the Student Research Workshop at EACL 2009*, pages 37–45, Athens, Greece, April. Association for Computational Linguistics.

Sandeepkumar Satpal and Sunita Sarawagi. 2007. Domain adaptation of conditional probability models via feature subsetting. In *Proceedings of 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, Warzaw, Poland.

Lawrence K. Saul, Tommi Jaakkola, and Michael I. Jordan. 1996. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4:61–76.

Paul Smolensky. 1986. Information processing in dynamical systems: foundations of harmony theory. In D. Rumehart and J McCelland, editors, *Parallel distributed processing: explorations in the microstructures of cognition*, volume 1 : Foundations, pages 194–281. MIT Press.

Ivan Titov and James Henderson. 2007a. Constituent parsing with Incremental Sigmoid Belief Networks. In *Proc. 45th Meeting of Association for Computational Linguistics (ACL)*, pages 632–639, Prague, Czech Republic.

Ivan Titov and James Henderson. 2007b. Fast and robust multilingual dependency parsing with a generative latent variable model. In *Proc. of the CoNLL shared task*, Prague, Czech Republic.

G.-R. Xue, W. Dai, Q. Yang, and Y. Yu. 2008. Topic-bridged PLSA for cross-domain text classification. In *Proceedings of the SIGIR Conference*.

Exact Decoding of Syntactic Translation Models through Lagrangian Relaxation

Alexander M. Rush
MIT CSAIL,
Cambridge, MA 02139, USA
srush@csail.mit.edu

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

Abstract

We describe an exact decoding algorithm for syntax-based statistical translation. The approach uses Lagrangian relaxation to decompose the decoding problem into tractable subproblems, thereby avoiding exhaustive dynamic programming. The method recovers exact solutions, with certificates of optimality, on over 97% of test examples; it has comparable speed to state-of-the-art decoders.

1 Introduction

Recent work has seen widespread use of synchronous probabilistic grammars in statistical machine translation (SMT). The decoding problem for a broad range of these systems (e.g., (Chiang, 2005; Marcu et al., 2006; Shen et al., 2008)) corresponds to the intersection of a (weighted) hypergraph with an n-gram language model.[1] The hypergraph represents a large set of possible translations, and is created by applying a synchronous grammar to the source language string. The language model is then used to rescore the translations in the hypergraph.

Decoding with these models is challenging, largely because of the cost of integrating an n-gram language model into the search process. Exact dynamic programming algorithms for the problem are well known (Bar-Hillel et al., 1964), but are too expensive to be used in practice.[2] Previous work on decoding for syntax-based SMT has therefore been focused primarily on approximate search methods.

This paper describes an efficient algorithm for exact decoding of synchronous grammar models for translation. We avoid the construction of (Bar-Hillel et al., 1964) by using *Lagrangian relaxation* to decompose the decoding problem into the following sub-problems:

1. Dynamic programming over the weighted hypergraph. This step does not require language model integration, and hence is highly efficient.

2. Application of an all-pairs shortest path algorithm to a directed graph derived from the weighted hypergraph. The size of the derived directed graph is linear in the size of the hypergraph, hence this step is again efficient.

Informally, the first decoding algorithm incorporates the weights and hard constraints on translations from the synchronous grammar, while the second decoding algorithm is used to integrate language model scores. Lagrange multipliers are used to enforce agreement between the structures produced by the two decoding algorithms.

In this paper we first give background on hypergraphs and the decoding problem. We then describe our decoding algorithm. The algorithm uses a subgradient method to minimize a dual function. The dual corresponds to a particular linear programming (LP) relaxation of the original decoding problem. The method will recover an exact solution, with a certificate of optimality, if the underlying LP relaxation has an integral solution. In some cases, however, the underlying LP will have a fractional solution, in which case the method will not be exact. The second technical contribution of this paper is to describe a method that iteratively tightens the underlying LP relaxation until an exact solution is produced. We do this by gradually introducing constraints to step 1 (dynamic programming over the hypergraph), while still maintaining efficiency.

[1] This problem is also relevant to other areas of statistical NLP, for example NL generation (Langkilde, 2000).

[2] E.g., with a trigram language model they run in $O(|E|w^6)$ time, where $|E|$ is the number of edges in the hypergraph, and w is the number of distinct lexical items in the hypergraph.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 72–82,
Portland, Oregon, June 19-24, 2011.

We report experiments using the tree-to-string model of (Huang and Mi, 2010). Our method gives exact solutions on over 97% of test examples. The method is comparable in speed to state-of-the-art decoding algorithms; for example, over 70% of the test examples are decoded in 2 seconds or less. We compare our method to cube pruning (Chiang, 2007), and find that our method gives improved model scores on a significant number of examples. One consequence of our work is that we give accurate estimates of the number of search errors for cube pruning.

2 Related Work

A variety of approximate decoding algorithms have been explored for syntax-based translation systems, including cube-pruning (Chiang, 2007; Huang and Chiang, 2007), left-to-right decoding with beam search (Watanabe et al., 2006; Huang and Mi, 2010), and coarse-to-fine methods (Petrov et al., 2008).

Recent work has developed decoding algorithms based on finite state transducers (FSTs). Iglesias et al. (2009) show that exact FST decoding is feasible for a phrase-based system with limited reordering (the MJ1 model (Kumar and Byrne, 2005)), and de Gispert et al. (2010) show that exact FST decoding is feasible for a specific class of hierarchical grammars (shallow-1 grammars). Approximate search methods are used for more complex reordering models or grammars. The FST algorithms are shown to produce higher scoring solutions than cube-pruning on a large proportion of examples.

Lagrangian relaxation is a classical technique in combinatorial optimization (Korte and Vygen, 2008). Lagrange multipliers are used to add linear constraints to an existing problem that can be solved using a combinatorial algorithm; the resulting dual function is then minimized, for example using subgradient methods. In recent work, *dual decomposition*—a special case of Lagrangian relaxation, where the linear constraints enforce agreement between two or more models—has been applied to inference in Markov random fields (Wainwright et al., 2005; Komodakis et al., 2007; Sontag et al., 2008), and also to inference problems in NLP (Rush et al., 2010; Koo et al., 2010). There are close connections between dual decomposition and work on belief propagation (Smith and Eisner, 2008).

3 Background: Hypergraphs

Translation with many syntax-based systems (e.g., (Chiang, 2005; Marcu et al., 2006; Shen et al., 2008; Huang and Mi, 2010)) can be implemented as a two-step process. The first step is to take an input sentence in the source language, and from this to create a hypergraph (sometimes called a translation forest) that represents the set of possible translations (strings in the target language) and derivations under the grammar. The second step is to integrate an n-gram language model with this hypergraph. For example, in the system of (Chiang, 2005), the hypergraph is created as follows: first, the source side of the synchronous grammar is used to create a parse forest over the source language string. Second, transduction operations derived from synchronous rules in the grammar are used to create the target-language hypergraph. Chiang's method uses a synchronous context-free grammar, but the hypergraph formalism is applicable to a broad range of other grammatical formalisms, for example dependency grammars (e.g., (Shen et al., 2008)).

A hypergraph is a pair (V, E) where $V = \{1, 2, \ldots, |V|\}$ is a set of vertices, and E is a set of hyperedges. A single distinguished vertex is taken as the root of the hypergraph; without loss of generality we take this vertex to be $v = 1$. Each hyperedge $e \in E$ is a tuple $\langle\langle v_1, v_2, \ldots, v_k\rangle, v_0\rangle$ where $v_0 \in V$, and $v_i \in \{2 \ldots |V|\}$ for $i = 1 \ldots k$. The vertex v_0 is referred to as the *head* of the edge. The ordered sequence $\langle v_1, v_2, \ldots, v_k\rangle$ is referred to as the *tail* of the edge; in addition, we sometimes refer to $v_1, v_2, \ldots v_k$ as the *children* in the edge. The number of children k may vary across different edges, but $k \geq 1$ for all edges (i.e., each edge has at least one child). We will use $h(e)$ to refer to the head of an edge e, and $t(e)$ to refer to the tail.

We will assume that the hypergraph is acyclic: intuitively this will mean that no derivation (as defined below) contains the same vertex more than once (see (Martin et al., 1990) for a formal definition).

Each vertex $v \in V$ is either a *non-terminal* in the hypergraph, or a *leaf*. The set of non-terminals is

$$V_N = \{v \in V : \exists e \in E \quad \text{such that} \quad h(e) = v\}$$

Conversely, the set of leaves is defined as

$$V_L = \{v \in V : \nexists e \in E \quad \text{such that} \quad h(e) = v\}$$

Finally, we assume that each $v \in V$ has a label $l(v)$. The labels for leaves will be *words*, and will be important in defining strings and language model scores for those strings. The labels for non-terminal nodes will not be important for results in this paper.[3]

We now turn to derivations. Define an *index set* $\mathcal{I} = V \cup E$. A derivation is represented by a vector $y = \{y_r : r \in \mathcal{I}\}$ where $y_v = 1$ if vertex v is used in the derivation, $y_v = 0$ otherwise (similarly $y_e = 1$ if edge e is used in the derivation, $y_e = 0$ otherwise). Thus y is a vector in $\{0,1\}^{|\mathcal{I}|}$. A valid derivation satisfies the following constraints:

- $y_1 = 1$ (the root must be in the derivation).
- For all $v \in V_N$, $y_v = \sum_{e:h(e)=v} y_e$.
- For all $v \in 2 \ldots |V|$, $y_v = \sum_{e:v \in t(e)} y_e$.

We use $\mathcal{Y}$ to refer to the set of valid derivations. The set $\mathcal{Y}$ is a subset of $\{0,1\}^{|\mathcal{I}|}$ (not all members of $\{0,1\}^{|\mathcal{I}|}$ will correspond to valid derivations).

Each derivation y in the hypergraph will imply an ordered sequence of leaves $v_1 \ldots v_n$. We use $s(y)$ to refer to this sequence. The *sentence* associated with the derivation is then $l(v_1) \ldots l(v_n)$.

In a weighted hypergraph problem, we assume a parameter vector $\theta = \{\theta_r : r \in \mathcal{I}\}$. The score for any derivation is $f(y) = \theta \cdot y = \sum_{r \in \mathcal{I}} \theta_r y_r$. Simple bottom-up dynamic programming—essentially the CKY algorithm—can be used to find $y^* = \arg\max_{y \in \mathcal{Y}} f(y)$ under these definitions.

The focus of this paper will be to solve problems involving the integration of a k'th order language model with a hypergraph. In these problems, the score for a derivation is modified to be

$$f(y) = \sum_{r \in \mathcal{I}} \theta_r y_r + \sum_{i=k}^{n} \theta(v_{i-k+1}, v_{i-k+2}, \ldots, v_i) \quad (1)$$

where $v_1 \ldots v_n = s(y)$. The $\theta(v_{i-k+1}, \ldots, v_i)$ parameters score n-grams of length k. These parameters are typically defined by a language model, for example with $k = 3$ we would have $\theta(v_{i-2}, v_{i-1}, v_i) = \log p(l(v_i)|l(v_{i-2}), l(v_{i-1}))$. The problem is then to find $y^* = \arg\max_{y \in \mathcal{Y}} f(y)$ under this definition.

Throughout this paper we make the following assumption when using a bigram language model:

[3]They might for example be non-terminal symbols from the grammar used to generate the hypergraph.

Assumption 3.1 *(*Bigram start/end assumption.*)* *For any derivation y, with leaves $s(y) = v_1, v_2, \ldots, v_n$, it is the case that: (1) $v_1 = 2$ and $v_n = 3$; (2) the leaves 2 and 3 cannot appear at any other position in the strings $s(y)$ for $y \in \mathcal{Y}$; (3) $l(2) =$ <s> where <s> is the start symbol in the language model; (4) $l(3) =$ </s> where </s> is the end symbol.*

This assumption allows us to incorporate language model terms that depend on the start and end symbols. It also allows a clean solution for boundary conditions (the start/end of strings).[4]

4 A Simple Lagrangian Relaxation Algorithm

We now give a Lagrangian relaxation algorithm for integration of a hypergraph with a bigram language model, in cases where the hypergraph satisfies the following simplifying assumption:

Assumption 4.1 *(*The strict ordering assumption.*)* *For any two leaves v and w, it is either the case that: 1) for all derivations y such that v and w are both in the sequence $l(y)$, v precedes w; or 2) for all derivations y such that v and w are both in $l(y)$, w precedes v.*

Thus under this assumption, the relative ordering of any two leaves is fixed. This assumption is overly restrictive:[5] the next section describes an algorithm that does not require this assumption. However deriving the simple algorithm will be useful in developing intuition, and will lead directly to the algorithm for the unrestricted case.

4.1 A Sketch of the Algorithm

At a high level, the algorithm is as follows. We introduce Lagrange multipliers $u(v)$ for all $v \in V_L$, with initial values set to zero. The algorithm then involves the following steps: (1) For each leaf v, find the previous leaf w that maximizes the score $\theta(w, v) - u(w)$ (call this leaf $\alpha^*(v)$, and define $\alpha_v = \theta(\alpha^*(v), v) - u(\alpha^*(v))$). (2) find the highest scoring derivation using dynamic programming

[4]The assumption generalizes in the obvious way to k'th order language models: e.g., for trigram models we assume that $v_1 = 2$, $v_2 = 3$, $v_n = 4$, $l(2) = l(3) =$ <s>, $l(4) =$ </s>.

[5]It is easy to come up with examples that violate this assumption: for example a hypergraph with edges $\langle\langle 4, 5\rangle, 1\rangle$ and $\langle\langle 5, 4\rangle, 1\rangle$ violates the assumption. The hypergraphs found in translation frequently contain alternative orderings such as this.

over the original (non-intersected) hypergraph, with leaf nodes having weights $\theta_v + \alpha_v + u(v)$. (3) If the output derivation from step 2 has the same set of bigrams as those from step 1, then we have an exact solution to the problem. Otherwise, the Lagrange multipliers $u(v)$ are modified in a way that encourages agreement of the two steps, and we return to step 1.

Steps 1 and 2 can be performed efficiently; in particular, we avoid the classical dynamic programming intersection, instead relying on dynamic programming over the original, simple hypergraph.

4.2 A Formal Description

We now give a formal description of the algorithm. Define $\mathcal{B} \subseteq V_L \times V_L$ to be the set of all ordered pairs $\langle v, w \rangle$ such that there is at least one derivation y with v directly preceding w in $s(y)$. Extend the bit-vector y to include variables $y(v, w)$ for $\langle v, w \rangle \in \mathcal{B}$ where $y(v, w) = 1$ if leaf v is followed by w in $s(y)$, 0 otherwise. We redefine the index set to be $\mathcal{I} = V \cup E \cup \mathcal{B}$, and define $\mathcal{Y} \subseteq \{0, 1\}^{|\mathcal{I}|}$ to be the set of all possible derivations. Under assumptions 3.1 and 4.1 above, $\mathcal{Y} = \{y : y \text{ satisfies constraints } \mathbf{C0}, \mathbf{C1}, \mathbf{C2}\}$ where the constraint definitions are:

- **(C0)** The y_v and y_e variables form a derivation in the hypergraph, as defined in section 3.
- **(C1)** For all $v \in V_L$ such that $v \neq 2$, $y_v = \sum_{w:\langle w,v \rangle \in \mathcal{B}} y(w, v)$.
- **(C2)** For all $v \in V_L$ such that $v \neq 3$, $y_v = \sum_{w:\langle v,w \rangle \in \mathcal{B}} y(v, w)$.

C1 states that each leaf in a derivation has exactly one in-coming bigram, and that each leaf not in the derivation has 0 incoming bigrams; **C2** states that each leaf in a derivation has exactly one out-going bigram, and that each leaf not in the derivation has 0 outgoing bigrams.[6]

The score of a derivation is now $f(y) = \theta \cdot y$, i.e.,

$$f(y) = \sum_v \theta_v y_v + \sum_e \theta_e y_e + \sum_{\langle v,w \rangle \in \mathcal{B}} \theta(v, w) y(v, w)$$

where $\theta(v, w)$ are scores from the language model. Our goal is to compute $y^* = \arg\max_{y \in \mathcal{Y}} f(y)$.

[6] Recall that according to the bigram start/end assumption the leaves 2/3 are reserved for the start/end of the sequence $s(y)$, and hence do not have an incoming/outgoing bigram.

Initialization: Set $u^0(v) = 0$ for all $v \in V_L$
Algorithm: For $t = 1 \ldots T$:

- $y^t = \arg\max_{y \in \mathcal{Y}'} L(u^{t-1}, y)$
- **If** y^t satisfies constraints **C2**, **return** y^t,
  Else $\forall v \in V_L$, $u^t(v) = u^{t-1}(v) - \delta^t \left(y^t(v) - \sum_{w:\langle v,w \rangle \in \mathcal{B}} y^t(v, w) \right)$.

Figure 1: A simple Lagrangian relaxation algorithm. $\delta^t > 0$ is the step size at iteration t.

Next, define $\mathcal{Y}'$ as

$$\mathcal{Y}' = \{y : y \text{ satisfies constraints } \mathbf{C0} \text{ and } \mathbf{C1}\}$$

In this definition we have dropped the **C2** constraints. To incorporate these constraints, we use Lagrangian relaxation, with one Lagrange multiplier $u(v)$ for each constraint in **C2**. The Lagrangian is

$$\begin{aligned} L(u, y) &= f(y) + \sum_v u(v)(y(v) - \sum_{w:\langle v,w \rangle \in \mathcal{B}} y(v, w)) \\ &= \beta \cdot y \end{aligned}$$

where $\beta_v = \theta_v + u(v)$, $\beta_e = \theta_e$, and $\beta(v, w) = \theta(v, w) - u(v)$.

The dual problem is to find $\min_u L(u)$ where

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

Figure 1 shows a *subgradient* method for solving this problem. At each point the algorithm finds $y^t = \arg\max_{y \in \mathcal{Y}'} L(u^{t-1}, y)$, where u^{t-1} are the Lagrange multipliers from the previous iteration. If y^t satisfies the **C2** constraints in addition to **C0** and **C1**, then it is returned as the output from the algorithm. Otherwise, the multipliers $u(v)$ are updated. Intuitively, these updates encourage the values of y_v and $\sum_{w:\langle v,w \rangle \in \mathcal{B}} y(v, w)$ to be equal; formally, these updates correspond to subgradient steps.

The main computational step at each iteration is to compute $\arg\max_{y \in \mathcal{Y}'} L(u^{t-1}, y)$ This step is easily solved, as follows (we again use β_v, β_e and $\beta(v_1, v_2)$ to refer to the parameter values that incorporate Lagrange multipliers):

- For all $v \in V_L$, define $\alpha^*(v) = \arg\max_{w:\langle w,v \rangle \in \mathcal{B}} \beta(w, v)$ and $\alpha_v = \beta(\alpha^*(v), v)$. For all $v \in V_N$ define $\alpha_v = 0$.

- Using dynamic programming, find values for the y_v and y_e variables that form a valid derivation, and that maximize $f'(y) = \sum_v (\beta_v + \alpha_v) y_v + \sum_e \beta_e y_e$.
- Set $y(v, w) = 1$ iff $y(w) = 1$ and $\alpha^*(w) = v$.

The critical point here is that through our definition of $\mathcal{Y}'$, which ignores the **C2** constraints, we are able to do efficient search as just described. In the first step we compute the highest scoring incoming bigram for each leaf v. In the second step we use conventional dynamic programming over the hypergraph to find an optimal derivation that incorporates weights from the first step. Finally, we fill in the $y(v, w)$ values. Each iteration of the algorithm runs in $O(|E| + |\mathcal{B}|)$ time.

There are close connections between Lagrangian relaxation and linear programming relaxations. The most important formal results are: 1) for any value of u, $L(u) \geq f(y^*)$ (hence the dual value provides an upper bound on the optimal primal value); 2) under an appropriate choice of the step sizes δ^t, the subgradient algorithm is guaranteed to converge to the minimum of $L(u)$ (i.e., we will minimize the upper bound, making it as tight as possible); 3) if at any point the algorithm in figure 1 finds a y^t that satisfies the **C2** constraints, then this is guaranteed to be the optimal primal solution.

Unfortunately, this algorithm may fail to produce a good solution for hypergraphs where the strict ordering constraint does not hold. In this case it is possible to find derivations y that satisfy constraints **C0**, **C1**, **C2**, but which are invalid. As one example, consider a derivation with $s(y) = 2, 4, 5, 3$ and $y(2, 3) = y(4, 5) = y(5, 4) = 1$. The constraints are all satisfied in this case, but the bigram variables are invalid (e.g., they contain a cycle).

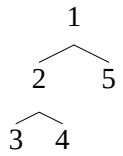
5 The Full Algorithm

We now describe our full algorithm, which does not require the strict ordering constraint. In addition, the full algorithm allows a trigram language model. We first give a sketch, and then give a formal definition.

5.1 A Sketch of the Algorithm

A crucial idea in the new algorithm is that of *paths* between leaves in hypergraph derivations. Previously, for each derivation y, we had defined $s(y) = v_1, v_2, \ldots, v_n$ to be the sequence of leaves in y. In addition, we will define $g(y) = p_0, v_1, p_1, v_2, p_2, v_3, p_3, \ldots, p_{n-1}, v_n, p_n$ where each p_i is a path in the derivation between leaves v_i and v_{i+1}. The path traces through the nonterminals that are between the two leaves in the tree.

As an example, consider the following derivation (with hyperedges $\langle\langle 2, 5\rangle, 1\rangle$ and $\langle\langle 3, 4\rangle, 2\rangle$):



For this example $g(y)$ is $\langle 1\ \downarrow, 2\ \downarrow\rangle\ \langle 2\ \downarrow, 3\ \downarrow\rangle\ \langle 3\ \downarrow\rangle$, 3, $\langle 3\ \uparrow\rangle\ \langle 3\ \uparrow, 4\ \downarrow\rangle\ \langle 4\ \downarrow\rangle$, 4, $\langle 4\ \uparrow\rangle\ \langle 4\ \uparrow, 2\ \uparrow\rangle\ \langle 2\ \uparrow, 5\ \downarrow\rangle\ \langle 5\ \downarrow\rangle$, 5, $\langle 5\ \uparrow\rangle\ \langle 5\ \uparrow, 1\ \uparrow\rangle$. States of the form $\langle a\ \downarrow\rangle$ and $\langle a\ \uparrow\rangle$ where a is a leaf appear in the paths respectively before/after the leaf a. States of the form $\langle a, b\rangle$ correspond to the steps taken in a top-down, left-to-right, traversal of the tree, where down and up arrows indicate whether a node is being visited for the first or second time (the traversal in this case would be $1, 2, 3, 4, 2, 5, 1$).

The mapping from a derivation y to a path $g(y)$ can be performed using the algorithm in figure 2. For a given derivation y, define $E(y) = \{y : y_e = 1\}$, and use $E(y)$ as the set of input edges to this algorithm. The output from the algorithm will be a set of states S, and a set of directed edges T, which together fully define the path $g(y)$.

In the simple algorithm, the first step was to predict the previous leaf for each leaf v, under a score that combined a language model score with a Lagrange multiplier score (i.e., compute $\arg\max_w \beta(w, v)$ where $\beta(w, v) = \theta(w, v) + u(w)$). In this section we describe an algorithm that for each leaf v again predicts the previous leaf, but in addition predicts the full *path* back to that leaf. For example, rather than making a prediction for leaf 5 that it should be preceded by leaf 4, we would also predict the path $\langle 4\ \uparrow\rangle\langle 4\ \uparrow, 2\ \uparrow\rangle\ \langle 2\ \uparrow, 5\ \downarrow\rangle\langle 5\ \downarrow\rangle$ between these two leaves. Lagrange multipliers will be used to enforce consistency between these predictions (both paths and previous words) and a valid derivation.

Input: A set E of hyperedges. **Output:** A directed graph S, T where S is a set of vertices, and T is a set of edges.
Step 1: Creating S: Define $S = \cup_{e \in E} S(e)$ where $S(e)$ is defined as follows. Assume $e = \langle \langle v_1, v_2, \ldots, v_k \rangle, v_0 \rangle$. Include the following states in $S(e)$: (1) $\langle v_0 \downarrow, v_1 \downarrow \rangle$ and $\langle v_k \uparrow, v_0 \uparrow \rangle$. (2) $\langle v_j \uparrow, v_{j+1} \downarrow \rangle$ for $j = 1 \ldots k-1$ (if $k = 1$ then there are no such states). (3) In addition, for any v_j for $j = 1 \ldots k$ such that $v_j \in V_L$, add the states $\langle v_j \downarrow \rangle$ and $\langle v_j \uparrow \rangle$.
Step 2: Creating T**:** T is formed by including the following directed arcs: (1) Add an arc from $\langle a, b \rangle \in S$ to $\langle c, d \rangle \in S$ whenever $b = c$. (2) Add an arc from $\langle a, b \downarrow \rangle \in S$ to $\langle c \downarrow \rangle \in S$ whenever $b = c$. (3) Add an arc from $\langle a \uparrow \rangle \in S$ to $\langle b \uparrow, c \rangle \in S$ whenever $a = b$.

Figure 2: Algorithm for constructing a directed graph (S, T) from a set of hyperedges E.

5.2 A Formal Description

We first use the algorithm in figure 2 with the entire set of hyperedges, E, as its input. The result is a directed graph (S, T) that contains *all possible paths* for valid derivations in V, E (it also contains additional, ill-formed paths). We then introduce the following definition:

Definition 5.1 A trigram path p *is* $p = \langle v_1, p_1, v_2, p_2, v_3 \rangle$ *where: a)* $v_1, v_2, v_3 \in V_L$*; b)* p_1 *is a path (sequence of states) between nodes* $\langle v_1 \uparrow \rangle$ *and* $\langle v_2 \downarrow \rangle$ *in the graph* (S, T)*; c)* p_2 *is a path between nodes* $\langle v_2 \uparrow \rangle$ *and* $\langle v_3 \downarrow \rangle$ *in the graph* (S, T)*. We define* $\mathcal{P}$ *to be the set of all trigram paths in* (S, T)*.*

The set $\mathcal{P}$ of trigram paths plays an analogous role to the set $\mathcal{B}$ of bigrams in our previous algorithm.

We use $v_1(p), p_1(p), v_2(p), p_2(p), v_3(p)$ to refer to the individual components of a path p. In addition, define S_N to be the set of states in S of the form $\langle a, b \rangle$ (as opposed to the form $\langle c \downarrow \rangle$ or $\langle c \uparrow \rangle$ where $c \in V_L$).

We now define a new index set, $\mathcal{I} = V \cup E \cup S_N \cup \mathcal{P}$, adding variables y_s for $s \in S_N$, and y_p for $p \in \mathcal{P}$. If we take $\mathcal{Y} \subset \{0, 1\}^{|\mathcal{I}|}$ to be the set of valid derivations, the optimization problem is to find $y^* = \arg\max_{y \in \mathcal{Y}} f(y)$, where $f(y) = \theta \cdot y$, that is,

$$f(y) = \sum_v \theta_v y_v + \sum_e \theta_e y_e + \sum_s \theta_s y_s + \sum_p \theta_p y_p$$

In particular, we might define $\theta_s = 0$ for all s, and $\theta_p = \log p(l(v_3(p)) | l(v_1(p)), l(v_2(p)))$ where $p(w_3 | w_1, w_2)$ is a trigram probability.

- **D0.** The y_v and y_e variables form a valid derivation in the original hypergraph.
- **D1.** For all $s \in S_N$, $y_s = \sum_{e : s \in S(e)} y_e$ (see figure 2 for the definition of $S(e)$).
- **D2.** For all $v \in V_L$, $y_v = \sum_{p : v_3(p) = v} y_p$
- **D3.** For all $v \in V_L$, $y_v = \sum_{p : v_2(p) = v} y_p$
- **D4.** For all $v \in V_L$, $y_v = \sum_{p : v_1(p) = v} y_p$
- **D5.** For all $s \in S_N$, $y_s = \sum_{p : s \in p_1(p)} y_p$
- **D6.** For all $s \in S_N$, $y_s = \sum_{p : s \in p_2(p)} y_p$
- Lagrangian with Lagrange multipliers for **D3**–**D6**:

$$\begin{aligned} L(y, \lambda, \gamma, u, v) = \theta \cdot y \\ + \textstyle\sum_v \lambda_v \left(y_v - \sum_{p : v_2(p) = v} y_p \right) \\ + \textstyle\sum_v \gamma_v \left(y_v - \sum_{p : v_1(p) = v} y_p \right) \\ + \textstyle\sum_s u_s \left(y_s - \sum_{p : s \in p_1(p)} y_p \right) \\ + \textstyle\sum_s v_s \left(y_s - \sum_{p : s \in p_2(p)} y_p \right). \end{aligned}$$

Figure 3: Constraints **D0**–**D6**, and the Lagrangian.

The set $\mathcal{P}$ is large (typically exponential in size): however, we will see that we do not need to represent the y_p variables explicitly. Instead we will be able to leverage the underlying structure of a path as a sequence of states.

The set of valid derivations is $\mathcal{Y} = \{y : y \text{ satisfies constraints } \mathbf{D0}\text{–}\mathbf{D6}\}$ where the constraints are shown in figure 3. **D1** simply states that $y_s = 1$ iff there is exactly one edge e in the derivation such that $s \in S(e)$. Constraints **D2**–**D4** enforce consistency between leaves in the trigram paths, and the y_v values. Constraints **D5** and **D6** enforce consistency between states seen in the paths, and the y_s values.

The Lagrangian relaxation algorithm is then derived in a similar way to before. Define

$$\mathcal{Y}' = \{y : y \text{ satisfies constraints } \mathbf{D0}\text{–}\mathbf{D2}\}$$

We have dropped the **D3**–**D6** constraints, but these will be introduced using Lagrange multipliers. The resulting Lagrangian is shown in figure 3, and can be written as $L(y, \lambda, \gamma, u, v) = \beta \cdot y$ where $\beta_v = \theta_v + \lambda_v + \gamma_v$, $\beta_s = \theta_s + u_s + v_s$, $\beta_p = \theta_p - \lambda(v_2(p)) - \gamma(v_1(p)) - \sum_{s \in p_1(p)} u(s) - \sum_{s \in p_2(p)} v(s)$.

The dual is $L(\lambda, \gamma, u, v) = \max_{y \in \mathcal{Y}'} L(y, \lambda, \gamma, u, v)$; figure 4 shows a subgradient method that minimizes this dual. The key step in the algorithm at each iteration is to compute

Initialization: Set $\lambda^0 = 0, \gamma^0 = 0, u^0 = 0, v^0 = 0$
Algorithm: For $t = 1 \ldots T$:

- $y^t = \arg\max_{y \in \mathcal{Y}'} L(y, \lambda^{t-1}, \gamma^{t-1}, u^{t-1}, v^{t-1})$
- If y^t satisfies the constraints **D3**–**D6**, return y^t, else:
  - $\forall v \in V_L, \lambda_v^t = \lambda_v^{t-1} - \delta^t(y_v^t - \sum_{p:v_2(p)=v} y_p^t)$
  - $\forall v \in V_L, \gamma_v^t = \gamma_v^{t-1} - \delta^t(y_v^t - \sum_{p:v_1(p)=v} y_p^t)$
  - $\forall s \in S_N, u_s^t = u_s^{t-1} - \delta^t(y_s^t - \sum_{p:s \in p_1(p)} y_p^t)$
  - $\forall s \in S_N, v_s^t = v_s^{t-1} - \delta^t(y_s^t - \sum_{p:s \in p_2(p)} y_p^t)$

Figure 4: The full Lagrangian relaxation algortihm. $\delta^t > 0$ is the step size at iteration t.

$\arg\max_{y \in \mathcal{Y}'} L(y, \lambda, \gamma, u, v) = \arg\max_{y \in \mathcal{Y}'} \beta \cdot y$ where β is defined above. Again, our definition of $\mathcal{Y}'$ allows this maximization to be performed efficiently, as follows:

1. For each $v \in V_L$, define $\alpha_v^* = \arg\max_{p:v_3(p)=v} \beta(p)$, and $\alpha_v = \beta(\alpha_v^*)$. (i.e., for each v, compute the highest scoring trigram path ending in v.)

2. Find values for the y_v, y_e and y_s variables that form a valid derivation, and that maximize $f'(y) = \sum_v (\beta_v + \alpha_v) y_v + \sum_e \beta_e y_e + \sum_s \beta_s y_s$

3. Set $y_p = 1$ iff $y_{v_3(p)} = 1$ and $p = \alpha^*_{v_3(p)}$.

The first step involves finding the highest scoring incoming trigram path for each leaf v. This step can be performed efficiently using the Floyd-Warshall all-pairs shortest path algorithm (Floyd, 1962) over the graph (S, T); the details are given in the appendix. The second step involves simple dynamic programming over the hypergraph (V, E) (it is simple to integrate the β_s terms into this algorithm). In the third step, the path variables y_p are filled in.

5.3 Properties

We now describe some important properties of the algorithm:

Efficiency. The main steps of the algorithm are: 1) construction of the graph (S, T); 2) at each iteration, dynamic programming over the hypergraph (V, E); 3) at each iteration, all-pairs shortest path algorithms over the graph (S, T). Each of these steps is vastly more efficient than computing an exact intersection of the hypergraph with a language model.

Exact solutions. By usual guarantees for Lagrangian relaxation, if at any point the algorithm returns a solution y^t that satisfies constraints **D3**–**D6**, then y^t exactly solves the problem in Eq. 1.

Upper bounds. At each point in the algorithm, $L(\lambda^t, \gamma^t, u^t, v^t)$ is an upper bound on the score of the optimal primal solution, $f(y^*)$. Upper bounds can be useful in evaluating the quality of primal solutions from either our algorithm or other methods such as cube pruning.

Simplicity of implementation. Construction of the (S, T) graph is straightforward. The other steps—hypergraph dynamic programming, and all-pairs shortest path—are widely known algorithms that are simple to implement.

6 Tightening the Relaxation

The algorithm that we have described minimizes the dual function $L(\lambda, \gamma, u, v)$. By usual results for Lagrangian relaxation (e.g., see (Korte and Vygen, 2008)), L is the dual function for a particular LP relaxation arising from the definition of $\mathcal{Y}'$ and the additional constaints **D3**–**D6**. In some cases the LP relaxation has an integral solution, in which case the algorithm will return an optimal solution y^t.[7] In other cases, when the LP relaxation has a fractional solution, the subgradient algorithm will still converge to the minimum of L, but the primal solutions y^t will move between a number of solutions.

We now describe a method that incrementally adds hard constraints to the set $\mathcal{Y}'$, until the method returns an exact solution. For a given $y \in \mathcal{Y}'$, for any v with $y_v = 1$, we can recover the previous two leaves (the trigram ending in v) from either the path variables y_p, or the hypergraph variables y_e. Specifically, define $v_{-1}(v, y)$ to be the leaf preceding v in the trigram path p with y_p = 1 and $v_3(p) = v$, and $v_{-2}(v, y)$ to be the leaf two positions before v in the trigram path p with $y_p = 1$ and $v_3(p) = v$. Similarly, define $v'_{-1}(v, y)$ and $v'_{-2}(v, y)$ to be the preceding two leaves under the y_e variables. If the method has not converged, these two trigram definitions may not be consistent. For a con-

[7] Provided that the algorithm is run for enough iterations for convergence.

sistent solution, we require $v_{-1}(v, y) = v'_{-1}(v, y)$ and $v_{-2}(v, y) = v'_{-2}(v, y)$ for all v with $y_v = 1$. Unfortunately, explicitly enforcing all of these constraints would require exhaustive dynamic programming over the hypergraph using the (Bar-Hillel et al., 1964) method, something we wish to avoid.

Instead, we enforce a weaker set of constraints, which require far less computation. Assume some function $\pi : V_L \rightarrow \{1, 2, \ldots q\}$ that partitions the set of leaves into q different partitions. Then we will add the following constraints to $\mathcal{Y}'$:

$$\begin{aligned} \pi(v_{-1}(v, y)) &= \pi(v'_{-1}(v, y)) \\ \pi(v_{-2}(v, y)) &= \pi(v'_{-2}(v, y)) \end{aligned}$$

for all v such that $y_v = 1$. Finding $\arg\max_{y \in \mathcal{Y}'} \theta \cdot y$ under this new definition of $\mathcal{Y}'$ can be performed using the construction of (Bar-Hillel et al., 1964), with q different lexical items (for brevity we omit the details). This is efficient if q is small.[8]

The remaining question concerns how to choose a partition π that is effective in tightening the relaxation. To do this we implement the following steps: 1) run the subgradient algorithm until L is close to convergence; 2) then run the subgradient algorithm for m further iterations, keeping track of all pairs of leaf nodes that violate the constraints (i.e., pairs $a = v_{-1}(v, y)/b = v'_{-1}(v, y)$ or $a = v_{-2}(v, y)/b = v'_{-2}(v, y)$ such that $a \neq b$); 3) use a graph coloring algorithm to find a small partition that places all pairs $\langle a, b \rangle$ into separate partitions; 4) continue running Lagrangian relaxation, with the new constraints added. We expand π at each iteration to take into account new pairs $\langle a, b \rangle$ that violate the constraints.

In related work, Sontag et al. (2008) describe a method for inference in Markov random fields where additional constraints are chosen to tighten an underlying relaxation. Other relevant work in NLP includes (Tromble and Eisner, 2006; Riedel and Clarke, 2006). Our use of partitions π is related to previous work on coarse-to-fine inference for machine translation (Petrov et al., 2008).

7 Experiments

We report experiments on translation from Chinese to English, using the tree-to-string model described in (Huang and Mi, 2010). We use an identical model, and identical development and test data, to that used by Huang and Mi.[9] The translation model is trained on 1.5M sentence pairs of Chinese-English data; a trigram language model is used. The development data is the newswire portion of the 2006 NIST MT evaluation test set (616 sentences). The test set is the newswire portion of the 2008 NIST MT evaluation test set (691 sentences).

Time	%age (LR)	%age (DP)	%age (ILP)	%age (LP)
0.5s	37.5	10.2	8.8	21.0
1.0s	57.0	11.6	13.9	31.1
2.0s	72.2	15.1	21.1	45.9
4.0s	82.5	20.7	30.7	63.7
8.0s	88.9	25.2	41.8	78.3
16.0s	94.4	33.3	54.6	88.9
32.0s	97.8	42.8	68.5	95.2
Median time	0.79s	77.5s	12.1s	2.4s

Figure 5: Results showing percentage of examples that are decoded in less than t seconds, for $t = 0.5, 1.0, 2.0, \ldots, 32.0$. LR = Lagrangian relaxation; DP = exhaustive dynamic programming; ILP = integer linear programming; LP = linear programming (LP does not recover an exact solution). The (I)LP experiments were carried out using Gurobi, a high-performance commercial-grade solver.

We ran the full algorithm with the tightening method described in section 6. We ran the method for a limit of 200 iterations, hence some examples may not terminate with an exact solution. Our method gives exact solutions on 598/616 development set sentences (97.1%), and 675/691 test set sentences (97.7%).

In cases where the method does not converge within 200 iterations, we can return the best primal solution y^t found by the algorithm during those iterations. We can also get an upper bound on the difference $f(y^*) - f(y^t)$ using $\min_t L(u_t)$ as an upper bound on $f(y^*)$. Of the examples that did not converge, the worst example had a bound that was 1.4% of $f(y^t)$ (more specifically, $f(y^t)$ was -24.74, and the upper bound on $f(y^*) - f(y^t)$ was 0.34).

Figure 5 gives information on decoding time for our method and two other exact decoding methods: integer linear programming (using constraints **D0**–**D6**), and exhaustive dynamic programming using the construction of (Bar-Hillel et al., 1964). Our

[8] In fact in our experiments we use the original hypergraph to compute admissible outside scores for an exact A* search algorithm for this problem. We have found the resulting search algorithm to be very efficient.

[9] We thank Liang Huang and Haitao Mi for providing us with their model and data.

method is clearly the most efficient, and is comparable in speed to state-of-the-art decoding algorithms.

We also compare our method to cube pruning (Chiang, 2007; Huang and Chiang, 2007). We reimplemented cube pruning in C++, to give a fair comparison to our method. Cube pruning has a parameter, b, dictating the maximum number of items stored at each chart entry. With $b = 50$, our decoder finds higher scoring solutions on 50.5% of all examples (349 examples), the cube-pruning method gets a strictly higher score on only 1 example (this was one of the examples that did not converge within 200 iterations). With $b = 500$, our decoder finds better solutions on 18.5% of the examples (128 cases), cube-pruning finds a better solution on 3 examples. The median decoding time for our method is 0.79 seconds; the median times for cube pruning with $b = 50$ and $b = 500$ are 0.06 and 1.2 seconds respectively.

Our results give a very good estimate of the percentage of search errors for cube pruning. A natural question is how large b must be before exact solutions are returned on almost all examples. Even at $b = 1000$, we find that our method gives a better solution on 95 test examples (13.7%).

Figure 5 also gives a speed comparison of our method to a linear programming (LP) solver that solves the LP relaxation defined by constraints **D0**–**D6**. We still see speed-ups, in spite of the fact that our method is solving a harder problem (it provides integral solutions). The Lagrangian relaxation method, when run without the tightening method of section 6, is solving a dual of the problem being solved by the LP solver. Hence we can measure how often the tightening procedure is absolutely necessary, by seeing how often the LP solver provides a fractional solution. We find that this is the case on 54.0% of the test examples: the tightening procedure is clearly important. Inspection of the tightening procedure shows that the number of partitions required (the parameter q) is generally quite small: 59% of examples that require tightening require $q \leq 6$; 97.2% require $q \leq 10$.

8 Conclusion

We have described a Lagrangian relaxation algorithm for exact decoding of syntactic translation models, and shown that it is significantly more efficient than other exact algorithms for decoding tree-to-string models. There are a number of possible ways to extend this work. Our experiments have focused on tree-to-string models, but the method should also apply to Hiero-style syntactic translation models (Chiang, 2007). Additionally, our experiments used a trigram language model, however the constraints in figure 3 generalize to higher-order language models. Finally, our algorithm recovers the 1-best translation for a given input sentence; it should be possible to extend the method to find k-best solutions.

A Computing the Optimal Trigram Paths

For each $v \in V_L$, define $\alpha_v = \max_{p:v_3(p)=v} \beta(p)$, where $\beta(p) = h(v_1(p), v_2(p), v_3(p)) - \lambda_1(v_1(p)) - \lambda_2(v_2(p)) - \sum_{s \in p_1(p)} u(s) - \sum_{s \in p_2(p)} v(s)$. Here h is a function that computes language model scores, and the other terms involve Lagrange mulipliers. Our task is to compute α_v^* for all $v \in V_L$.

It is straightforward to show that the S, T graph is *acyclic*. This will allow us to apply shortest path algorithms to the graph, even though the weights $u(s)$ and $v(s)$ can be positive or negative.

For any pair $v_1, v_2 \in V_L$, define $\mathcal{P}(v_1, v_2)$ to be the set of paths between $\langle v_1 \uparrow \rangle$ and $\langle v_2 \downarrow \rangle$ in the graph S, T. Each path p gets a score $score_u(p) = -\sum_{s \in p} u(s)$. Next, define $p_u^*(v_1, v_2) = \arg\max_{p \in \mathcal{P}(v_1, v_2)} score_u(p)$, and $score_u^*(v_1, v_2) = score_u(p^*)$. We assume similar definitions for $p_v^*(v_1, v_2)$ and $score_v^*(v_1, v_2)$. The p_u^* and $score_u^*$ values can be calculated using an all-pairs shortest path algorithm, with weights $u(s)$ on nodes in the graph. Similarly, p_v^* and $score_v^*$ can be computed using all-pairs shortest path with weights $v(s)$ on the nodes.

Having calculated these values, define $\mathcal{T}(v)$ for any leaf v to be the set of trigrams (x, y, v) such that: 1) $x, y \in V_L$; 2) there is a path from $\langle x \uparrow \rangle$ to $\langle y \downarrow \rangle$ and from $\langle y \uparrow \rangle$ to $\langle v \downarrow \rangle$ in the graph S, T. Then we can calculate

$$\alpha_v = \max_{(x,y,v) \in \mathcal{T}(v)} (h(x, y, v) - \lambda_1(x) - \lambda_2(y) + p_u^*(x, y) + p_v^*(y, v))$$

in $O(|\mathcal{T}(v)|)$ time, by brute force search through the set $\mathcal{T}(v)$.

Acknowledgments Alexander Rush and Michael Collins were supported under the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022. Michael Collins was also supported by NSF grant IIS-0915176. We also thank the anonymous reviewers for very helpful comments; we hope to fully address these in an extended version of the paper.

References

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

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270. Association for Computational Linguistics.

D. Chiang. 2007. Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

Adria de Gispert, Gonzalo Iglesias, Graeme Blackwood, Eduardo R. Banga, and William Byrne. 2010. Hierarchical Phrase-Based Translation with Weighted Finite-State Transducers and Shallow-n Grammars. In *Computational linguistics*, volume 36, pages 505–533.

Robert W. Floyd. 1962. Algorithm 97: Shortest path. *Commun. ACM*, 5:345.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151, Prague, Czech Republic, June. Association for Computational Linguistics.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 273–283, Cambridge, MA, October. Association for Computational Linguistics.

Gonzalo Iglesias, Adrià de Gispert, Eduardo R. Banga, and William Byrne. 2009. Rule filtering by pattern for efficient hierarchical translation. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 380–388, Athens, Greece, March. Association for Computational Linguistics.

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

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

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

Shankar Kumar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 161–168, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

I. Langkilde. 2000. Forest-based statistical sentence generation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 170–177. Morgan Kaufmann Publishers Inc.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. Spmt: Statistical machine translation with syntactified target language phrases. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 44–52, Sydney, Australia, July. Association for Computational Linguistics.

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

Slav Petrov, Aria Haghighi, and Dan Klein. 2008. Coarse-to-fine syntactic machine translation using language projections. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 108–116, Honolulu, Hawaii, October. Association for Computational Linguistics.

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

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

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585, Columbus, Ohio, June. Association for Computational Linguistics.

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

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

Roy W. Tromble and Jason Eisner. 2006. A fast finite-state relaxation method for enforcing global constraints on sequence decoding. In *Proceedings of*

the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, HLT-NAACL '06, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2006. Left-to-right target generation for hierarchical phrase-based translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 777–784, Morristown, NJ, USA. Association for Computational Linguistics.

Jigs and Lures: Associating Web Queries with Structured Entities

Patrick Pantel
Microsoft Research
Redmond, WA, USA
ppantel@microsoft.com

Ariel Fuxman
Microsoft Research
Mountain View, CA, USA
arielf@microsoft.com

Abstract

We propose methods for estimating the probability that an entity from an entity database is associated with a web search query. Association is modeled using a query entity click graph, blending general query click logs with vertical query click logs. Smoothing techniques are proposed to address the inherent data sparsity in such graphs, including interpolation using a query synonymy model. A large-scale empirical analysis of the smoothing techniques, over a 2-year click graph collected from a commercial search engine, shows significant reductions in modeling error. The association models are then applied to the task of recommending products to web queries, by annotating queries with products from a large catalog and then mining query-product associations through web search session analysis. Experimental analysis shows that our smoothing techniques improve coverage while keeping precision stable, and overall, that our top-performing model affects 9% of general web queries with 94% precision.

1 Introduction

Commercial search engines use query associations in a variety of ways, including the recommendation of related queries in Bing, 'something different' in Google, and 'also try' and related concepts in Yahoo. Mining techniques to extract such query associations generally fall into four categories: (a) clustering queries by their co-clicked url patterns (Wen et al., 2001; Baeza-Yates et al., 2004); (b) leveraging co-occurrences of sequential queries in web search query sessions (Zhang and Nasraoui, 2006; Boldi et al., 2009); (c) pattern-based extraction over lexico-syntactic structures of individual queries (Paşca and Durme, 2008; Jain and Pantel, 2009); and (d) distributional similarity techniques over news or web corpora (Agirre et al., 2009; Pantel et al., 2009). These techniques operate at the surface level, associating one surface context (e.g., queries) to another.

In this paper, we focus instead on associating surface contexts with entities that refer to a particular entry in a knowledge base such as Freebase, IMDB, Amazon's product catalog, or The Library of Congress. Whereas the former models might associate the string "*Ronaldinho*" with the strings "*AC Milan*" or "*Lionel Messi*", our goal is to associate "*Ronaldinho*" with, for example, the Wikipedia entity page "*wiki/AC_Milan*" or the Freebase entity "*en/lionel_mess*". Or for the query string "*ice fishing*", we aim to recommend products in a commercial catalog, such as jigs or lures.

The benefits and potential applications are large. By knowing the entity identifiers associated with a query (instead of strings), one can greatly improve both the presentation of search results as well as the click-through experience. For example, consider when the associated entity is a product. Not only can we present the product name to the web user, but we can also display the image, price, and reviews associated with the entity identifier. Once the entity is clicked, instead of issuing a simple web search query, we can now directly show a product page for the exact product; or we can even perform actions directly on the entity, such as buying the entity on Amazon.com, retrieving the product's oper-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 83–92,
Portland, Oregon, June 19-24, 2011.

ating manual, or even polling your social network for friends that own the product. This is a big step towards a richer semantic search experience.

In this paper, we define the association between a query string q and an entity id e as the probability that e is relevant given the query q, $P(e|q)$. Following Baeza-Yates et al. (2004), we model relevance as the likelihood that a user would click on e given q, events which can be observed in large query-click graphs. Due to the extreme sparsity of query click graphs (Baeza-Yates, 2004), we propose several smoothing models that extend the click graph with query synonyms and then use the synonym click probabilities as a background model. We demonstrate the effectiveness of our smoothing models, via a large-scale empirical study over real-world data, which significantly reduce model errors. We further apply our models to the task of query-product recommendation. Queries in session logs are annotated using our association probabilities and recommendations are obtained by modeling session-level query-product co-occurrences in the annotated sessions. Finally, we demonstrate that our models affect 9% of general web queries with 94% recommendation precision.

2 Related Work

We introduce a novel application of significant commercial value: entity recommendations for general Web queries. This is different from the vast body of work on query suggestions (Baeza-Yates et al., 2004; Fuxman et al., 2008; Mei et al., 2008b; Zhang and Nasraoui, 2006; Craswell and Szummer, 2007; Jagabathula et al., 2011), because our suggestions are actual entities (as opposed to queries or documents). There is also a rich literature on recommendation systems (Sarwar et al., 2001), including successful commercial systems such as the Amazon product recommendation system (Linden et al., 2003) and the Netflix movie recommendation system (Bell et al., 2007). However, these are entity-to-entity recommendations systems. For example, Netflix recommends movies based on previously seen movies (i.e., entities). Furthermore, these systems have access to previous transactions (i.e., actual movie rentals or product purchases), whereas our recommendation system leverages a different resource, namely query sessions.

In principle, one could consider vertical search engines (Nie et al., 2007) as a mechanism for associating queries to entities. For example, if we type the query "canon eos digital camera" on a commerce search engine such as Bing Shopping or Google Products, we get a listing of digital camera entities that satisfy our query. However, vertical search engines are essentially rankers that given a query, return a sorted list of (pointers to) entities that are related to the query. That is, they do not expose actual association scores, which is a key contribution of our work, nor do they operate on general search queries.

Our smoothing methods for estimating association probabilities are related to techniques developed by the NLP and speech communities to smooth n-gram probabilities in language modeling. The simplest are discounting methods, such as *additive smoothing* (Lidstone, 1920) and *Good-Turing* (Good, 1953). Other methods leverage lower-order background models for low-frequency events, such as Katz' backoff smoothing (Katz, 1987), Witten-Bell discounting (Witten and Bell, 1991), Jelinek-Mercer interpolation (Jelinek and Mercer, 1980), and Kneser-Ney (Kneser and Ney, 1995).

In the information retrieval community, Ponte and Croft (1998) are credited for accelerating the use of language models. Initial proposals were based on learning *global* smoothing models, where the smoothing of a word would be independent of the document that the word belongs to (Zhai and Lafferty, 2001). More recently, a number of *local* smoothing models have been proposed (Liu and Croft, 2004; Kurland and Lee, 2004; Tao et al., 2006). Unlike global models, local models leverage relationships between documents in a corpus. In particular, they rely on a graph structure that represents document similarity. Intuitively, the smoothing of a word in a document is influenced by the smoothing of the word in similar documents. For a complete survey of these methods and a general optimization framework that encompasses all previous proposals, please see the work of Mei, Zhang et al. (2008a). All the work on local smoothing models has been applied to the prediction of priors for words in documents. To the best of our knowledge, we are the first to establish that query-click graphs can be used to

create accurate models of query-entity associations.

3 Association Model

Task Definition: Consider a collection of entities E. Given a search query q, our task is to compute $P(e|q)$, the probability that an entity e is relevant to q, for all $e \in E$.

We limit our model to sets of entities that can be accessed through urls on the web, such as Amazon.com products, IMDB movies, Wikipedia entities, and Yelp points of interest.

Following Baeza-Yates et al. (2004), we model relevance as the click probability of an entity given a query, which we can observe from click logs of vertical search engines, i.e., domain-specific search engines such as the product search engine at Amazon, the local search engine at Yelp, or the travel search engine at Bing Travel. Clicked results in a vertical search engine are edges between queries and entities e in the vertical's knowledge base. General search query click logs, which capture direct user intent signals, have shown significant improvements when used for web search ranking (Agichtein et al., 2006). Unlike for general search engines, vertical search engines have typically much less traffic resulting in extremely sparse click logs.

In this section, we define a graph structure for recording click information and we propose several models for estimating $P(e|q)$ using the graph.

3.1 Query Entity Click Graph

We define a *query entity click graph*, $QEC(Q \cup U \cup E, C_u \cup C_e)$, as a tripartite graph consisting of a set of query nodes Q, url nodes U, entity nodes E, and weighted edges C_u exclusively between nodes of Q and nodes of U, as well as weighted edges C_e exclusively between nodes of Q and nodes of E. Each edge in C_u and C_e represents the number of clicks observed between query-url pairs and query-entity pairs, respectively. Let $w_u(q, u)$ be the click weight of the edges in C_u, and $w_e(q, e)$ be the click weight of the edges in C_e.

If C_e is very large, then we can model the association probability, $P(e|q)$, as the maximum likelihood estimation (MLE) of observing clicks on e given the query q:

$$\hat{P}_{mle}(e|q) = \frac{w_e(q,e)}{\sum_{e' \in E} w_e(q,e')} \quad (3.1)$$

Figure 1 illustrates an example query entity graph linking general web queries to entities in a large commercial product catalog. Figure 1a illustrates eight queries in Q with their observed clicks (solid lines) with products in E[1]. Some probability estimates, assigned by Equation 3.1, include: $\hat{P}_{mle}(\text{panfish jigs}, e_1) = 0$, $\hat{P}_{mle}(\text{ice jigs}, e_1) = 1$, and $\hat{P}_{mle}(\text{ice auger}, e_4) = \frac{c_e(\text{ice auger},e_4)}{c_e(\text{ice auger},e_3)+c_e(\text{ice auger},e_4)}$.

Even for the largest search engines, query click logs are extremely sparse, and smoothing techniques are necessary (Craswell and Szummer, 2007; Gao et al., 2009). By considering only C_e, those clicked urls that map to our entity collection E, the sparsity situation is even more dire. The sparsity of the graph comes in two forms: a) there are many queries for which an entity is relevant that will never be seen in the click logs (e.g., "panfish jig" in Figure 1a); and b) the query-click distribution is Zipfian and most observed edges will have very low click counts yielding unreliable statistics. In the following subsections, we present a method to expand QEC with unseen queries that are associated with entities in E. Then we propose smoothing methods for leveraging a background model over the expanded click graph.

Throughout our models, we make the simplifying assumption that the knowledge base E is complete.

3.2 Graph Expansion

Following Gao et al. (2009), we address the sparsity of edges in C_e by inferring new edges through traversing the query-url click subgraph, $UC(Q \cup U, C_u)$, *which contains many more edges than* C_e. If two queries q_i and q_j are synonyms or near synonyms[2], then we expect their click patterns to be similar.

We define the synonymy similarity, $s(q_i, q_j)$ as the cosine of the angle between $\mathbf{q_i}$ and $\mathbf{q_j}$, the click pattern vectors of q_i and q_j, respectively:

$$\text{cosine}(\mathbf{q_i}, \mathbf{q_j}) = \frac{\mathbf{q_i} \cdot \mathbf{q_j}}{\sqrt{\mathbf{q_i} \cdot \mathbf{q_i}} \cdot \sqrt{\mathbf{q_j} \cdot \mathbf{q_j}}}$$

where $\mathbf{q}$ is an n_u dimensional vector consisting of the pointwise mutual information between q and each url u in U, $\text{pmi}(q, u)$:

[1] Clicks are collected from a commerce vertical search engine described in Section 5.1.

[2] A query q_i is a near synonym of a query q_j if most relevant results of q_i are also relevant to q_j. Section 5.2.1 describes our adopted metric for near synonymy.

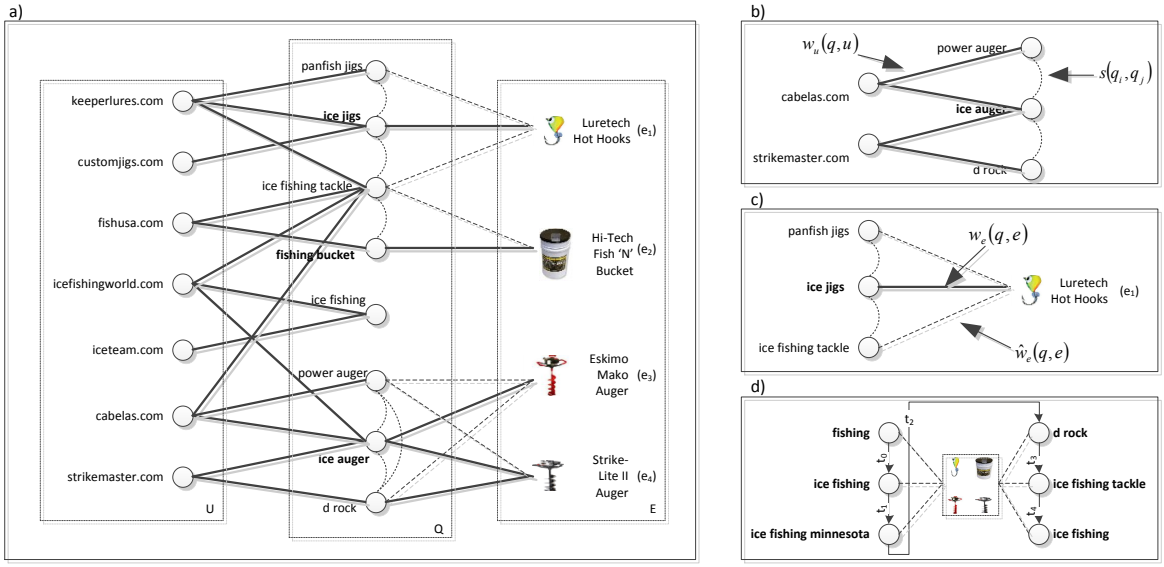


Figure 1: Example QEC graph: (a) Sample queries in Q, clicks connecting queries with urls in U, and clicks to entities in E; (b) Zoom on edges in C_u illustrating clicks observed on urls with weight $w_u(q,u)$ as well as synonymy edges between queries with similarity score $s(q_i,q_j)$ (Section 3.2); (c) Zoom on edges in C_e where solid lines indicate observed clicks with weight $w_e(q,e)$ and dotted lines indicate inferred clicks with smoothed weight $\hat{w}_e(q,e)$ (Section 3.3); and (d) A temporal sequence of queries in a search session illustrating entity associations propagating from the QEC graph to the queries in the session (Section 4).

$$\text{pmi}(q,u) = \log\left(\frac{w_u(q,u)\times\sum_{q'\in Q,u'\in U} w_u(q',u')}{\sum_{u'\in U} w_u(q,u')\sum_{q'\in Q} w_u(q',u)}\right) \tag{3.2}$$

PMI is known to be biased towards infrequent events. We apply the discounting factor, $\delta(q,u)$, proposed in (Pantel and Lin, 2002):

$$\delta(q,u)=\frac{w_u(q,u)}{w_u(q,u)+1}\cdot\frac{\min\left(\sum_{q'\in Q} w_u(q',u),\sum_{u'\in U} w_u(q,u')\right)}{\min\left(\sum_{q'\in Q} w_u(q',u),\sum_{u'\in U} w_u(q,u')\right)+1}$$

Enrichment: We enrich the original QEC graph by creating a new edge $\{q',e\}$, where $q'\in Q$ and $e\in E$, if there exists a query q where $s(q,q')>\rho$ and $w_e(q,e)>0$. ρ is set experimentally, as described in Section 5.2.

Figure 1b illustrates similarity edges created between query "ice auger" and both "power auger" and "d rock". Since "ice auger" was connected to entities e_3 and e_4 in the original QEC, our expansion model creates new edges in C_e between {power auger, e_3}, {power auger, e_4}, and {d rock, e_3}.

For each newly added edge $\{q,e\}$, $\hat{P}_{mle}=0$ according to our model from Equation 3.1 since we have never observed any clicks between q and e. Instead, we define a new model that uses $\hat{P}_{mle}$ when clicks are observed and otherwise assigns uniform probability mass, as:

$$\hat{P}_{hybr}(e|q)=\begin{cases}\hat{P}_{mle}(e|q) & \text{if } \exists e'|w_e(q,e')>0\\ \frac{1}{\sum_{e'\in E}\phi(q,e')} & \text{otherwise}\end{cases} \tag{3.3}$$

where $\phi(q,e)$ is an indicator variable which is 1 if there is an edge between $\{q,e\}$ in C_e.

This model does not leverage the local synonymy graph in order to transfer edge weight to unseen edges. In the next section, we investigate smoothing techniques for achieving this.

3.3 Smoothing

Smoothing techniques can be useful to alleviate data sparsity problems common in statistical models. In practice, methods that leverage a background model (e.g., a lower-order *n*-gram model) have shown most promise (Katz, 1987; Witten and Bell, 1991; Jelinek and Mercer, 1980; Kneser and Ney, 1995). In this section, we present two smoothing methods, derived from Jelinek-Mercer interpolation (Jelinek and Mercer, 1980), for estimating the target association probability $P(e|q)$.

Figure 1c highlights two edges, illustrated with dashed lines, inserted into C_e during the graph expansion phase of Section 3.2. $\hat{w}_e(q,e)$ represents the weight of our background model, which can be viewed as smoothed click counts, and are obtained

Label	Model	Reference
UNIF	$\hat{P}_{unif}(e\|q)$	Eq. 3.8
MLE	$\hat{P}_{mle}(e\|q)$	Eq. 3.1
HYBR	$\hat{P}_{hybr}(e\|q)$	Eq. 3.3
INTU	$\hat{P}_{intu}(e\|q)$	Eq. 3.6
INTP	$\hat{P}_{intp}(e\|q)$	Eq. 3.7

Table 1: Models for estimating the association probability $P(e|q)$.

by propagating clicks to unseen edges using the synonymy model as follows:

$$\hat{w}_e(q,e) = \sum_{q' \in Q} \frac{s(q,q')}{N_{s_q}} \times \hat{P}_{mle}(e|q') \quad (3.4)$$

where $N_{s_q} = \sum_{q' \in Q} s(q,q')$. By normalizing the smoothed weights, we obtain our background model, $\hat{P}_{bsim}$:

$$\hat{P}_{bsim}(e|q) = \frac{\hat{w}_e(q,e)}{\sum_{e' \in E} \hat{w}_e(q,e')} \quad (3.5)$$

Below we propose two models for interpolating our foreground model from Equation 3.1 with the background model from Equation 3.5.

Basic Interpolation: This smoothing model, $\hat{P}_{intu}(e|q)$, linearly combines our foreground and background models using a model parameter α:

$$\hat{P}_{intu}(e|q) = \alpha \hat{P}_{mle}(e|q) + (1-\alpha)\hat{P}_{bsim}(e|q) \quad (3.6)$$

Bucket Interpolation: Intuitively, edges $\{q,e\} \in C_e$ with higher observed clicks, $w_e(q,e)$, should be trusted more than those with low or no clicks. A limitation of $\hat{P}_{intu}(e|q)$ is that it weighs the foreground and background models in the same way irrespective of the observed foreground clicks. Our final model, $\hat{P}_{intp}(e|q)$ parameterizes the interpolation by the number of observed clicks:

$$\begin{aligned}\hat{P}_{intp}(e|q) = &\alpha[w_e(q,e)]\hat{P}_{mle}(e|q) \\ &+ (1 - \alpha[w_e(q,e)])\hat{P}_{bsim}(e|q)\end{aligned} \quad (3.7)$$

In practice, we bucket the observed click parameter, $w_e(q,e)$, into eleven buckets: {1-click, 2-clicks, ..., 10-clicks, more than 10 clicks}.

Section 5.2 outlines our procedure for learning the model parameters for both $\hat{P}_{intu}(e|q)$ and $\hat{P}_{intp}(e|q)$.

3.4 Summary

Table 1 summarizes the association models presented in this section as well as a strawman that assigns uniform probability to all edges in QEC:

$$\hat{P}_{unif}(e|q) = \frac{1}{\sum_{e' \in E} \phi(q,e')} \quad (3.8)$$

In the following section, we apply these models to the task of extracting product recommendations for general web search queries. A large-scale experimental study is presented in Section 5 supporting the effectiveness of our models.

4 Entity Recommendation

Query recommendations are pervasive in commercial search engines. Many systems extract recommendations by mining temporal query chains from search sessions and clickthrough patterns (Zhang and Nasraoui, 2006). We adopt a similar strategy, except instead of mining query-query associations, we propose to mine query-entity associations, where entities come from an entity database as described in Section 1. Our technical challenge lies in annotating sessions with entities that are relevant to the session.

4.1 Product Entity Domain

Although our model generalizes to any entity domain, we focus now on a product domain. Specifically, our universe of entities, E, consists of the entities in a large commercial product catalog, for which we observe query-click-product clicks, C_e, from the vertical search logs. Our QEC graph is completed by extracting query-click-urls from a search engine's general search logs, C_u. These datasets are described in Section 5.1.

4.2 Recommendation Algorithm

We hypothesize that if an entity is relevant to a query, then it is relevant to all other queries co-occurring in the same session. Key to our method are the models from Section 3.

Step 1 – Query Annotation: For each query q in a session s, we annotate it with a set E_q, consisting of every pair $\{e, \hat{P}(e|q)\}$, where $e \in E$ such that there exists an edge $\{q,e\} \in C_e$ with probability $\hat{P}(e|q)$. Note that E_q will be empty for many queries.

Step 2 – Session Analysis: We build a query-entity frequency co-occurrence matrix, $\mathbf{A}$, consisting of $n_{|Q|}$ rows and $n_{|E|}$ columns, where each row corresponds to a query and each column to an entity.

The value of the cell A_{qe} is the sum over each session s, of the maximum edge weight between any query $q' \in s$ and e[3]:

$$A_{qe} = \sum_{s \in \mathbf{S}} \psi(s, e)$$

where $\mathbf{S}$ consists of all observed search sessions and:

$$\psi(s, e) = \arg\max_{\hat{P}(e|q')} (\{e, \hat{P}(e|q')\} \in E_{q'}), \forall q' \in s$$

Step 3 – Ranking: We compute ranking scores between each query q and entity e using pointwise mutual information over the frequencies in $\mathbf{A}$, similarly to Eq. 3.2.

The final recommendations for a query q are obtained by returning the top-k entities e according to Step 3. Filters may be applied on: f the frequency A_{qe}; and p the pointwise mutual information ranking score between q and e.

5 Experimental Results

5.1 Datasets

We instantiate our models from Sections 3 and 4 using search query logs and a large catalog of products from a commercial search engine. We form our QEC graphs by first collecting in C_e aggregate query-click-entity counts observed over two years in a commerce vertical search engine. Similarly, C_u is formed by collecting aggregate query-click-url counts observed over six months in a web search engine, where each query must have frequency at least 10. Three final QEC graphs are sampled by taking various snapshots of the above graph as follows: a) TRAIN consists of 50% of the graph; b) TEST consists of 25% of the graph; c) DEV consists of 25% of the graph.

5.2 Association Models

5.2.1 Model Parameters

We tune the α parameters for $\hat{P}_{intu}$ and $\hat{P}_{intp}$ against the DEV QEC graph. There are twelve parameters to be tuned: α for $\hat{P}_{intu}$ and $\alpha(1)$, $\alpha(2)$, ..., $\alpha(10)$, $\alpha(> 10)$ for $\hat{P}_{intp}$, where $\alpha(x)$ is the observed click bucket as described in Section 3.3. For each, we choose the parameter value that minimizes the mean-squared error (MSE) of the DEV set, where model probabilities are computed using the TRAIN QEC graph. Figure 2 illustrates the MSE ranging over [0, 0.05, 0.1, ..., 1].

[3]Note that this co-occurrence occurs because q' was annotated with entity e in the same session as q occurred.

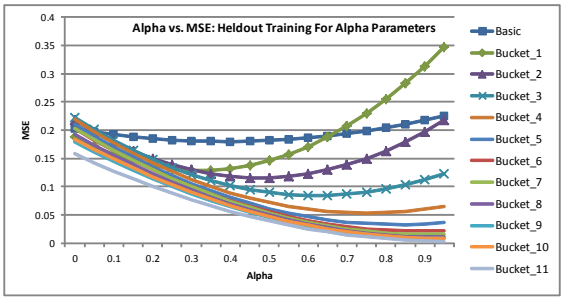


Figure 2: Alpha tuning on held out data.

Model	MSE	Var	Err/MLE	MSE_W	Var	Err/MLE
$\hat{P}_{unif}$	0.0328†	0.0112	-25.7%	0.0663†	0.0211	-71.8%
$\hat{P}_{mle}$	0.0261	0.0111	–	0.0386	0.0141	–
$\hat{P}_{hybr}$	0.0232†	0.0071	11.1%	0.0385	0.0132	0.03%
$\hat{P}_{intu}$	**0.0226**†	**0.0075**	**13.4%**	**0.0369**†	**0.0133**	**4.4%**
$\hat{P}_{intp}$	**0.0213**†	**0.0068**	**18.4%**	**0.0375**†	**0.0131**	**2.8%**

Table 2: Model analysis: MSE and MSE_W with variance and error reduction relative to $\hat{P}_{mle}$. † indicates statistical significance over $\hat{P}_{mle}$ with 95% confidence.

We trained the query synonym model of Section 3.2 on the DEV set and hand-annotated 100 random synonymy pairs according to whether or not the pairs were synonyms [2]. Setting $\rho = 0.4$ results in a precision > 0.9.

5.2.2 Analysis

We evaluate the quality of our models in Table 1 by evaluating their mean-squared error (MSE) against the target $P(e|q)$ computed on the TEST set:

$$MSE(\hat{P}) = \sum_{\{q,e\} \in C_e^T} (P^T(e|q) - \hat{P}(e|q))^2$$

$$MSE_W(\hat{P}) = \sum_{\{q,e\} \in C_e^T} w_e^T(q,e) \cdot (P^T(e|q) - \hat{P}(e|q))^2$$

where C_e^T are the edges in the TEST QEC graph with weight $w_e^T(q, e)$, $P^T(e|q)$ is the target probability computed over the TEST QEC graph, and $\hat{P}$ is one of our models trained on the TRAIN QEC graph. MSE measures against each edge type, which makes it sensitive to the long tail of the click graph. Conversely, MSE_W measures against each edge instance, which makes it a good measure against the head of the click graph. We expect our smoothing models to have much more impact on MSE (i.e., the tail) than on MSE_W since head queries do not suffer from data sparsity.

Table 2 lists the MSE and MSE_W results for each model. We consider $\hat{P}_{unif}$ as a strawman and $\hat{P}_{mle}$ as a strong baseline (i.e., without any graph expansion nor any smoothing against a background

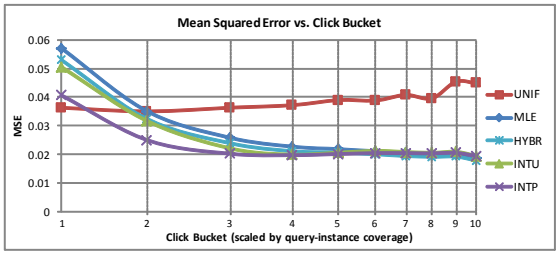


Figure 3: MSE of each model against the number of clicks in the TEST corpus. Buckets scaled by query *instance* coverage of all queries with 10 or fewer clicks.

model). $\hat{P}_{unif}$ performs generally very poorly, however $\hat{P}_{mle}$ is much better, with an expected estimation error of 0.16 accounting for an MSE of 0.0261. As expected, our smoothing models have little improvement on the head-sensitive metric (MSE_W) relative to $\hat{P}_{mle}$. In particular, $\hat{P}_{hybr}$ performs nearly identically to $\hat{P}_{mle}$ on the head. On the tail, all three smoothing models significantly outperform $\hat{P}_{mle}$ with $\hat{P}_{intp}$ reducing the error by 18.4%. Table 3 lists query-product associations for five randomly sampled products along with their model scores from $\hat{P}_{mle}$ with $\hat{P}_{intp}$.

Figure 3 provides an intrinsic view into MSE as a function of the number of observed clicks in the TEST set. As expected, for larger observed click counts (>4), all models perform roughly the same, indicating that smoothing is not necessary. However, for low click counts, which in our dataset accounts for over 20% of the overall click instances, we see a large reduction in MSE with $\hat{P}_{intp}$ outperforming $\hat{P}_{intu}$, which in turn outperforms $\hat{P}_{hybr}$. $\hat{P}_{unif}$ performs very poorly. The reason it does worse as the observed click count rises is that head queries tend to result in more distinct urls with high-variance clicks, which in turn makes a uniform model susceptible to more error.

Figure 3 illustrates that the benefit of the smoothing models is in the tail of the click graph, which supports the larger error reductions seen in MSE in Table 2. For associations only observed once, $\hat{P}_{intp}$ reduces the error by 29% relative to $\hat{P}_{mle}$.

We also performed an editorial evaluation of the query-entity associations obtained with bucket interpolation. We created two samples from the TEST dataset: one randomly sampled by taking click weights into account, and the other sampled uniformly at random. Each set contains results for

Query	$\hat{P}_{mle}$	$\hat{P}_{intp}$	Query	$\hat{P}_{mle}$	$\hat{P}_{intp}$
Garmin GTM 20 GPS			Canon PowerShot SX110 IS		
garmin gtm 20	0.44	0.45	canon sx110	0.57	0.57
garmin traffic receiver	0.30	0.27	powershot sx110	0.48	0.48
garmin nuvi 885t	0.02	0.02	powershot sx110 is	0.38	0.36
gtm 20	0	0.33	**powershot sx130 is**	0	0.33
garmin gtm20	0	0.33	**canon power shot sx110**	0	0.20
nuvi 885t	0	0.01	**canon dig camera review**	0	0.10
Samsung PN50A450 50" TV			Devil May Cry: 5th Anniversary Col.		
samsung 50 plasma hdtv	0.75	0.83	devil may cry	0.76	0.78
samsung 50	0.33	0.32	**devilmaycry**	0	1.00
50" hdtv	0.17	0.12	High Island Hammock/Stand Combo		
samsung plasma tv review	0	0.42	high island hammocks	1.00	1.00
50" samsung plasma hdtv	0	0.35	**hammocks and stands**	0	0.10

Table 3: Example query-product association scores for a random sample of five products. Bold queries resulted from the expansion algorithm in Section 3.2.

100 queries. The former consists of 203 query-product associations, and the latter of 159 associations. The evaluation was done using Amazon Mechanical Turk[4]. We created a Mechanical Turk HIT[5] where we show to the Mechanical Turk workers the query and the actual Web page in a Product search engine. For each query-entity association, we gathered seven labels and considered an association to be correct if five Mechanical Turk workers gave a positive label. An association was considered to be incorrect if at least five workers gave a negative label. Borderline cases where no label got five votes were discarded (14% of items were borderline for the uniform sample; 11% for the weighted sample). To ensure the quality of the results, we introduced 30% of incorrect associations as honeypots. We blocked workers who responded incorrectly on the honeypots so that the precision on honeypots is 1. The result of the evaluation is that the precision of the associations is 0.88 on the weighted sample and 0.90 on the uniform sample.

5.3 Related Product Recommendation

We now present an experimental evaluation of our product recommendation system using the baseline model $\hat{P}_{mle}$ and our best-performing model $\hat{P}_{intp}$. The goals of this evaluation are to (1) determine the quality of our product recommendations; and (2) assess the impact of our association models on the product recommendations.

5.3.1 Experimental Setup

We instantiate our recommendation algorithm from Section 4.2 using session co-occurrence frequencies

[4] `https://www.mturk.com`

[5] HIT stands for Human Intelligence Task

	Query Set Sample				Query Bag Sample			
f	10	25	50	100	10	25	50	100
p	10	10	10	10	10	10	10	10
$\hat{P}_{mle}$ precision	0.89	0.93	0.96	0.96	0.94	0.94	0.93	0.92
$\hat{P}_{intp}$ precision	0.86	0.92	0.96	0.96	0.94	0.94	0.93	0.94
$\hat{P}_{mle}$ coverage	0.007	0.004	0.002	0.001	0.085	0.067	0.052	0.039
$\hat{P}_{intp}$ coverage	0.008	0.005	0.003	0.002	0.094	0.076	0.059	0.045
$R_{intp,mle}$	1.16	1.14	1.13	1.14	1.11	1.13	1.15	1.19

Table 4: Experimental results for product recommendations. All configurations are for $k = 10$.

from a one-month snapshot of user query sessions at a Web search engine, where session boundaries occur when 60 seconds elapse in between user queries. We experiment with the recommendation parameters defined at the end of Section 4.2 as follows: k = 10, f ranging from 10 to 100, and p ranging from 3 to 10.

For each configuration, we report *coverage* as the total number of queries in the output (i.e., the queries for which there is some recommendation) divided by the total number of queries in the log. For our performance metrics, we sampled two sets of queries: (a) `Query Set Sample`: uniform random sample of 100 queries from the *unique* queries in the one-month log; and (b) `Query Bag Sample`: weighted random sample, by query frequency, of 100 queries from the query *instances* in the one-month log. For each sample query, we pooled together and randomly shuffled all recommendations by our algorithm using both $\hat{P}_{mle}$ and $\hat{P}_{intp}$ on each parameter configuration. We then manually annotated each {query, product} pair as *relevant*, *mildly relevant* or *non-relevant*. In total, 1127 pairs were annotated. Interannotator agreement between two judges on this task yielded a Cohen's Kappa (Cohen, 1960) of 0.56. We therefore collapsed the *mildly relevant* and *non-relevant* classes yielding two final classes: *relevant* and *non-relevant*. Cohen's Kappa on this binary classification is 0.71.

Let C_M be the number of relevant (i.e., correct) suggestions recommended by a configuration M and let $|M|$ be the number of recommendations returned by M. Then we define the (micro-) precision of M as: $P_M = \frac{C_M}{C}$. We define relative recall (Pantel et al., 2004) between two configurations M_1 and M_2 as $R_{M_1,M_2} = \frac{P_{M_1} \times |M_1|}{P_{M_2} \times |M_2|}$.

5.3.2 Results

Table 4 summarizes our results for some configurations (others omitted for lack of space). Most remarkable is the $\{f = 10, p = 10\}$ configuration where the $\hat{P}_{intp}$ model *affected 9.4% of all query instances posed by the millions of users of a major search engine*, with a precision of 94%. Although this model covers 0.8% of the unique queries, the fact that it covers many head queries such as *walmart* and *iphone* accounts for the large query instance coverage. Also since there may be many general web queries for which there is no appropriate product in the database, a coverage of 100% is not attainable (nor desirable); in fact the upper bound for the coverage is likely to be much lower.

Query	Product Recommendation
wedding gowns	27 Dresses (Movie Soundtrack)
wedding gowns	Bridal Gowns: The Basics of Designing, [...] (Book)
wedding gowns	Wedding Dress Hankie
wedding gowns	The Perfect Wedding Dress (Magazine)
wedding gowns	Imagine Wedding Designer (Video Game)
low blood pressure	Omron Blood Pressure Monitor
low blood pressure	Healthcare Automatic Blood Pressure Monitor
low blood pressure	Ridgecrest Blood Pressure Formula - 60 Capsules
low blood pressure	Omron Portable Wrist Blood Pressure Monitor
'hello cupcake' cookbook	Giant Cupcake Cast Pan
'hello cupcake' cookbook	Ultimate 3-In-1 Storage Caddy
'hello cupcake' cookbook	13 Cup Cupcakes and More Dessert Stand
'hello cupcake' cookbook	Cupcake Stand Set (Toys)
1 800 flowers	Todd Oldham Party Perfect Bouquet
1 800 flowers	Hugs and Kisses Flower Bouquet with Vase

Table 5: Sample product recommendations.

Turning to the impact of the association models on product recommendations, we note that precision is stable in our $\hat{P}_{intp}$ model relative to our baseline $\hat{P}_{mle}$ model. However, a large lift in relative recall is observed, up to a 19% increase for the $\{f = 100, p = 10\}$ configuration. These results are consistent with those of Section 5.2, which compared the association models independently of the application and showed that $\hat{P}_{intp}$ outperforms $\hat{P}_{mle}$.

Table 5 shows sample product recommendations discovered by our $\hat{P}_{intp}$ model. Manual inspection revealed two main sources of errors. First, *ambiguity* is introduced both by the click model and the graph expansion algorithm of Section 3.2. In many cases, the ambiguity is resolved by user click patterns (i.e., users disambiguate queries through their browsing behavior), but one such error was seen for the query "*shark attack videos*" where several *Shark*-branded vacuum cleaners are recommended. This is because of the ambiguous query "*shark*" that is found in the click logs and in query sessions co-occurring with the query "*shark attack videos*". The second source of errors is caused by *systematic user errors* commonly found in session logs such as a user accidentally submitting a query while typing. An example

session is: {"*speedo*", "*speedometer*"} where the intended session was just the second query and the unintended first query is associated with products such as *Speedo swimsuits*. This ultimately causes our system to recommend various swimsuits for the query "*speedometer*".

6 Conclusion

Learning associations between web queries and entities has many possible applications, including query-entity recommendation, personalization by associating entity vectors to users, and direct advertising. Although many techniques have been developed for associating queries to queries or queries to documents, to the best of our knowledge this is the first that aims to associate queries to entities by leveraging click graphs from both general search logs and vertical search logs.

We developed several models for estimating the probability that an entity is relevant given a user query. The sparsity of *query entity graphs* is addressed by first expanding the graph with query synonyms, and then smoothing query-entity click counts over these unseen queries. Our best performing model, which interpolates between a foreground click model and a smoothed background model, significantly reduces testing error when compared against a strong baseline, by 18%. On associations observed only once in our test collection, the modeling error is reduced by 29% over the baseline.

We applied our best performing model to the task of query-entity recommendation, by analyzing session co-occurrences between queries and annotated entities. Experimental analysis shows that our smoothing techniques improve coverage while keeping precision stable, and overall, that our top-performing model affects 9% of general web queries with 94% precision.

References

[Agichtein et al.2006] Eugene Agichtein, Eric Brill, and Susan T. Dumais. 2006. Improving web search ranking by incorporating user behavior information. In *SIGIR*, pages 19–26.

[Agirre et al.2009] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *NAACL*, pages 19–27.

[Baeza-Yates et al.2004] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. 2004. Query recommendation using query logs in search engines. In Wolfgang Lindner, Marco Mesiti, Can Türker, Yannis Tzitzikas, and Athena Vakali, editors, *EDBT Workshops*, volume 3268 of *Lecture Notes in Computer Science*, pages 588–596. Springer.

[Baeza-Yates2004] Ricardo Baeza-Yates. 2004. Web usage mining in search engines. In *In Web Mining: Applications and Techniques, Anthony Scime, editor. Idea Group*, pages 307–321.

[Bell et al.2007] R. Bell, Y. Koren, and C. Volinsky. 2007. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD*, pages 95–104.

[Boldi et al.2009] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, and Sebastiano Vigna. 2009. Query suggestions using query-flow graphs. In *WSCD '09: Proceedings of the 2009 workshop on Web Search Click Data*, pages 56–63. ACM.

[Cohen1960] Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, April.

[Craswell and Szummer2007] Nick Craswell and Martin Szummer. 2007. Random walks on the click graph. In *SIGIR*, pages 239–246.

[Fuxman et al.2008] A. Fuxman, P. Tsaparas, K. Achan, and R. Agrawal. 2008. Using the wisdom of the crowds for keyword generation. In *WWW*, pages 61–70.

[Gao et al.2009] Jianfeng Gao, Wei Yuan, Xiao Li, Kefeng Deng, and Jian-Yun Nie. 2009. Smoothing clickthrough data for web search ranking. In *SIGIR*, pages 355–362.

[Good1953] Irving John Good. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3 and 4):237–264.

[Jagabathula et al.2011] S. Jagabathula, N. Mishra, and S. Gollapudi. 2011. Shopping for products you don't know you need. In *To appear at WSDM*.

[Jain and Pantel2009] Alpa Jain and Patrick Pantel. 2009. Identifying comparable entities on the web. In *CIKM*, pages 1661–1664.

[Jelinek and Mercer1980] Frederick Jelinek and Robert L. Mercer. 1980. Interpolated estimation of markov source parameters from sparse data. In *In Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397.

[Katz1987] Slava M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *IEEE Transactions on*

Acoustics, Speech and Signal Processing, pages 400–401.

[Kneser and Ney1995] Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 181–184.

[Kurland and Lee2004] O. Kurland and L. Lee. 2004. Corpus structure, language models, and ad-hoc information retrieval. In *SIGIR*, pages 194–201.

[Lidstone1920] George James Lidstone. 1920. Note on the general case of the bayes-laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries*, 8:182–192.

[Linden et al.2003] G. Linden, B. Smith, and J. York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80.

[Liu and Croft2004] X. Liu and W. Croft. 2004. Cluster-based retrieval using language models. In *SIGIR*, pages 186–193.

[Mei et al.2008a] Q. Mei, D. Zhang, and C. Zhai. 2008a. A general optimization framework for smoothing language models on graph structures. In *SIGIR*, pages 611–618.

[Mei et al.2008b] Q. Mei, D. Zhou, and Church K. 2008b. Query suggestion using hitting time. In *CIKM*, pages 469–478.

[Nie et al.2007] Z. Nie, J. Wen, and W. Ma. 2007. Object-level vertical search. In *Conference on Innovative Data Systems Research (CIDR)*, pages 235–246.

[Pantel and Lin2002] Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *SIGKDD*, pages 613–619, Edmonton, Canada.

[Pantel et al.2004] Patrick Pantel, Deepak Ravichandran, and Eduard Hovy. 2004. Towards terascale knowledge acquisition. In *COLING*, pages 771–777.

[Pantel et al.2009] Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *EMNLP*, pages 938–947.

[Paşca and Durme2008] Marius Paşca and Benjamin Van Durme. 2008. Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. In *ACL*, pages 19–27.

[Ponte and Croft1998] J. Ponte and B. Croft. 1998. A language modeling approach to information retrieval. In *SIGIR*, pages 275–281.

[Sarwar et al.2001] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. 2001. Item-based collaborative filtering recommendation system. In *WWW*, pages 285–295.

[Tao et al.2006] T. Tao, X. Wang, Q. Mei, and C. Zhai. 2006. Language model information retrieval with document expansion. In *HLT/NAACL*, pages 407–414.

[Wen et al.2001] Ji-Rong Wen, Jian-Yun Nie, and HongJiang Zhang. 2001. Clustering user queries of a search engine. In *WWW*, pages 162–168.

[Witten and Bell1991] I.H. Witten and T.C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4).

[Zhai and Lafferty2001] C. Zhai and J. Lafferty. 2001. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR*, pages 334–342.

[Zhang and Nasraoui2006] Z. Zhang and O. Nasraoui. 2006. Mining search engine query logs for query recommendation. In *WWW*, pages 1039–1040.

Semi-Supervised SimHash for Efficient Document Similarity Search

Qixia Jiang and Maosong Sun
State Key Laboratory on Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Sci. and Tech., Tsinghua University, Beijing 100084, China
qixia.jiang@gmail.com, sms@tsinghua.edu.cn

Abstract

Searching documents that are similar to a query document is an important component in modern information retrieval. Some existing hashing methods can be used for efficient document similarity search. However, unsupervised hashing methods cannot incorporate prior knowledge for better hashing. Although some supervised hashing methods can derive effective hash functions from prior knowledge, they are either computationally expensive or poorly discriminative. This paper proposes a novel (semi-)supervised hashing method named Semi-Supervised SimHash (S^3H) for high-dimensional data similarity search. The basic idea of S^3H is to learn the optimal feature weights from prior knowledge to relocate the data such that similar data have similar hash codes. We evaluate our method with several state-of-the-art methods on two large datasets. All the results show that our method gets the best performance.

1 Introduction

Document Similarity Search (DSS) is to find similar documents to a query doc in a text corpus or on the web. It is an important component in modern information retrieval since DSS can improve the traditional search engines and user experience (Wan *et al.*, 2008; Dean *et al.*, 1999). Traditional search engines accept several terms submitted by a user as a query and return a set of docs that are relevant to the query. However, for those users who are not search experts, it is always difficult to accurately specify some query terms to express their search purposes. Unlike short-query based search, DSS queries by a full (long) document, which allows users to directly submit a page or a document to the search engines as the description of their information needs. Meanwhile, the explosion of information has brought great challenges to traditional methods. For example, Inverted List (IL) which is a primary key-term access method would return a very large set of docs for a query document, which leads to the time-consuming post-processing. Therefore, a new effective algorithm is required.

Hashing methods can perform highly efficient but approximate similarity search, and have gained great success in many applications such as Content-Based Image Retrieval (CBIR) (Ke *et al.*, 2004; Kulis *et al.*, 2009b), near-duplicate data detection (Ke *et al.*, 2004; Manku *et al.*, 2007; Costa *et al.*, 2010), etc. Hashing methods project high-dimensional objects to compact binary codes called *fingerprint*s and make similar fingerprints for similar objects. The similarity search in the Hamming space[1] is much more efficient than in the original attribute space (Manku *et al.*, 2007).

Recently, several hashing methods have been proposed. Specifically, SimHash (SH) (Charikar M.S., 2002) uses random projections to hash data. Although it works well with long fingerprints, SH has poor discrimination power for short fingerprints. A kernelized variant of SH, called Kernelized Locality Sensitive Hashing (KLSH) (Kulis *et al.*, 2009a), is proposed to handle non-linearly separable data. These methods are unsupervised thus cannot incorporate prior knowledge for better hashing. Moti-

[1] Hamming space is a set of binary strings of length L.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 93–101,
Portland, Oregon, June 19-24, 2011.

vated by this, some supervised methods are proposed to derive effective hash functions from prior knowledge, i.e., Spectral Hashing (Weiss *et al.*, 2009) and Semi-Supervised Hashing (SSH) (Wang *et al.*, 2010a). Regardless of different objectives, both methods derive hash functions via Principle Component Analysis (PCA) (Jolliffe, 1986). However, PCA is computationally expensive, which limits their usage for high-dimensional data.

This paper proposes a novel (semi-)supervised hashing method, Semi-Supervised SimHash (S^3H), for high-dimensional data similarity search. Unlike SSH that tries to find a sequence of hash functions, S^3H fixes the random projection directions and seeks the optimal feature weights from prior knowledge to relocate the objects such that similar objects have similar fingerprints. This is implemented by maximizing the empirical accuracy on the prior knowledge (labeled data) and the entropy of hash functions (estimated over labeled and unlabeled data). The proposed method avoids using PCA which is computationally expensive especially for high-dimensional data, and leads to an efficient Quasi-Newton based solution. To evaluate our method, we compare with several state-of-the-art hashing methods on two large datasets, i.e., 20 Newsgroups (20K points) and Open Directory Project (ODP) (2.4 million points). All experiments show that S^3H gets the best search performance.

This paper is organized as follows: Section 2 briefly introduces the background and some related works. In Section 3, we describe our proposed Semi-Supervised SimHash (S^3H). Section 4 provides experimental validation on two datasets. The conclusions are given in Section 5.

2 Background and Related Works

Suppose we are given a set of N documents, $\mathcal{X} = \{\mathbf{x}_i \mid \mathbf{x}_i \in \mathbb{R}^M\}_{i=1}^N$. For a given query doc $\mathbf{q}$, DSS tries to find its nearest neighbors in $\mathcal{X}$ or a subset $\mathcal{X}' \subset \mathcal{X}$ in which distance from the documents to the query doc $\mathbf{q}$ is less than a give threshold. However, such two tasks are computationally infeasible for large-scale data. Thus, it turns to the approximate similarity search problem (Indyk *et al.*, 1998). In this section, we briefly review some related approximate similarity search methods.

2.1 SimHash

SimHash (SH) is first proposed by Charikar (Charikar M.S., 2002). SH uses random projections as hash functions, i.e.,

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x}) = \begin{cases} +1, & if\ \mathbf{w}^T\mathbf{x} \geq 0 \\ -1, & otherwise \end{cases} \quad (1)$$

where $\mathbf{w} \in \mathbb{R}^M$ is a vector randomly generated. SH specifies the distribution on a family of hash functions $\mathcal{H} = \{h\}$ such that for two objects $\mathbf{x}_i$ and $\mathbf{x}_j$,

$$\Pr_{h\in\mathcal{H}}\{h(\mathbf{x}_i) = h(\mathbf{x}_j)\} = 1 - \frac{\theta(\mathbf{x}_i, \mathbf{x}_j)}{\pi} \quad (2)$$

where $\theta(\mathbf{x}_i, \mathbf{x}_j)$ is the angle between $\mathbf{x}_i$ and $\mathbf{x}_j$. Obviously, SH is an unsupervised hashing method.

2.2 Kernelized Locality Sensitive Hashing

A kernelized variant of SH, named Kernelized Locality Sensitive Hashing (KLSH) (Kulis *et al.*, 2009a), is proposed for non-linearly separable data. KLSH approximates the underling Gaussian distribution in the implicit embedding space of data based on central limit theory. To calculate the value of hashing fuction $h(\cdot)$, KLSH projects points onto the eigenvectors of the kernel matrix. In short, the complete procedure of KLSH can be summarized as follows: 1) randomly select P (a small value) points from $\mathcal{X}$ and form the kernel matrix, 2) for each hash function $h(\phi(\mathbf{x}))$, calculate its weight $\boldsymbol{\omega} \in \mathbb{R}^P$ just as Kernel PCA (Schölkopf *et al.*, 1997), and 3) the hash function is defined as:

$$h(\phi(\mathbf{x})) = \text{sign}(\sum_{i=1}^{P} \omega_i \cdot \kappa(\mathbf{x}, \mathbf{x}_i)) \quad (3)$$

where $\kappa(\cdot,\cdot)$ can be any kernel function.

KLSH can improve hashing results via the kernel trick. However, KLSH is unsupervised, thus designing a data-specific kernel remains a big challenge.

2.3 Semi-Supervised Hashing

Semi-Supervised Hashing (SSH) (Wang *et al.*, 2010a) is recently proposed to incorporate prior knowledge for better hashing. Besides $\mathcal{X}$, prior knowledge in the form of similar and dissimilar object-pairs is also required in SSH. SSH tries to find L optimal hash functions which have maximum

empirical accuracy on prior knowledge and maximum entropy by finding the top L eigenvectors of an extended covariance matrix[2] via PCA or SVD.

However, despite of the potential problems of numerical stability, SVD requires massive computational space and $O(M^3)$ computational time where M is feature dimension, which limits its usage for high-dimensional data (Trefethen *et al.*, 1997). Furthermore, the variance of directions obtained by PCA decreases with the decrease of the rank (Jolliffe, 1986). Thus, lower hash functions tend to have smaller entropy and larger empirical errors.

2.4 Others

Some other related works should be mentioned. A notable method is Locality Sensitive Hashing (LSH) (Indyk *et al.*, 1998). LSH performs a random linear projection to map similar objects to similar hash codes. However, LSH suffers from the efficiency problem that it tends to generate long codes (Salakhutdinov *et al.*, 2007). LAMP (Mu *et al.*, 2009) considers each hash function as a binary partition problem as in SVMs (Burges, 1998). Spectral Hashing (Weiss *et al.*, 2009) maintains similarity between objects in the reduced Hamming space by minimizing the averaged Hamming distance[3] between similar neighbors in the original Euclidean space. However, spectral hashing takes the assumption that data should be distributed uniformly, which is always violated in real-world applications.

3 Semi-Supervised SimHash

In this section, we present our hashing method, named Semi-Supervised SimHash (S^{3}H). Let $\mathcal{X}_L = \{(\mathbf{x}_1, c_1) \dots (\mathbf{x}_u, c_u)\}$ be the labeled data, $c \in \{1 \dots C\}$, $\mathbf{x} \in \mathbb{R}^M$, and $\mathcal{X}_U = \{\mathbf{x}_{u+1} \dots \mathbf{x}_N\}$ the unlabeled data. Let $\mathcal{X} = \mathcal{X}_L \cup \mathcal{X}_U$. Given the labeled data $\mathcal{X}_L$, we construct two sets, attraction set Θ_a and repulsion set Θ_r. Specifically, any pair $(\mathbf{x}_i, \mathbf{x}_j) \in \Theta_a$, $i, j \le u$, denotes that $\mathbf{x}_i$ and $\mathbf{x}_j$ are in the same class, i.e., $c_i = c_j$, while any pair $(\mathbf{x}_i, \mathbf{x}_j) \in \Theta_r$, $i, j \le u$, denotes that $c_i \neq c_j$. Unlike previews works that attempt to find L optimal hyperplanes, the basic idea of S^{3}H is to fix L random hyperplanes and to find an optimal feature-weight vector to relocate the objects such that similar objects have similar codes.

3.1 Data Representation

Since L random hyperplanes are fixed, we can represent a object $\mathbf{x} \in \mathcal{X}$ as its relative position to these random hyperplanes, i.e.,

$$\mathbf{D} = \mathbf{\Lambda} \cdot \mathbf{V} \tag{4}$$

where the element $\mathrm{V}_{ml} \in \{+1, -1, 0\}$ of $\mathbf{V}$ indicates that the object $\mathbf{x}$ is above, below or just in the l-th hyperplane with respect to the m-th feature, and $\mathbf{\Lambda} = \mathrm{diag}(|x_1|, |x_2|, \dots, |x_M|)$ is a diagonal matrix which, to some extent, reflects the distance from $\mathbf{x}$ to these hyperplanes.

3.2 Formulation

Hashing maps the data set $\mathcal{X}$ to an L-dimensional Hamming space for compact representations. If we represent each object as Equation (4), the l-th hash function is then defined as:

$$h_l(\mathbf{x}) = \hbar_l(\mathbf{D}) = \mathrm{sign}(\mathbf{w}^T \mathbf{d}_l) \tag{5}$$

where $\mathbf{w} \in \mathbb{R}^M$ is the feature weight to be determined and $\mathbf{d}_l$ is the l-th column of the matrix $\mathbf{D}$.

Intuitively, the "contribution" of a specific feature to different classes is different. Therefore, we hope to incorporate this side information in S^{3}H for better hashing. Inspired by (Madani *et al.*, 2009), we can measure this contribution over $\mathcal{X}_L$ as in Algorithm 1. Clearly, if objects are represented as the occurrence numbers of features, the output of Algorithm 1 is just the conditional probability $\Pr(\mathrm{class}|\mathrm{feature})$. Finally, each object $(\mathbf{x}, c) \in \mathcal{X}_L$ can be represented as an $M \times L$ matrix $\mathbf{G}$:

$$\mathbf{G} = \mathrm{diag}(\nu_{1,c}, \nu_{2,c}, \dots, \nu_{M,c}) \cdot \mathbf{D} \tag{6}$$

Note that, one pair $(\mathbf{x}_i, \mathbf{x}_j)$ in Θ_a or Θ_r corresponds to $(\mathbf{G}_i, \mathbf{G}_j)$ while $(\mathbf{D}_i, \mathbf{D}_j)$ if we ignore features' contribution to different classes.

Furthermore, we also hope to maximize the empirical accuracy on the labeled data Θ_a and Θ_r and

[2]The extended covariance matrix is composed of two components, one is an unsupervised covariance term and another is a constraint matrix involving labeled information.

[3]Hamming distance is defined as the number of bits that are different between two binary strings.

Algorithm 1: Feature Contribution Calculation

```
for each (x, c) ∈ X_L do
    for each f ∈ x do
        ν_f ← ν_f + x_f;
        ν_{f,c} ← ν_{f,c} + x_f;
    end
end
for each feature f and class c do
    ν_{f,c} ← ν_{f,c} / ν_f;
end
```

maximize the entropy of hash functions. So, we define the following objective for $\hbar(\cdot)$s:

$$J(\mathbf{w}) = \frac{1}{N_p}\sum_{l=1}^{L}\Big\{\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\Theta_a}\hbar_l(\mathbf{x}_i)\hbar_l(\mathbf{x}_j) - \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\Theta_r}\hbar_l(\mathbf{x}_i)\hbar_l(\mathbf{x}_j)\Big\} + \lambda_1\sum_{l=1}^{L}\mathrm{H}(\hbar_l) \tag{7}$$

where $N_p = |\Theta_a| + |\Theta_r|$ is the number of attraction and repulsion pairs and λ_1 is a tradeoff between two terms. Wang *et al.* have proven that hash functions with maximum entropy must maximize the variance of the hash values, and vice-versa (Wang *et al.*, 2010b). Thus, $\mathrm{H}(\hbar(\cdot))$ can be estimated over the labeled and unlabeled data, $\mathcal{X}_L$ and $\mathcal{X}_U$.

Unfortunately, direct solution for above problem is non-trivial since Equation (7) is not differentiable. Thus, we relax the objective and add an additional regularization term which could effectively avoid overfitting. Finally, we obtain the total objective:

$$\begin{aligned}\mathcal{L}(\mathbf{w}) = {} & \frac{1}{N_p}\sum_{l=1}^{L}\{\sum_{(\mathbf{G}_i,\mathbf{G}_j)\in\Theta_a}\psi(\mathbf{w}^T\mathbf{g}_{i,l})\psi(\mathbf{w}^T\mathbf{g}_{j,l}) \\ & - \sum_{(\mathbf{G}_i,\mathbf{G}_j)\in\Theta_r}\psi(\mathbf{w}^T\mathbf{g}_{i,l})\psi(\mathbf{w}^T\mathbf{g}_{j,l})\} \\ & + \frac{\lambda_1}{2N}\sum_{l=1}^{L}\{\sum_{i=1}^{u}\psi^2(\mathbf{w}^T\mathbf{g}_{i,l}) + \sum_{i=u+1}^{N}\psi^2(\mathbf{w}^T\mathbf{d}_{i,l})\} \\ & - \frac{\lambda_2}{2}\|\mathbf{w}\|_2^2\end{aligned} \tag{8}$$

where $\mathbf{g}_{i,l}$ and $\mathbf{d}_{i,l}$ denote the l-th column of $\mathbf{G}_i$ and $\mathbf{D}_i$ respectively, and $\psi(t)$ is a piece-wise linear function defined as:

$$\psi(t) = \begin{cases} T_g & t > T_g \\ t & -T_g \le t \le T_g \\ -T_g & t < -T_g \end{cases} \tag{9}$$

This relaxation has a good intuitive explanation. That is, similar objects are desired to not only have the similar fingerprints but also have sufficient large projection magnitudes, while dissimilar objects are desired to not only differ in their fingerprints but also have large projection margin. However, we do not hope that a small fraction of object-pairs with very large projection magnitude or margin dominate the complete model. Thus, a piece-wise linear function $\psi(\cdot)$ is applied in S^{3}H.

As a result, Equation (8) is a simply unconstrained optimization problem, which can be efficiently solved by a notable Quasi-Newton algorithm, i.e., L-BFGS (Liu *et al.*, 1989). For description simplicity, only attraction set Θ_a is considered and the extension to repulsion set Θ_r is straightforward. Thus, the gradient of $\mathcal{L}(\mathbf{w})$ is as follows:

$$\begin{aligned}\frac{\partial\mathcal{L}(\mathbf{w})}{\partial\mathbf{w}} = {} & \frac{1}{N_p}\sum_{l=1}^{L}\Big\{\sum_{\substack{(\mathbf{G}_i,\mathbf{G}_j)\in\Theta_a,\\ |\mathbf{w}^T\mathbf{g}_{i,l}|\le T_g}}\psi(\mathbf{w}^T\mathbf{g}_{j,l})\cdot\mathbf{g}_{i,l} \\ & + \sum_{\substack{(\mathbf{G}_i,\mathbf{G}_j)\in\Theta_a,\\ |\mathbf{w}^T\mathbf{g}_{j,l}|\le T_g}}\psi(\mathbf{w}^T\mathbf{g}_{i,l})\cdot\mathbf{g}_{j,l}\Big\} \\ & + \frac{\lambda_1}{N}\sum_{l=1}^{L}\Big\{\sum_{\substack{i=1,\\ |\mathbf{w}^T\mathbf{g}_{i,l}|\le T_g}}^{u}\psi(\mathbf{w}^T\mathbf{g}_{i,l})\cdot\mathbf{g}_{i,l} \\ & + \sum_{\substack{i=u+1,\\ |\mathbf{w}^T\mathbf{d}_{i,l}|\le T_g}}^{N}\psi(\mathbf{w}^T\mathbf{d}_{i,l})\cdot\mathbf{d}_{i,l}\Big\} - \lambda_2\mathbf{w}\end{aligned} \tag{10}$$

Note that $\partial\psi(t)/\partial t = 0$ when $|t| > T_g$.

3.3 Fingerprint Generation

When we get the optimal weight $\mathbf{w}^*$, we generate fingerprints for given objects through Equation (5). Then, it tunes to the problem how to efficiently obtain the representation as in Figure 4 for a object. After analysis, we find: 1) hyperplanes are randomly generated and we only need to determine which sides of these hyperplanes the given object lies on, and 2) in real-world applications, objects such as docs are always very sparse. Thus, we can avoid heavy computational demands and efficiently generate fingerprints for objects.

In practice, given an object $\mathbf{x}$, the procedure of generating an L-bit fingerprint is as follows: it maintains an L-dimensional vector initialized to zero. Each feature $f \in \mathbf{x}$ is firstly mapped to an L-bit hash value by *Jenkins Hashing Function*[4]. Then,

[4]http://www.burtleburtle.net/bob/hash/doobs.html

Algorithm 2: Fast Fingerprint Generation

INPUT: $\mathbf{x}$ and $\mathbf{w}^*$;
initialize $\boldsymbol{\alpha} \leftarrow 0, \boldsymbol{\beta} \leftarrow 0, \boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^L$;
for *each* $f \in \mathbf{x}$ **do**
  randomly project f to $\mathbf{h}_f \in \{-1, +1\}^L$;
  $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + x_f \cdot w_f^* \cdot \mathbf{h}_f$;
end
for $l = 1$ *to* L **do**
  if $\alpha_l > 0$ **then**
    $\beta_l \leftarrow 1$;
  end
end
RETURN $\boldsymbol{\beta}$;

these L bits increment or decrement the L components of the vector by the value $x_f \times w_f^*$. After all features processed, the signs of components determine the corresponding bits of the final fingerprint. The complete algorithm is presented in Algorithm 2.

3.4 Algorithmic Analysis

This section briefly analyzes the relation between S^{3}H and some existing methods. For analysis simplicity, we assume $\psi(t) = t$ and ignore the regularization terms. So, Equation (8) can be rewritten as follows:

$$J(\mathbf{w})_{S^3H} = \frac{1}{2}\mathbf{w}^T[\sum_{l=1}^{L}\boldsymbol{\Gamma}_l(\boldsymbol{\Phi}^+ - \boldsymbol{\Phi}^-)\boldsymbol{\Gamma}_l^T]\mathbf{w} \quad (11)$$

where Φ_{ij}^+ equals to 1 when $(\mathbf{x}_i, \mathbf{x}_j) \in \Theta_a$ otherwise 0, Φ_{ij}^- equals to 1 when $(\mathbf{x}_i, \mathbf{x}_j) \in \Theta_r$ otherwise 0, and $\boldsymbol{\Gamma}_l = [\mathbf{g}_{1,l} \dots \mathbf{g}_{u,l}, \mathbf{d}_{u+1,l} \dots \mathbf{d}_{N,l}]$. We denote $\sum_l \boldsymbol{\Gamma}_l\boldsymbol{\Phi}^+\boldsymbol{\Gamma}_l^T$ and $\sum_l \boldsymbol{\Gamma}_l\boldsymbol{\Phi}^-\boldsymbol{\Gamma}_l^T$ as $\mathbf{S}^+$ and $\mathbf{S}^-$ respectively. Therefore, maximizing above function is equivalent to maximizing the following:

$$\widetilde{J}(\mathbf{w})_{S^3H} = \frac{|\mathbf{w}^T\mathbf{S}^+\mathbf{w}|}{|\mathbf{w}^T\mathbf{S}^-\mathbf{w}|} \quad (12)$$

Clearly, Equation (12) is analogous to Linear Discriminant Analysis (LDA) (Duda *et al.*, 2000) except for the difference: 1) measurement. S^{3}H uses similarity while LDA uses distance. As a result, the objective function of S^{3}H is just the reciprocal of LDA's. 2) embedding space. LDA seeks the best separative direction in the original attribute space. In contrast, S^{3}H firstly maps data from $\mathbb{R}^M$ to $\mathbb{R}^{M\times L}$ through the following projection function

$$\phi(\mathbf{x}) = \mathbf{x} \cdot [\mathrm{diag}(\mathrm{sign}(\mathbf{r}_1)), \dots, \mathrm{diag}(\mathrm{sign}(\mathbf{r}_L))] \quad (13)$$

where $\mathbf{r}_l \in \mathbb{R}^M, l = 1, \dots, L$, are L random hyperplanes. Then, in that space ($\mathbb{R}^{M\times L}$), S^{3}H seeks a direction[5] that can best separate the data.

From this point of view, it is obvious that the basic SH is a special case of S^{3}H when $\mathbf{w}$ is set to $\mathbf{e} = [1, 1, \dots, 1]$. That is, SH firstly maps the data via $\phi(\cdot)$ just as S^{3}H. But then, SH directly separates the data in that feature space at the direction $\mathbf{e}$.

Analogously, we ignore the regularization terms in SSH and rewrite the objective of SSH as:

$$J(\mathbf{W})_{SSH} = \frac{1}{2}\,\mathrm{tr}[\mathbf{W}^T\mathbf{X}(\boldsymbol{\Phi}^+ - \boldsymbol{\Phi}^-)\mathbf{X}^T\mathbf{W}] \quad (14)$$

where $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_L] \in \mathbb{R}^{M\times L}$ are L hyperplanes and $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$. Maximizing this objective is equivalent to maximizing the following:

$$\widetilde{J}(\mathbf{W})_{SSH} = \frac{|\,\mathrm{tr}[\mathbf{W}^T\mathbf{S}'^+\mathbf{W}]|}{|\,\mathrm{tr}[\mathbf{W}^T\mathbf{S}'^-\mathbf{W}]|} \quad (15)$$

where $\mathbf{S}'^+ = \mathbf{X}\boldsymbol{\Phi}^+\mathbf{X}^T$ and $\mathbf{S}'^- = \mathbf{X}\boldsymbol{\Phi}^-\mathbf{X}^T$. Equation (15) shows that SSH is analogous to Multiple Discriminant Analysis (MDA) (Duda *et al.*, 2000). In fact, SSH uses top L best-separative hyperplanes in the original attribute space found via PCA to hash the data. Furthermore, we rewrite the projection function $\phi(\cdot)$ in S^{3}H as:

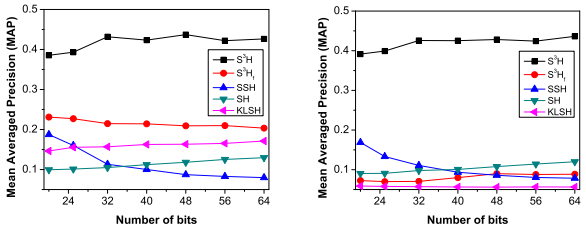
$$\phi(\mathbf{x}) = \mathbf{x} \cdot [\mathbf{R}_1, \dots, \mathbf{R}_L] \quad (16)$$

where $\mathbf{R}_l = \mathrm{diag}(\mathrm{sign}(\mathbf{r}_l))$. Each $\mathbf{R}_l$ is a mapping from $\mathbb{R}^M$ to $\mathbb{R}^M$ and corresponds to one embedding space. From this perspective, unlike SSH, S^{3}H globally seeks a direction that can best separate the data in L different embedding spaces simultaneously.

4 Experiments

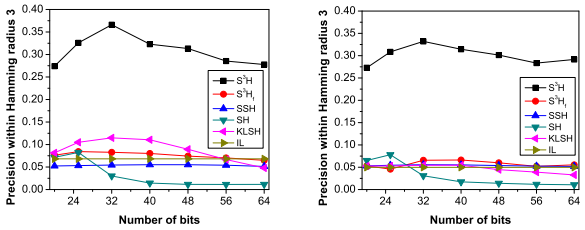
We use two datasets 20 Newsgroups and Open Directory Project (ODP) in our experiments. Each document is represented as a vector of occurrence numbers of the terms within it. The class information of docs is considered as prior knowledge that two docs within a same class should have more similar fingerprints while two docs within different classes should have dissimilar fingerprints. We will demonstrate that our S^{3}H can effectively incorporate this prior knowledge to improve the DSS performance.

[5] The direction is determined by concatenating $\mathbf{w}$ L times.



(a) (b)

Figure 1: Mean Averaged Precision (MAP) for different number of bits for hash ranking on 20 Newsgroups. (a) 10K features. (b) 30K features.



(a) (b)

Figure 2: Precision within Hamming radius 3 for hash lookup on 20 Newsgroups. (a) 10K features. (b) 30K features.

We use Inverted List (IL) (Manning *et al.*, 2002) as the baseline. In fact, given a query doc, IL returns all the docs that contain any term within it. We also compare our method with three state-of-the-art hashing methods, i.e., KLSH, SSH and SH. In KLSH, we adopt the RBF kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\delta^2})$, where the scaling factor δ^2 takes 0.5 and the other two parameters p and t are set to be 500 and 50 respectively. The parameter λ in SSH is set to 1. For S^{3}H, we simply set the parameters λ_1 and λ_2 in Equation (8) to 4 and 0.5 respectively. To objectively reflect the performance of S^{3}H, we evaluate our S^{3}H with and without Feature Contribution Calculation algorithm (FCC) (Algorithm 1). Specifically, FCC-free S^{3}H (denoted as S^{3}H$_f$) is just a simplification when $\mathbf{G}$s in S^{3}H are simply set to $\mathbf{D}$s.

For quantitative evaluation, as in literature (Wang *et al.*, 2010b; Mu *et al.*, 2009), we calculate the precision under two scenarios: hash lookup and hash ranking. For hash lookup, the proportion of good neighbors (have the same class label as the query) among the searched objects within a given Hamming radius is calculated as precision. Similarly to (Wang *et al.*, 2010b; Weiss *et al.*, 2009), for a query document, if no neighbors within the given Hamming radius can be found, it is considered as zero precision. Note that, the precision of IL is the proportion of good neighbors among the whole searched objects. For hash ranking, all the objects in $\mathcal{X}$ are ranked in terms of their Hamming distance from the query document, and the top K nearest neighbors are returned as the result. Then, Mean Averaged Precision (MAP) (Manning *et al.*, 2002) is calculated. We also calculate the averaged intra- and inter- class Hamming distance for various hashing methods. Intuitively, a good hashing method should have small intra-class distance while large inter-class distance.

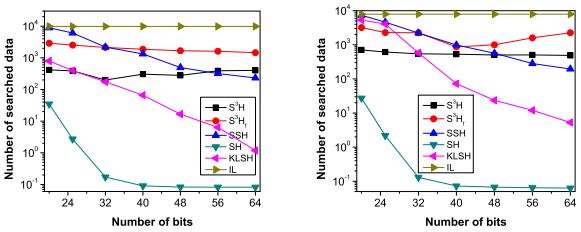
We test all the methods on a PC with a 2.66 GHz processor and 12GB RAM. All experiments repeate 10 times and the averaged results are reported.

4.1 20 Newsgroups

20 Newsgroups[6] contains 20K messages, about 1K messages from each of 20 different newsgroups. The entire vocabulary includes 62,061 words. To evaluate the performance for different feature dimensions, we use Chi-squared feature selection algorithm (Forman, 2003) to select 10K and 30K features. The averaged message length is 54.1 for 10K features and 116.2 for 30K features. We randomly select 4K massages as the test set and the remain 16K as the training set. To train SSH and S^{3}H, from the training set, we randomly generate 40K message-pairs as Θ_a and 80K message-pairs as Θ_r.

For hash ranking, Figure 1 shows MAP for various methods using different number of bits. It shows that performance of SSH decreases with the growing of hash bits. This is mainly because the variance of the directions obtained by PCA decreases with the decrease of their ranks. Thus, lower bits have larger empirical errors. For S^{3}H, FCC (Algorithm 1) can significantly improve the MAP just as discussed in Section 3.2. Moreover, the MAP of FCC-free S^{3}H (S^{3}H$_f$) is affected by feature dimensions while FCC-based (S^{3}H) is relatively stable. This implies FCC can also improve the satiability of S^{3}H. As we see, S^{3}H$_f$ ignores the contribution of features to different classes. However, besides the local description of data locality in the form of object-pairs, such

[6] http://www.cs.cmu.edu/afs/cs/project/theo-3/www/



(a) (b)

Figure 3: Averaged searched sample numbers using 4K query messages for hash lookup. (a) 10K features. (b) 30K features.

(global) information also provides a proper guidance for hashing. So, for S^3H_f, the reason why its results with 30K features are worse than the results with 10K features is probably because S^3H_f learns to hash only according to the local description of data locality and many not too relevant features lead to relatively poor description. In contrast, S^3H can utilize global information to better understand the similarity among objects. In short, S^3H obtains the best MAP for all bits and feature dimensions.

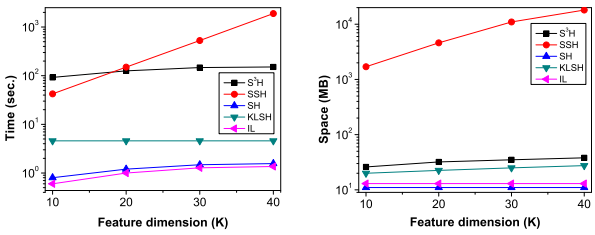
For hash lookup, Figure 2 presents the precision within Hamming radius 3 for different number of bits. It shows that IL even outperforms SH. This is because few objects can be hashed by SH into one hash bucket. Thus, for many queries, SH fails to return any neighbor even in a large Hamming radius of 3. Clearly, S^3H outperforms all the other methods for different number of hash bits and features.

The number of messages searched by different methods are reported in Figure 3. We find that the number of searched data of S^3H (with/without FCC) decreases much more slowly than KLSH, SH and SSH with the growing of the number of hash bits. As discussed in Section 3.4, this mainly benefits from the design of S^3H that S^3H (globally) seeks a direction that can best separate the data in L embedding spaces simultaneously. We also find IL returns a large number of neighbors of each query message which leads to its poor efficiency.

The averaged intra- and inter- class Hamming distance of different methods are reported in Table 1. As it shows, S^3H has relatively larger margin (Δ) between intra- and inter-class Hamming distance. This indicates that S^3H is more effective to make similar points have similar fingerprints while keep the dissimilar points away enough from each other.

	intra-class	inter-class	Δ
S^3H	13.1264	15.6342	**2.5078**
S^3H_f	12.5754	13.3479	0.7725
SSH	6.4134	6.5262	0.1128
SH	15.3908	15.6339	0.2431
KLSH	10.2876	10.8713	0.5841

Table 1: Averaged intra- and inter- class Hamming distance of 20 Newsgroups for 32-bit fingerprint. Δ is the difference between the averaged inter- and intra- class Hamming distance. Large Δ implies good hashing.



(a) (b)

Figure 4: Computational complexity of training for different feature dimensions for 32-bit fingerprint. (a) Training time (sec). (b) Training space cost (MB).

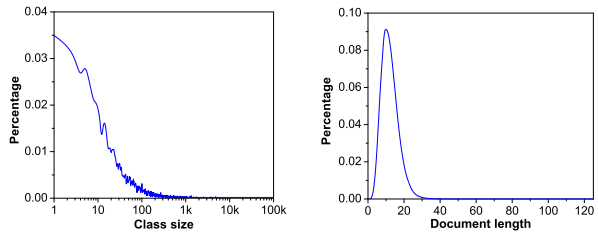
Figure 4 shows the (training) computational complexity of different methods. We find that the time and space cost of SSH grows much faster than SH, KLSH and S^3H with the growing of feature dimension. This is mainly because SSH requires SVD to find the optimal hashing functions which is computational expensive. Instead, S^3H seeks the optimal feature weights via L-BFGS, which is still efficient even for very high-dimensional data.

4.2 Open Directory Project (ODP)

Open Directory Project (ODP)[7] is a multilingual open content directory of web links (docs) organized by a hierarchical ontology scheme. In our experiment, only English docs[8] at level 3 of the category tree are utilized to evaluate the performance. In short, the dataset contains 2,483,388 docs within 6,008 classes. There are totally 862,050 distinct words and each doc contains 14.13 terms on average. Since docs are too short, we do not conduct

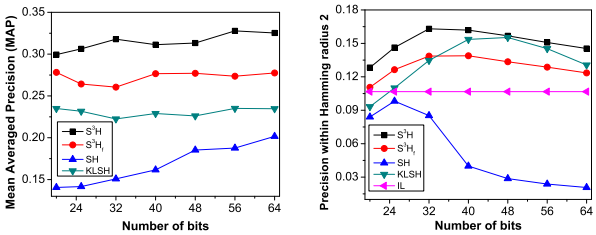
[7] http://rdf.dmoz.org/

[8] The title together with the corresponding short description of a page are considered as a document in our experiments.



(a) (b)

Figure 5: Overview of ODP data set. (a) Class distribution at level 3. (b) Distribution of document length.



(a) (b)

Figure 6: Retrieval performance of different methods on ODP. (a) Mean Averaged Precision (MAP) for different number of bits for hash ranking. (b) Precision within Hamming radius 2 for hash lookup.

	intra-class	inter-class	Δ
S^3H	14.0029	15.9508	**1.9479**
S^3H_f	14.3801	15.5260	1.1459
SH	14.7725	15.6432	0.8707
KLSH	9.3382	10.5700	1.2328

Table 2: Averaged intra- and inter- class Hamming distance of ODP for 32-bit fingerprint (860K features). Δ is the difference between averaged intra- and inter- class Hamming distance.

feature selection[9]. An overview of ODP is shown in Figure 5. We randomly sample 10% docs as the test set and the remain as the training set. Furthermore, from training set, we randomly generate 800K doc-pairs as Θ_a, and 1 million doc-pairs as Θ_r. Note that, since there are totally over 800K features, it is extremely inefficient to train SSH. Therefore, we only compare our S^3H with IL, KLSH and SH.

The search performance is given in Figure 6. Figure 6(a) shows the MAP for various methods using different number of bits. It shows KLSH outperforms SH, which mainly contributes to the kernel trick. S^3H and S^3H_f have higher MAP than KLSH and SH. Clearly, FCC algorithm can improve the MAP of S^3H for all bits. Figure 6(b) presents the precision within Hamming radius 2 for hash lookup. We find that IL outperforms SH since SH fails for many queries. It also shows that S^3H (with FCC) can obtain the best precision for all bits.

Table 2 reports the averaged intra- and inter-class Hamming distance for various methods. It shows that S^3H has the largest margin (Δ). This demonstrates S^3H can measure the similarity among the data better than KLSH and SH.

We should emphasize that KLSH needs 0.3ms to return the results for a query document for hash lookup, and S^3H needs <0.1ms. In contrast, IL requires about 75ms to finish searching. This is mainly because IL always returns a large number of objects (dozens or hundreds times more than S^3H and KLSH) and requires much time for post-processing.

All the experiments show S^3H is more effective, efficient and stable than the baseline method and the state-of-the-art hashing methods.

5 Conclusions

We have proposed a novel supervised hashing method named Semi-Supervised Simhash (S^3H) for high-dimensional data similarity search. S^3H learns the optimal feature weights from prior knowledge to relocate the data such that similar objects have similar fingerprints. This is implemented by maximizing the empirical accuracy on labeled data together with the entropy of hash functions. The proposed method leads to a simple Quasi-Newton based solution which is efficient even for very high-dimensional data. Experiments performed on two large datasets have shown that S^3H has better search performance than several state-of-the-art methods.

6 Acknowledgements

We thank Fangtao Li for his insightful suggestions. We would also like to thank the anonymous reviewers for their helpful comments. This work is supported by the National Natural Science Foundation of China under Grant No. 60873174.

[9]We have tested feature selection. However, if we select 40K features via Chi-squared feature selection method, documents are represented by 3.15 terms on average. About 44.9% documents are represented by no more than 2 terms.

References

Christopher J.C. Burges. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121-167.

Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th annual ACM symposium on Theory of computing*, pages 380-388.

Gianni Costa, Giuseppe Manco and Riccardo Ortale. 2010. An incremental clustering scheme for data deduplication. *Data Mining and Knowledge Discovery*, 20(1):152-187.

Jeffrey Dean and Monika R. Henzinge. 1999. Finding Related Pages in the World Wide Web. *Computer Networks*, 31:1467-1479.

Richard O. Duda, Peter E. Hart and David G. Stork. 2000. Pattern classification, 2nd edition. Wiley-Interscience.

George Forman 2003. An extensive empirical study of feature selection metrics for text classification. *The Journal of Machine Learning Research*, 3:1289-1305.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the 30th annual ACM symposium on Theory of computing*, pages 604-613.

Ian Jolliffe. 1986. Principal Component Analysis. *Springer-Verlag*, New York.

Yan Ke, Rahul Sukthankar and Larry Huston. 2004. Efficient near-duplicate detection and sub-image retrieval. In *Proceedings of the ACM International Conference on Multimedia*.

Brian Kulis and Kristen Grauman. 2009. Kernelized locality-sensitive hashing for scalable image search. In *Proceedings of the 12th International Conference on Computer Vision*, pages 2130-2137.

Brian Kulis, Prateek Jain and Kristen Grauman. 2009. Fast similarity search for learned metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 2143-2157.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1): 503-528.

Omid Madani, Michael Connor and Wiley Greiner. 2009. Learning when concepts abound. *The Journal of Machine Learning Research*, 10:2571-2613.

Gurmeet Singh Manku, Arvind Jain and Anish Das Sarma. 2007. Detecting near-duplicates for web crawling. In *Proceedings of the 16th international conference on World Wide Web*, pages 141-150.

Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze. 2002. An introduction to information retrieval. *Spring*.

Yadong Mu, Jialie Shen and Shuicheng Yan. 2010. Weakly-Supervised Hashing in Kernel Space. In *Proceedings of International Conference on Computer Vision and Pattern Recognition*, pages 3344-3351.

Ruslan Salakhutdinov and Geoffrey Hintona. 2007. Semantic hashing. In *SIGIR workshop on Information Retrieval and applications of Graphical Models*.

Bernhard Schölkopf, Alexander Smola and Klaus-Robert Müller. 1997. Kernel principal component analysis. *Advances in Kernel Methods - Support Vector Learning*, pages 583-588. MIT.

Lloyd N. Trefethen and David Bau. 1997. Numerical linear algebra. *Society for Industrial Mathematics*.

Xiaojun Wan, Jianwu Yang and Jianguo Xiao. 2008. Towards a unified approach to document similarity search using manifold-ranking of blocks. *Information Processing & Management*, 44(3):1032-1048.

Jun Wang, Sanjiv Kumar and Shih-Fu Chang. 2010a. Semi-Supervised Hashing for Scalable Image Retrieval. In *Proceedings of International Conference on Computer Vision and Pattern Recognition*, pages 3424-3431.

Jun Wang, Sanjiv Kumar and Shih-Fu Chang. 2010b. Sequential Projection Learning for Hashing with Compact Codes. In *Proceedings of International Conference on Machine Learning*.

Yair Weiss, Antonio Torralba and Rob Fergus. 2009. Spectral hashing. In *Proceedings of Advances in Neural Information Processing Systems*.

Joint Annotation of Search Queries

Michael Bendersky
Dept. of Computer Science
University of Massachusetts
Amherst, MA
bemike@cs.umass.edu

W. Bruce Croft
Dept. of Computer Science
University of Massachusetts
Amherst, MA
croft@cs.umass.edu

David A. Smith
Dept. of Computer Science
University of Massachusetts
Amherst, MA
dasmith@cs.umass.edu

Abstract

Marking up search queries with linguistic annotations such as part-of-speech tags, capitalization, and segmentation, is an important part of query processing and understanding in information retrieval systems. Due to their brevity and idiosyncratic structure, search queries pose a challenge to existing NLP tools. To address this challenge, we propose a probabilistic approach for performing joint query annotation. First, we derive a robust set of unsupervised independent annotations, using queries and pseudo-relevance feedback. Then, we stack additional classifiers on the independent annotations, and exploit the dependencies between them to further improve the accuracy, even with a very limited amount of available training data. We evaluate our method using a range of queries extracted from a web search log. Experimental results verify the effectiveness of our approach for both short keyword queries, and verbose natural language queries.

1 Introduction

Automatic mark-up of textual documents with linguistic annotations such as part-of-speech tags, sentence constituents, named entities, or semantic roles is a common practice in natural language processing (NLP). It is, however, much less common in information retrieval (IR) applications. Accordingly, in this paper, we focus on annotating search queries submitted by the users to a search engine.

There are several key differences between user queries and the documents used in NLP (e.g., news articles or web pages). As previous research shows, these differences severely limit the applicability of standard NLP techniques for annotating queries and require development of novel annotation approaches for query corpora (Bergsma and Wang, 2007; Barr et al., 2008; Lu et al., 2009; Bendersky et al., 2010; Li, 2010).

The most salient difference between queries and documents is their length. Most search queries are very short, and even longer queries are usually shorter than the average written sentence. Due to their brevity, queries often cannot be divided into sub-parts, and do not provide enough context for accurate annotations to be made using the standard NLP tools such as taggers, parsers or chunkers, which are trained on more syntactically coherent textual units.

A recent analysis of web query logs by Bendersky and Croft (2009) shows, however, that despite their brevity, queries are grammatically diverse. Some queries are keyword concatenations, some are semi-complete verbal phrases and some are wh-questions. It is essential for the search engine to correctly annotate the query structure, and the quality of these query annotations has been shown to be a crucial first step towards the development of reliable and robust query processing, representation and understanding algorithms (Barr et al., 2008; Guo et al., 2008; Guo et al., 2009; Manshadi and Li, 2009; Li, 2010).

However, in current query annotation systems, even sentence-like queries are often hard to parse and annotate, as they are prone to contain misspellings and idiosyncratic grammatical structures.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 102–111,
Portland, Oregon, June 19-24, 2011.

(a)

Term	CAP	TAG	SEG
who	L	X	B
won	L	V	I
the	L	X	B
2004	L	X	B
kentucky	C	N	B
derby	C	N	I

(b)

Term	CAP	TAG	SEG
kindred	C	N	B
where	C	X	B
would	C	X	I
i	C	X	I
be	C	V	I

(c)

Term	CAP	TAG	SEG
shih	C	N	B
tzu	C	N	I
health	L	N	B
problems	L	N	I

Figure 1: **E**xamples of a mark-up scheme for annotating capitalization (L – lowercase, C – otherwise), POS tags (N – noun, V – verb, X – otherwise) and segmentation (B/I – beginning of/inside the chunk).

They also tend to lack prepositions, proper punctuation, or capitalization, since users (often correctly) assume that these features are disregarded by the retrieval system.

In this paper, we propose a novel joint query annotation method to improve the effectiveness of existing query annotations, especially for longer, more complex search queries. Most existing research focuses on using a single type of annotation for information retrieval such as subject-verb-object dependencies (Balasubramanian and Allan, 2009), named-entity recognition (Guo et al., 2009), phrase chunking (Guo et al., 2008), or semantic labeling (Li, 2010).

In contrast, the main focus of this work is on developing a unified approach for performing reliable annotations of different types. To this end, we propose a probabilistic method for performing a *joint query annotation*. This method allows us to exploit the dependency between different unsupervised annotations to further improve the accuracy of the entire set of annotations. For instance, our method can leverage the information about estimated parts-of-speech tags and capitalization of query terms to improve the accuracy of query segmentation.

We empirically evaluate the joint query annotation method on a range of query types. Instead of just focusing our attention on keyword queries, as is often done in previous work (Barr et al., 2008; Bergsma and Wang, 2007; Tan and Peng, 2008; Guo et al., 2008), we also explore the performance of our annotations with more complex natural language search queries such as verbal phrases and wh-questions, which often pose a challenge for IR applications (Bendersky et al., 2010; Kumaran and Allan, 2007; Kumaran and Carvalho, 2009; Lease, 2007). We show that even with a very limited amount of training data, our joint annotation method significantly outperforms annotations that were done independently for these queries.

The rest of the paper is organized as follows. In Section 2 we demonstrate several examples of annotated search queries. Then, in Section 3, we introduce our joint query annotation method. In Section 4 we describe two types of independent query annotations that are used as input for the joint query annotation. Section 5 details the related work and Section 6 presents the experimental results. We draw the conclusions from our work in Section 7.

2 Query Annotation Example

To demonstrate a possible implementation of linguistic annotation for search queries, Figure 1 presents a simple mark-up scheme, exemplified using three web search queries (as they appear in a search log): (a) *who won the 2004 kentucky derby*, (b) *kindred where would i be*, and (c) *shih tzu health problems*. In this scheme, each query is marked-up using three annotations: capitalization, POS tags, and segmentation indicators.

Note that all the query terms are non-capitalized, and no punctuation is provided by the user, which complicates the query annotation process. While the simple annotation described in Figure 1 can be done with a very high accuracy for standard document corpora, both previous work (Barr et al., 2008; Bergsma and Wang, 2007; Jones and Fain, 2003) and the experimental results in this paper indicate that it is challenging to perform well on queries.

The queries in Figure 1 illustrate this point. Query (a) in Figure 1 is a wh-question, and it contains

a capitalized concept (*"Kentucky Derby"*), a single verb, and four segments. Query (b) is a combination of an artist name and a song title and should be interpreted as *Kindred — "Where Would I Be"*. Query (c) is a concatenation of two short noun phrases: *"Shih Tzu"* and *"health problems"*.

3 Joint Query Annotation

Given a search query Q, which consists of a sequence of terms $(q_1, \ldots, q_n)$, our goal is to annotate it with an appropriate set of linguistic structures $\mathcal{Z}_Q$. In this work, we assume that the set $\mathcal{Z}_Q$ consists of *shallow* sequence annotations $\mathbf{z}_Q$, each of which takes the form

$$\mathbf{z}_Q = (\zeta_1, \ldots, \zeta_n).$$

In other words, each symbol $\zeta_i \in \mathbf{z}_Q$ annotates a single query term.

Many query annotations that are useful for IR can be represented using this simple form, including capitalization, POS tagging, phrase chunking, named entity recognition, and stopword indicators, to name just a few. For instance, Figure 1 demonstrates an example of a set of annotations $\mathcal{Z}_Q$. In this example,

$$\mathcal{Z}_Q = \{\mathbf{CAP}, \mathbf{TAG}, \mathbf{SEG}\}.$$

Most previous work on query annotation makes the independence assumption — every annotation $\mathbf{z}_Q \in \mathcal{Z}_Q$ is done separately from the others. That is, it is assumed that the optimal linguistic annotation $\mathbf{z}_Q^{*(I)}$ is the annotation that has the highest probability given the query Q, regardless of the other annotations in the set $\mathcal{Z}_Q$. Formally,

$$\mathbf{z}_Q^{*(I)} = \operatorname*{argmax}_{\mathbf{z}_Q} p(\mathbf{z}_Q|Q) \qquad (1)$$

The main shortcoming of this approach is in the assumption that the linguistic annotations in the set $\mathcal{Z}_Q$ are independent. In practice, there are dependencies between the different annotations, and they can be leveraged to derive a better estimate of the entire set of annotations.

For instance, imagine that we need to perform two annotations: capitalization and POS tagging. Knowing that a query term is capitalized, we are more likely to decide that it is a proper noun. Vice versa, knowing that it is a preposition will reduce its probability of being capitalized. We would like to capture this intuition in the annotation process.

To address the problem of joint query annotation, we first assume that we have an initial set of annotations $\mathcal{Z}_Q^{*(I)}$, which were performed for query Q independently of one another (we will show an example of how to derive such a set in Section 4). Given the initial set $\mathcal{Z}_Q^{*(I)}$, we are interested in obtaining an annotation set $\mathcal{Z}_Q^{*(J)}$, which jointly optimizes the probability of *all* the annotations, i.e.

$$\mathcal{Z}_Q^{*(J)} = \operatorname*{argmax}_{\mathcal{Z}_Q} p(\mathcal{Z}_Q|\mathcal{Z}_Q^{*(I)}).$$

If the initial set of estimations is reasonably accurate, we can make the assumption that the annotations in the set $\mathcal{Z}_Q^{*(J)}$ are independent given the initial estimates $\mathcal{Z}_Q^{*(I)}$, allowing us to separately optimize the probability of each annotation $\mathbf{z}_Q^{*(J)} \in \mathcal{Z}_Q^{*(J)}$:

$$\mathbf{z}_Q^{*(J)} = \operatorname*{argmax}_{\mathbf{z}_Q} p(\mathbf{z}_Q|\mathcal{Z}_Q^{*(I)}). \qquad (2)$$

From Eq. 2, it is evident that the joint annotation task becomes that of finding some optimal unobserved sequence (annotation $\mathbf{z}_Q^{*(J)}$), given the observed sequences (independent annotation set $\mathcal{Z}_Q^{*(I)}$).

Accordingly, we can directly use a supervised sequential probabilistic model such as CRF (Lafferty et al., 2001) to find the optimal $\mathbf{z}_Q^{*(J)}$. In this CRF model, the optimal annotation $\mathbf{z}_Q^{*(J)}$ is the *label* we are trying to predict, and the set of independent annotations $\mathcal{Z}_Q^{*(I)}$ is used as the basis for the *features* used for prediction. Figure 2 outlines the algorithm for performing the joint query annotation.

As input, the algorithm receives a training set of queries and their ground truth annotations. It then produces a set of independent annotation estimates, which are jointly used, together with the ground truth annotations, to learn a CRF model for each annotation type. Finally, these CRF models are used to predict annotations on a held-out set of queries, which are the output of the algorithm.

Input:	$\mathbf{Q}_t$ — training set of queries. $\mathcal{Z}_{\mathbf{Q}_t}$ — ground truth annotations for the training set of queries. $\mathbf{Q}_h$ — held-out set of queries.
(1)	Obtain a set of independent annotation estimates $\mathcal{Z}^{*(I)}_{\mathbf{Q}_t}$
(2)	Initialize $\mathcal{Z}^{*(J)}_{\mathbf{Q}_t} \leftarrow \emptyset$
(3)	*for each* $\mathbf{z}^{*(I)}_{\mathbf{Q}_t} \in \mathcal{Z}^{*(I)}_{\mathbf{Q}_t}$:
(4)	$\mathcal{Z}'_{\mathbf{Q}_t} \leftarrow \mathcal{Z}^{*(I)}_{\mathbf{Q}_t} \setminus \mathbf{z}^{*(I)}_{\mathbf{Q}_t}$
(5)	Train a CRF model $\mathcal{CRF}(\mathbf{z}_{\mathbf{Q}_t})$ using $\mathbf{z}_{\mathbf{Q}_t}$ as a *label* and $\mathcal{Z}'_{\mathbf{Q}_t}$ as *features*.
(6)	Predict annotation $\mathbf{z}^{*(J)}_{\mathbf{Q}_h}$, using $\mathcal{CRF}(\mathbf{z}_{\mathbf{Q}_t})$.
(7)	$\mathcal{Z}^{*(J)}_{\mathbf{Q}_h} \leftarrow \mathcal{Z}^{*(J)}_{\mathbf{Q}_h} \cup \mathbf{z}^{*(J)}_{\mathbf{Q}_h}$.
Output:	$\mathcal{Z}^{*(J)}_{\mathbf{Q}_h}$ — predicted annotations for the held-out set of queries.

Figure 2: **A**lgorithm for performing joint query annotation.

Note that this formulation of joint query annotation can be viewed as a *stacked classification*, in which a second, more effective, classifier is trained using the labels inferred by the first classifier as features. Stacked classifiers were recently shown to be an efficient and effective strategy for structured classification in NLP (Nivre and McDonald, 2008; Martins et al., 2008).

4 Independent Query Annotations

While the joint annotation method proposed in Section 3 is general enough to be applied to any set of independent query annotations, in this work we focus on two previously proposed independent annotation methods based on either the query itself, or the top sentences retrieved in response to the query (Bendersky et al., 2010). The main benefits of these two annotation methods are that they can be easily implemented using standard software tools, do not require any labeled data, and provide reasonable annotation accuracy. Next, we briefly describe these two independent annotation methods.

4.1 Query-based estimation

The most straightforward way to estimate the conditional probabilities in Eq. 1 is using the query itself. To make the estimation feasible, Bendersky et al. (2010) take a *bag-of-words* approach, and assume independence between both the query terms and the corresponding annotation symbols. Thus, the indepentent annotations in Eq. 1 are given by

$$\mathbf{z}^{*(QRY)}_{Q} = \underset{(\zeta_1,\ldots,\zeta_n)}{\operatorname{argmax}} \prod_{i\in(1,\ldots,n)} p(\zeta_i|q_i). \qquad (3)$$

Following Bendersky et al. (2010) we use a large n-gram corpus (Brants and Franz, 2006) to estimate $p(\zeta_i|q_i)$ for annotating the query with capitalization and segmentation mark-up, and a standard POS tagger[1] for part-of-speech tagging of the query.

4.2 PRF-based estimation

Given a short, often ungrammatical query, it is hard to accurately estimate the conditional probability in Eq. 1 using the query terms alone. For instance, a keyword query *hawaiian falls*, which refers to a location, is inaccurately interpreted by a standard POS tagger as a *noun-verb* pair. On the other hand, given a sentence from a corpus that is relevant to the query such as *"Hawaiian Falls is a family-friendly waterpark"*, the word "falls" is correctly identified by a standard POS tagger as a proper noun.

Accordingly, the document corpus can be bootstrapped in order to better estimate the query annotation. To this end, Bendersky et al. (2010) employ the *pseudo-relevance feedback* (PRF) — a method that has a long record of success in IR for tasks such as query expansion (Buckley, 1995; Lavrenko and Croft, 2001).

In the most general form, given the set of *all* retrievable sentences r in the corpus $\mathcal{C}$ one can derive

$$p(\mathbf{z}_Q|Q) = \sum_{r\in\mathcal{C}} p(\mathbf{z}_Q|r)p(r|Q).$$

Since for most sentences the conditional probability of relevance to the query $p(r|Q)$ is vanishingly small, the above can be closely approximated

[1] `http://crftagger.sourceforge.net/`

by considering only a set of sentences R, retrieved at top-k positions in response to the query Q. This yields

$$p(\mathbf{z}_Q|Q) \approx \sum_{r \in R} p(\mathbf{z}_Q|r)p(r|Q).$$

Intuitively, the equation above models the query as a mixture of top-k retrieved sentences, where each sentence is weighted by its relevance to the query. Furthermore, to make the estimation of the conditional probability $p(\mathbf{z}_Q|r)$ feasible, it is assumed that the symbols ζ_i in the annotation sequence are independent, given a sentence r. Note that this assumption differs from the independence assumption in Eq. 3, since here the annotation symbols are *not independent* given the query Q.

Accordingly, the PRF-based estimate for independent annotations in Eq. 1 is

$$\mathbf{z}_Q^{*(PRF)} = \underset{(\zeta_1,\ldots,\zeta_n)}{\operatorname{argmax}} \sum_{r \in R} \prod_{i \in (1,\ldots,n)} p(\zeta_i|r)p(r|Q). \quad (4)$$

Following Bendersky et al. (2010), an estimate of $p(\zeta_i|r)$ is a smoothed estimator that combines the information from the retrieved sentence r with the information about unigrams (for capitalization and POS tagging) and bigrams (for segmentation) from a large n-gram corpus (Brants and Franz, 2006).

5 Related Work

In recent years, linguistic annotation of search queries has been receiving increasing attention as an important step toward better query processing and understanding. The literature on query annotation includes query segmentation (Bergsma and Wang, 2007; Jones et al., 2006; Guo et al., 2008; Hagen et al., 2010; Hagen et al., 2011; Tan and Peng, 2008), part-of-speech and semantic tagging (Barr et al., 2008; Manshadi and Li, 2009; Li, 2010), named-entity recognition (Guo et al., 2009; Lu et al., 2009; Shen et al., 2008; Paşca, 2007), abbreviation disambiguation (Wei et al., 2008) and stopword detection (Lo et al., 2005; Jones and Fain, 2003).

Most of the previous work on query annotation focuses on performing a particular annotation task (e.g., segmentation or POS tagging) in isolation. However, these annotations are often related, and thus we take a joint annotation approach, which combines several independent annotations to improve the overall annotation accuracy. A similar approach was recently proposed by Guo et al. (2008). There are several key differences, however, between the work presented here and their work.

First, Guo et al. (2008) focus on *query refinement* (spelling corrections, word splitting, etc.) of short keyword queries. Instead, we are interested in *annotation* of queries of different types, including verbose natural language queries. While there is an overlap between query refinement and annotation, the focus of the latter is on providing linguistic information about existing queries (after initial refinement has been performed). Such information is especially important for more verbose and gramatically complex queries. In addition, while all the methods proposed by Guo et al. (2008) require large amounts of training data (thousands of training examples), our joint annotation method can be effectively trained with a minimal human labeling effort (several hundred training examples).

An additional research area which is relevant to this paper is the work on joint structure modeling (Finkel and Manning, 2009; Toutanova et al., 2008) and stacked classification (Nivre and McDonald, 2008; Martins et al., 2008) in natural language processing. These approaches have been shown to be successful for tasks such as parsing and named entity recognition in newswire data (Finkel and Manning, 2009) or semantic role labeling in the Penn Treebank and Brown corpus (Toutanova et al., 2008). Similarly to this work in NLP, we demonstrate that a joint approach for modeling the linguistic query structure can also be beneficial for IR applications.

6 Experiments

6.1 Experimental Setup

For evaluating the performance of our query annotation methods, we use a random sample of 250 queries[2] from a search log. This sample is manually labeled with three annotations: *capitalization*, *POS tags*, and *segmentation*, according to the description of these annotations in Figure 1. In this set of 250 queries, there are 93 questions, 96 phrases contain-

[2]The annotations are available at `http://ciir.cs.umass.edu/~bemike/data.html`

CAP		
	F1 *(% impr)*	MQA *(% impr)*
i-QRY	0.641 *(-/-)*	0.779 *(-/-)*
i-PRF	0.711^{*}*(+10.9/-)*	0.811^{*}*(+4.1/-)*
j-QRY	$0.620_{\dagger}$*(-3.3/-12.8)*	0.805^{*}*(+3.3/-0.7)*
j-PRF	**0.718****(+12.0/+0.9)*	**0.840**$^{*}_{\dagger}$*(+7.8/+3.6)*
TAG		
	Acc. *(% impr)*	MQA *(% impr)*
i-QRY	0.893 *(-/-)*	0.878 *(-/-)*
i-PRF	0.916^{*}*(+2.6/-)*	0.914^{*}*(+4.1/-)*
j-QRY	0.913^{*}*(+2.2/-0.3)*	0.912^{*}*(+3.9/-0.2)*
j-PRF	**0.924****(+3.5/+0.9)*	**0.922****(+5.0/+0.9)*
SEG		
	F1 *(% impr)*	MQA *(% impr)*
i-QRY	0.694 *(-/-)*	0.672 *(-/-)*
i-PRF	0.753^{*}*(+8.5/-)*	0.710^{*}*(+5.7/-)*
j-QRY	$0.817^{*}_{\dagger}$*(+17.7/+8.5)*	**0.803**$^{*}_{\dagger}$*(+19.5/+13.1)*
j-PRF	**0.819**$^{*}_{\dagger}$*(+18.0/+8.8)*	**0.803**$^{*}_{\dagger}$*(+19.5/+13.1)*

Table 1: **S**ummary of query annotation performance for capitalization (CAP), POS tagging (TAG) and segmentation. Numbers in parentheses indicate % of improvement over the *i-QRY* and *i-PRF* baselines, respectively. Best result per measure and annotation is boldfaced. * and $_{\dagger}$ denote statistically significant differences with *i-QRY* and *i-PRF*, respectively.

ing a verb, and 61 short keyword queries (Figure 1 contains a single example of each of these types).

In order to test the effectiveness of the joint query annotation, we compare four methods. In the first two methods, *i-QRY* and *i-PRF* the three annotations are done independently. Method *i-QRY* is based on $\mathbf{z}_Q^{*(QRY)}$ estimator (Eq. 3). Method *i-PRF* is based on the $\mathbf{z}_Q^{*(PRF)}$ estimator (Eq. 4).

The next two methods, *j-QRY* and *j-PRF*, are joint annotation methods, which perform a joint optimization over the entire set of annotations, as described in the algorithm in Figure 2. *j-QRY* and *j-PRF* differ in their choice of the initial independent annotation set $\mathcal{Z}_Q^{*(I)}$ in line (1) of the algorithm (see Figure 2). *j-QRY* uses only the annotations performed by *i-QRY* (3 initial independent annotation estimates), while *j-PRF* combines the annotations performed by *i-QRY* with the annotations performed by *i-PRF* (6 initial annotation estimates). The CRF model training in line (6) of the algorithm is implemented using CRF++ toolkit[3].

[3] `http://crfpp.sourceforge.net/`

The performance of the joint annotation methods is estimated using a 10-fold cross-validation. In order to test the statistical significance of improvements attained by the proposed methods we use a two-sided Fisher's randomization test with 20,000 permutations. Results with p-value < 0.05 are considered statistically significant.

For reporting the performance of our methods we use two measures. The first measure is classification-oriented — treating the annotation decision for each query term as a classification. In case of capitalization and segmentation annotations these decisions are binary and we compute the precision and recall metrics, and report F1 — their harmonic mean. In case of POS tagging, the decisions are ternary, and hence we report the classification accuracy.

We also report an additional, IR-oriented performance measure. As is typical in IR, we propose measuring the performance of the annotation methods on a per-query basis, to verify that the methods have uniform impact across queries. Accordingly, we report the *mean of classification accuracies per query* (MQA). Formally, MQA is computed as

$$\frac{\sum_{i=1}^{N} acc_{Q_i}}{N},$$

where acc_{Q_i} is the classification accuracy for query Q_i, and N is the number of queries.

The empirical evaluation is conducted as follows. In Section 6.2, we discuss the general performance of the four annotation techniques, and compare the effectiveness of independent and joint annotations. In Section 6.3, we analyze the performance of the independent and joint annotation methods by query type. In Section 6.4, we compare the difficulty of performing query annotations for different query types. Finally, in Section 6.5, we compare the effectiveness of the proposed joint annotation for query segmentation with the existing query segmentation methods.

6.2 General Evaluation

Table 1 shows the summary of the performance of the two independent and two joint annotation methods for the entire set of 250 queries. For independent methods, we see that *i-PRF* outperforms *i-QRY* for

CAP	*Verbal Phrases*		*Questions*		*Keywords*	
	F1	MQA	F1	MQA	F1	MQA
i-PRF	**0.750**	**0.862**	0.590	0.839	0.784	0.687
j-PRF	0.687*(-8.4%)	0.839*(-2.7%)	**0.671***(+13.7%)	**0.913***(+8.8%)	**0.814** (+3.8%)	**0.732*** (+6.6%)

TAG	*Verbal Phrases*		*Questions*		*Keywords*	
	Acc.	MQA	Acc.	MQA	Acc.	MQA
i-PRF	**0.908**	**0.908**	0.932	0.935	0.880	0.890
j-PRF	0.904 (-0.4%)	0.906 (-0.2%)	**0.951*** (+2.1%)	**0.953*** (+1.9%)	**0.893** (+1.5%)	**0.900** (+1.1%)

SEG	*Verbal Phrases*		*Questions*		*Keywords*	
	F1	MQA	F1	MQA	F1	MQA
i-PRF	0.751	0.700	0.740	0.700	0.816	0.747
j-PRF	**0.772** (+2.8%)	**0.742***(+6.0%)	**0.858***(+15.9%)	**0.838***(+19.7%)	**0.844** (+3.4%)	**0.853***(+14.2%)

Table 2: **D**etailed analysis of the query annotation performance for capitalization (CAP), POS tagging (TAG) and segmentation by query type. Numbers in parentheses indicate % of improvement over the *i-PRF* baseline. Best result per measure and annotation is boldfaced. * denotes statistically significant differences with *i-PRF*.

all annotation types, using both performance measures.

In Table 1, we can also observe that the joint annotation methods are, in all cases, better than the corresponding independent ones. The highest improvements are attained by *j-PRF*, which always demonstrates the best performance both in terms of F1 and MQA. These results attest to both the importance of doing a joint optimization over the entire set of annotations and to the robustness of the initial annotations done by the *i-PRF* method. In all but one case, the *j-PRF* method, which uses these annotations as features, outperforms the *j-QRY* method that only uses the annotation done by *i-QRY*.

The most significant improvements as a result of joint annotation are observed for the segmentation task. In this task, joint annotation achieves close to 20% improvement in MQA over the *i-QRY* method, and more than 10% improvement in MQA over the *i-PRF* method. These improvements indicate that the segmentation decisions are strongly guided by capitalization and POS tagging. We also note that, in case of segmentation, the differences in performance between the two joint annotation methods, *j-QRY* and *j-PRF*, are not significant, indicating that the context of additional annotations in *j-QRY* makes up for the lack of more robust pseudo-relevance feedback based features.

We also note that the *lowest* performance improvement as a result of joint annotation is evidenced for POS tagging. The improvements of joint annotation method *j-PRF* over the *i-PRF* method are less than 1%, and are not statistically significant. This is not surprising, since the standard POS taggers often already use bigrams and capitalization at training time, and do not acquire much additional information from other annotations.

6.3 Evaluation by Query Type

Table 2 presents a detailed analysis of the performance of the best independent (*i-PRF*) and joint (*j-PRF*) annotation methods by the three query types used for evaluation: verbal phrases, questions and keyword queries. From the analysis in Table 2, we note that the contribution of joint annotation varies significantly across query types. For instance, using *j-PRF* always leads to statistically significant improvements over the *i-PRF* baseline for questions. On the other hand, it is either statistically indistinguishable, or even significantly worse (in the case of capitalization) than the *i-PRF* baseline for the verbal phrases.

Table 2 also demonstrates that joint annotation has a different impact on various annotations for the *same* query type. For instance, *j-PRF* has a significant positive effect on capitalization and segmentation for keyword queries, but only marginally improves the POS tagging. Similarly, for the verbal phrases, *j-PRF* has a significant positive effect only for the segmentation annotation.

These variances in the performance of the *j-PRF* method point to the differences in the structure be-

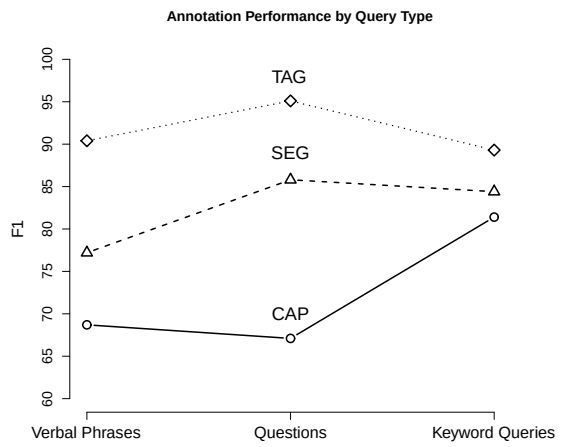


Figure 3: **C**omparative performance (in terms of F1 for capitalization and segmentation and accuracy for POS tagging) of the *j-PRF* method on the three query types.

tween the query types. While dependence between the annotations plays an important role for question and keyword queries, which often share a common grammatical structure, this dependence is less useful for verbal phrases, which have a more diverse linguistic structure. Accordingly, a more in-depth investigation of the linguistic structure of the verbal phrase queries is an interesting direction for future work.

6.4 Annotation Difficulty

Recall that in our experiments, out of the overall 250 annotated queries, there are 96 verbal phrases, 93 questions and 61 keyword queries. Figure 3 shows a plot that contrasts the relative performance for these three query types of our best-performing joint annotation method, *j-PRF*, on capitalization, POS tagging and segmentation annotation tasks. Next, we analyze the performance profiles for the annotation tasks shown in Figure 3.

For the capitalization task, the performance of *j-PRF* on verbal phrases and questions is similar, with the difference below 3%. The performance for keyword queries is much higher — with improvement over 20% compared to either of the other two types. We attribute this increase to both a larger number of positive examples in the short keyword queries (a higher percentage of terms in keyword queries is capitalized) and their simpler syntactic structure (adjacent terms in these queries are likely to have the same case).

SEG	F1	MQA
SEG-1	0.768	0.754
SEG-2	**0.824***	0.787*
j-PRF	0.819* *(+6.7%/-0.6%)*	**0.803*** *(+6.5%/+2.1%)*

Table 3: **C**omparison of the segmentation performance of the *j-PRF* method to two state-of-the-art segmentation methods. Numbers in parentheses indicate % of improvement over the *SEG-1* and *SEG-2* baselines respectively. Best result per measure and annotation is boldfaced. * denotes statistically significant differences with *SEG-1*.

For the segmentation task, the performance is at its best for the question and keyword queries, and at its worst (with a drop of 11%) for the verbal phrases. We hypothesize that this is due to the fact that question queries and keyword queries tend to have repetitive structures, while the grammatical structure for verbose queries is much more diverse.

For the tagging task, the performance profile is reversed, compared to the other two tasks — the performance is at its worst for keyword queries, since their grammatical structure significantly differs from the grammatical structure of sentences in news articles, on which the POS tagger is trained. For question queries the performance is the best (6% increase over the keyword queries), since they resemble sentences encountered in traditional corpora.

It is important to note that the results reported in Figure 3 are based on training the joint annotation model on *all* available queries with 10-fold cross-validation. We might get different profiles if a separate annotation model was trained for each query type. In our case, however, the number of queries from each type is not sufficient to train a reliable model. We leave the investigation of separate training of joint annotation models by query type to future work.

6.5 Additional Comparisons

In order to further evaluate the proposed joint annotation method, *j-PRF*, in this section we compare its performance to other query annotation methods previously reported in the literature. Unfortunately, there is not much published work on query capitalization and query POS tagging that goes beyond the simple query-based methods described in Sec-

tion 4.1. The published work on the more advanced methods usually requires access to large amounts of proprietary user data such as query logs and clicks (Barr et al., 2008; Guo et al., 2008; Guo et al., 2009).

Therefore, in this section we focus on recent work on query segmentation (Bergsma and Wang, 2007; Hagen et al., 2010). We compare the segmentation effectiveness of our best performing method, *j-PRF*, to that of these query segmentation methods.

The first method, *SEG-1*, was first proposed by Hagen et al. (2010). It is currently the most effective publicly disclosed *unsupervised* query segmentation method. *SEG-1* method requires an access to a large web n-gram corpus (Brants and Franz, 2006). The optimal segmentation for query Q, S^*_Q, is then obtained using

$$S^*_Q = \underset{S \in \mathcal{S}_Q}{\operatorname{argmax}} \sum_{s \in S, |s|>1} |s|^{|s|} count(s),$$

where $\mathcal{S}_Q$ is the set of all possible query segmentations, S is a possible segmentation, s is a segment in S, and $count(s)$ is the frequency of s in the web n-gram corpus.

The second method, *SEG-2*, is based on a successful supervised segmentation method, which was first proposed by Bergsma and Wang (2007). *SEG-2* employs a large set of features, and is pre-trained on the query collection described by Bergsma and Wang (2007). The features used by the *SEG-2* method are described by Bendersky et al. (2009), and include, among others, n-gram frequencies in a sample of a query log, web corpus and Wikipedia titles.

Table 3 demonstrates the comparison between the *j-PRF*, *SEG-1* and *SEG-2* methods. When compared to the *SEG-1* baseline, *j-PRF* is significantly more effective, even though it only employs bigram counts (see Eq. 4), instead of the high-order n-grams used by *SEG-1*, for computing the score of a segmentation. This results underscores the benefit of joint annotation, which leverages capitalization and POS tagging to improve the quality of the segmentation.

When compared to the *SEG-2* baseline, *j-PRF* and *SEG-2* are statistically indistinguishable. *SEG-2* posits a slightly better F1, while *j-PRF* has a better MQA. This result demonstrates that the segmentation produced by the *j-PRF* method is as effective as the segmentation produced by the current supervised state-of-the-art segmentation methods, which employ external data sources and high-order n-grams. The benefit of the *j-PRF* method compared to the *SEG-2* method, is that, simultaneously with the segmentation, it produces several additional query annotations (in this case, capitalization and POS tagging), eliminating the need to construct separate sequence classifiers for each annotation.

7 Conclusions

In this paper, we have investigated a joint approach for annotating search queries with linguistic structures, including capitalization, POS tags and segmentation. To this end, we proposed a probabilistic approach for performing joint query annotation that takes into account the dependencies that exist between the different annotation types.

Our experimental findings over a range of queries from a web search log unequivocally point to the superiority of the joint annotation methods over both query-based and pseudo-relevance feedback based independent annotation methods. These findings indicate that the different annotations are mutually-dependent.

We are encouraged by the success of our joint query annotation technique, and intend to pursue the investigation of its utility for IR applications. In the future, we intend to research the use of joint query annotations for additional IR tasks, e.g., for constructing better query formulations for ranking algorithms.

8 Acknowledgment

This work was supported in part by the Center for Intelligent Information Retrieval and in part by ARRA NSF IIS-9014442. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Niranjan Balasubramanian and James Allan. 2009. Syntactic query models for restatement retrieval. In *Proc. of SPIRE*, pages 143–155.

Cory Barr, Rosie Jones, and Moira Regelson. 2008. The linguistic structure of english web-search queries. In *Proc. of EMNLP*, pages 1021–1030.

Michael Bendersky and W. Bruce Croft. 2009. Analysis of long queries in a large scale search log. In *Proc. of Workshop on Web Search Click Data*, pages 8–14.

Michael Bendersky, David Smith, and W. Bruce Croft. 2009. Two-stage query segmentation for information retrieval. In *Proc. of SIGIR*, pages 810–811.

Michael Bendersky, W. Bruce Croft, and David A. Smith. 2010. Structural annotation of search queries using pseudo-relevance feedback. In *Proc. of CIKM*, pages 1537–1540.

Shane Bergsma and Qin I. Wang. 2007. Learning noun phrase query segmentation. In *Proc. of EMNLP*, pages 819–826.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram Version 1.

Chris Buckley. 1995. Automatic query expansion using SMART. In *Proc. of TREC-3*, pages 69–80.

Jenny R. Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proc. of NAACL*, pages 326–334.

Jiafeng Guo, Gu Xu, Hang Li, and Xueqi Cheng. 2008. A unified and discriminative model for query refinement. In *Proc. of SIGIR*, pages 379–386.

Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. 2009. Named entity recognition in query. In *Proc. of SIGIR*, pages 267–274.

Matthias Hagen, Martin Potthast, Benno Stein, and Christof Braeutigam. 2010. The power of naive query segmentation. In *Proc. of SIGIR*, pages 797–798.

Matthias Hagen, Martin Potthast, Benno Stein, and Christof Bräutigam. 2011. Query segmentation revisited. In *Proc. of WWW*, pages 97–106.

Rosie Jones and Daniel C. Fain. 2003. Query word deletion prediction. In *Proc. of SIGIR*, pages 435–436.

Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *Proc. of WWW*, pages 387–396.

Giridhar Kumaran and James Allan. 2007. A case for shorter queries, and helping user create them. In *Proc. of NAACL*, pages 220–227.

Giridhar Kumaran and Vitor R. Carvalho. 2009. Reducing long queries using query quality predictors. In *Proc. of SIGIR*, pages 564–571.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, pages 282–289.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance based language models. In *Proc. of SIGIR*, pages 120–127.

Matthew Lease. 2007. Natural language processing for information retrieval: the time is ripe (again). In *Proceedings of PIKM*.

Xiao Li. 2010. Understanding the semantic structure of noun phrase queries. In *Proc. of ACL*, pages 1337–1345, Morristown, NJ, USA.

Rachel T. Lo, Ben He, and Iadh Ounis. 2005. Automatically building a stopword list for an information retrieval system. In *Proc. of DIR*.

Yumao Lu, Fuchun Peng, Gilad Mishne, Xing Wei, and Benoit Dumoulin. 2009. Improving Web search relevance with semantic features. In *Proc. of EMNLP*, pages 648–657.

Mehdi Manshadi and Xiao Li. 2009. Semantic Tagging of Web Search Queries. In *Proc. of ACL*, pages 861–869.

André F. T. Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proc. of EMNLP*, pages 157–166.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proc. of ACL*, pages 950–958.

Marius Paşca. 2007. Weakly-supervised discovery of named entities using web search queries. In *Proc. of CIKM*, pages 683–690.

Dou Shen, Toby Walkery, Zijian Zhengy, Qiang Yangz, and Ying Li. 2008. Personal name classification in web queries. In *Proc. of WSDM*, pages 149–158.

Bin Tan and Fuchun Peng. 2008. Unsupervised query segmentation using generative language models and Wikipedia. In *Proc. of WWW*, pages 347–356.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34:161–191, June.

Xing Wei, Fuchun Peng, and Benoit Dumoulin. 2008. Analyzing web text association to disambiguate abbreviation in queries. In *Proc. of SIGIR*, pages 751–752.

Query Weighting for Ranking Model Adaptation

Peng Cai[1], **Wei Gao**[2], **Aoying Zhou**[1], **and Kam-Fai Wong**[2,3]
[1]East China Normal University, Shanghai, China
pengcai2010@gmail.com, ayzhou@sei.ecnu.edu.cn
[2]The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{wgao, kfwong}@se.cuhk.edu.hk
[3]Key Laboratory of High Confidence Software Technologies, Ministry of Education, China

Abstract

We propose to directly measure the importance of queries in the source domain to the target domain where no rank labels of documents are available, which is referred to as query weighting. Query weighting is a key step in ranking model adaptation. As the learning object of ranking algorithms is divided by query instances, we argue that it's more reasonable to conduct importance weighting at query level than document level. We present two query weighting schemes. The first compresses the query into a *query feature vector*, which aggregates all document instances in the same query, and then conducts query weighting based on the query feature vector. This method can efficiently estimate query importance by compressing query data, but the potential risk is information loss resulted from the compression. The second measures the similarity between the source query and each target query, and then combines these *fine-grained* similarity values for its importance estimation. Adaptation experiments on LETOR3.0 data set demonstrate that query weighting significantly outperforms document instance weighting methods.

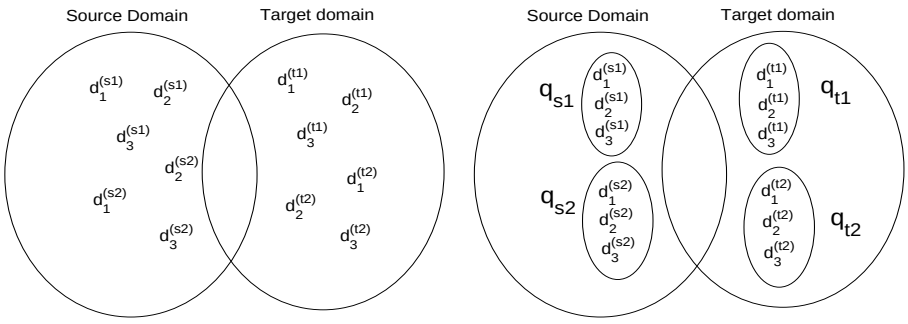
1 Introduction

Learning to rank, which aims at ranking documents in terms of their relevance to user's query, has been widely studied in machine learning and information retrieval communities (Herbrich et al., 2000; Freund et al., 2004; Burges et al., 2005; Yue et al., 2007; Cao et al., 2007; Liu, 2009). In general, large amount of training data need to be annotated by domain experts for achieving better ranking performance. In real applications, however, it is time consuming and expensive to annotate training data for each search domain. To alleviate the lack of training data in the target domain, many researchers have proposed to transfer ranking knowledge from the source domain with plenty of labeled data to the target domain where only a few or no labeled data is available, which is known as ranking model adaptation (Chen et al., 2008a; Chen et al., 2010; Chen et al., 2008b; Geng et al., 2009; Gao et al., 2009).

Intuitively, the more similar an source instance is to the target instances, it is expected to be more useful for cross-domain knowledge transfer. This motivated the popular domain adaptation solution based on instance weighting, which assigns larger weights to those transferable instances so that the model trained on the source domain can adapt more effectively to the target domain (Jiang and Zhai, 2007). Existing instance weighting schemes mainly focus on the adaptation problem for classification (Zadrozny, 2004; Huang et al., 2007; Jiang and Zhai, 2007; Sugiyama et al., 2008).

Although instance weighting scheme may be applied to documents for ranking model adaptation, the difference between classification and learning to rank should be highlighted to take careful consideration. Compared to classification, the learning object for ranking is essentially a query, which contains a list of document instances each with a relevance judgement. Recently, researchers proposed listwise ranking algorithms (Yue et al., 2007; Cao et al., 2007) to take the whole query as a learning object. The benchmark evaluation showed that list-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 112–122,
Portland, Oregon, June 19-24, 2011.



(a) Instance based weighting (b) Query based weighting

Figure 1: The information about which document instances belong to the same query is lost in document instance weighting scheme. To avoid losing this information, query weighting takes the query as a whole and directly measures its importance.

wise approach significantly outperformed pointwise approach, which takes each document instance as independent learning object, as well as pairwise approach, which concentrates learning on the order of a pair of documents (Liu, 2009). Inspired by the principle of listwise approach, we hypothesize that the importance weighting for ranking model adaptation could be done better at query level rather than document level.

Figure 1 demonstrates the difference between instance weighting and query weighting, where there are two queries q_{s1} and q_{s2} in the source domain and q_{t1} and q_{t2} in the target domain, respectively, and each query has three retrieved documents. In Figure 1(a), source and target domains are represented as a bag of document instances. It is worth noting that the information about which document instances belong to the same query is lost. To avoid this information loss, query weighting scheme shown as Figure 1(b) directly measures importance weight at query level.

Instance weighting makes the importance estimation of document instances inaccurate when documents of the same source query are similar to the documents from different target queries. Take Figure 2 as a toy example, where the document instance is represented as a feature vector with four features. No matter what weighting schemes are used, it makes sense to assign high weights to source queries q_{s1} and q_{s2} because they are similar to target queries q_{t1} and q_{t2}, respectively. Meanwhile, the source query q_{s3} should be weighted lower because

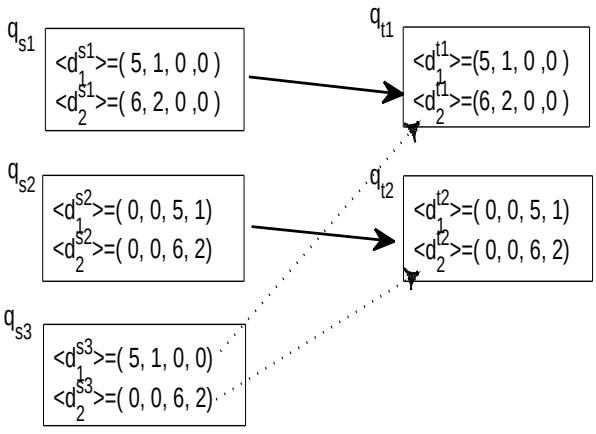


Figure 2: A toy example showing the problem of document instance weighting scheme.

it's not quite similar to any of q_{t1} and q_{t2} at query level, meaning that the ranking knowledge from q_{s3} is different from that of q_{t1} and q_{t2} and thus less useful for the transfer to the target domain. Unfortunately, the three source queries q_{s1}, q_{s2} and q_{s3} would be weighted equally by document instance weighting scheme. The reason is that all of their documents are similar to the two document instances in target domain despite the fact that the documents of q_{s3} correspond to their counterparts from *different* target queries.

Therefore, we should consider the source query as a whole and directly measure the query importance. However, it's not trivial to directly estimate

a query's weight because a query is essentially provided as a matrix where each row represents a vector of document features. In this work, we present two simple but very effective approaches attempting to resolve the problem from distinct perspectives: (1) we compress each query into a *query feature vector* by aggregating all of its document instances, and then conduct query weighting on these query feature vectors; (2) we measure the similarity between the source query and each target query one by one, and then combine these *fine-grained* similarity values to calculate its importance to the target domain.

2 Instance Weighting Scheme Review

The basic idea of instance weighting is to put larger weights on source instances which are more similar to target domain. As a result, the key problem is how to accurately estimate the instance's weight indicating its importance to target domain. (Jiang and Zhai, 2007) used a small number of *labeled* data from target domain to weight source instances. Recently, some researchers proposed to weight source instance only using *unlabeled* target instances (Shimodaira, 2000; Sugiyama et al., 2008; Huang et al., 2007; Zadrozny, 2004; Gao et al., 2010). In this work, we also focus on weighting source queries only using *unlabeled* target queries.

(Gao et al., 2010; Ben-David et al., 2010) proposed to use a classification hyperplane to separate source instances from target instances. With the domain separator, the probability that a source instance is classified to target domain can be used as the importance weight. Other instance weighting methods were proposed for the sample selection bias or covariate shift in the more general setting of classifier learning (Shimodaira, 2000; Sugiyama et al., 2008; Huang et al., 2007; Zadrozny, 2004). (Sugiyama et al., 2008) used a natural model selection procedure, referred to as Kullback-Leibler divergence Importance Estimation Procedure (KLIEP), for automatically tuning parameters, and showed that its importance estimation was more accurate. The main idea is to directly estimate the density function ratio of target distribution $p_t(x)$ to source distribution $p_s(x)$, i.e. $w(x) = \frac{p_t(x)}{p_s(x)}$. Then model $w(x)$ can be used to estimate the importance of source instances. Model parameters were computed with a linear model by minimizing the KL-divergence from $p_t(x)$ to its estimator $\hat{p}_t(x)$. Since $\hat{p}_t(x) = \hat{w}(x)p_s(x)$, the ultimate objective only contains model $\hat{w}(x)$.

For using instance weighting in pairwise ranking algorithms, the weights of document instances should be transformed into those of document pairs (Gao et al., 2010). Given a pair of documents $\langle x_i, x_j \rangle$ and their weights w_i and w_j, the pairwise weight w_{ij} could be estimated probabilistically as $w_i * w_j$. To consider query factor, query weight was further estimated as the average value of the weights over all the pairs, i.e., $w_q = \frac{1}{M} \sum_{i,j} w_{ij}$, where M is the number of pairs in query q. Additionally, to take the advantage of both query and document information, a probabilistic weighting for $\langle x_i, x_j \rangle$ was modeled by $w_q * w_{ij}$. Through the transformation, instance weighting schemes for classification can be applied to ranking model adaptation.

3 Query Weighting

In this section, we extend instance weighting to directly estimate query importance for more effective ranking model adaptation. We present two query weighting methods from different perspectives. Note that although our methods are based on domain separator scheme, other instance weighting schemes such as KLIEP (Sugiyama et al., 2008) can also be extended similarly.

3.1 Query Weighting by Document Feature Aggregation

Our first query weighting method is inspired by the recent work on local learning for ranking (Geng et al., 2008; Banerjee et al., 2009). The query can be compressed into a query feature vector, where each feature value is obtained by the aggregate of its corresponding features of all documents in the query. We concatenate two types of aggregates to construct the query feature vector: the mean $\vec{\mu} = \frac{1}{|q|} \sum_{i=1}^{|q|} \vec{f_i}$ and the variance $\vec{\sigma} = \frac{1}{|q|} \sum_{i=1}^{|q|} (\vec{f_i} - \vec{\mu})^2$, where $\vec{f_i}$ is the feature vector of document i and $|q|$ denotes the number of documents in q . Based on the aggregation of documents within each query, we can use a domain separator to directly weight the source queries with the set of queries from both domains.

Given query data sets $D_s = \{q_s^i\}_{i=1}^m$ and $D_t = \{q_t^j\}_{j=1}^n$ respectively from the source and target do-

Algorithm 1 Query Weighting Based on Document Feature Aggregation in the Query

Input:

Queries in the source domain, $D_s = \{q_s^i\}_{i=1}^m$;

Queries in the target domain, $D_t = \{q_t^j\}_{j=1}^n$;

Output:

Importance weights of queries in the source domain, $IW_s = \{W_i\}_{i=1}^m$;

1: $y_s = -1, y_t = +1$;
2: **for** $i = 1; i \leq m; i++$ **do**
3: Calculate the mean vector $\vec{\mu}_i$ and variance vector $\vec{\sigma}_i$ for q_s^i;
4: Add query feature vector $\vec{q}_s^i = (\vec{\mu}_i, \vec{\sigma}_i, y_s)$ to D'_s ;
5: **end for**
6: **for** $j = 1; j \leq n; j++$ **do**
7: Calculate the mean vector $\vec{\mu}_j$ and variance vector $\vec{\sigma}_j$ for q_t^j;
8: Add query feature vector $\vec{q}_t^j = (\vec{\mu}_j, \vec{\sigma}_j, y_t)$ to D'_t;
9: **end for**
10: Find classification hyperplane H_{st} which separates D'_s from D'_t;
11: **for** $i = 1; i \leq m; i++$ **do**
12: Calculate the distance of $\vec{q}_s^i$ to H_{st}, denoted as $\mathcal{L}(\vec{q}_s^i)$;
13: $W_i = P(q_s^i \in D_t) = \frac{1}{1+exp(\alpha * \mathcal{L}(\vec{q}_s^i)+\beta)}$
14: Add W_i to IW_s;
15: **end for**
16: **return** IW_s;

mains, we use algorithm 1 to estimate the probability that the query q_s^i can be classified to D_t, i.e. $P(q_s^i \in D_t)$, which can be used as the importance of q_s^i relative to the target domain. From step 1 to 9, D'_s and D'_t are constructed using query feature vectors from source and target domains. Then, a classification hyperplane H_{st} is used to separate D'_s from D'_t in step 10. The distance of the query feature vector $\vec{q}_s^i$ from H_{st} are transformed to the probability $P(q_s^i \in D_t)$ using a sigmoid function (Platt and Platt, 1999).

3.2 Query Weighting by Comparing Queries across Domains

Although the query feature vector in algorithm 1 can approximate a query by aggregating its documents' features, it potentially fails to capture important feature information due to the averaging effect during the aggregation. For example, the merit of features in some influential documents may be canceled out in the mean-variance calculation, resulting in many distorted feature values in the query feature vector that hurts the accuracy of query classification hyperplane. This urges us to propose another query weighting method from a different perspective of query similarity.

Intuitively, the importance of a source query to the target domain is determined by its overall similarity to every target query. Based on this intuition, we leverage domain separator to measure the similarity between a source query and each one of the target queries, where an individual domain separator is created for each pair of queries. We estimate the weight of a source query using algorithm 2. Note that we assume document instances in the same query are conditionally independent and all queries are independent of each other. In step 3, $D'_{q_s^i}$ is constructed by all the document instances $\{\vec{x}_k\}$ in query q_s^i with the domain label y_s. For each target query q_t^j, we use the classification hyperplane H_{ij} to estimate $P(\vec{x}_k \in D'_{q_t^j})$, i.e. the probability that each document $\vec{x}_k$ of q_s^i is classified into the document set of q_t^j (step 8). Then the similarity between q_s^i and q_t^j is measured by the probability $P(q_s^i \sim q_t^j)$ at step 9. Finally, the probability of q_s^i belonging to the target domain $P(q_s^i \in D_t)$ is calculated at step 11.

It can be expected that algorithm 2 will generate

Algorithm 2 Query Weighting by Comparing Source and Target Queries

Input:

Queries in source domain, $D_s = \{q_s^i\}_{i=1}^m$;

Queries in target domain, $D_t = \{q_t^j\}_{j=1}^n$;

Output:

Importance weights of queries in source domain, $IW_s = \{W_i\}_{i=1}^m$;

$y_s = -1, y_t = +1$;
for $i = 1; i \leq m; i++$ **do**
  Set $D'_{q_s^i}$=$\{\vec{x}_k, y_s)\}_{k=1}^{|q_s^i|}$;
  for $j = 1; j \leq n; j++$ **do**
    Set $D'_{q_t^j}$=$\{\vec{x}_{k'}, y_t)\}_{k'=1}^{|q_t^j|}$;
    Find a classification hyperplane H_{ij} which separates $D'_{q_s^i}$ from $D'_{q_t^j}$;
    For each k, calculate the distance of $\vec{x}_k$ to H_{ij}, denoted as $\mathcal{L}(\vec{x}_k)$;
    For each k, calculate $P(\vec{x}_k \in D'_{q_t^j}) = \frac{1}{1+exp(\alpha * \mathcal{L}(\vec{x}_k)+\beta)}$;
    Calculate $P(q_s^i \sim q_t^j) = \frac{1}{|q_s^i|} \sum_{k=1}^{|q_s^i|} P(\vec{x}_k \in D'_{q_t^j})$;
  end for
  Add $W_i = P(q_s^i \in D_t) = \frac{1}{n} \sum_{j=1}^n P(q_s^i \sim q_t^j)$ to IW_s;
end for
return IW_s;

more precise measures of query similarity by utilizing the more fine-grained classification hyperplane for separating the queries of two domains.

4 Ranking Model Adaptation via Query Weighting

To adapt the source ranking model to the target domain, we need to incorporate query weights into existing ranking algorithms. Note that query weights can be integrated with either pairwise or listwise algorithms. For pairwise algorithms, a straightforward way is to assign the query weight to all the document pairs associated with this query. However, document instance weighting cannot be appropriately utilized in listwise approach. In order to compare query weighting with document instance weighting, we need to fairly apply them for the same approach of ranking. Therefore, we choose pairwise approach to incorporate query weighting. In this section, we extend Ranking SVM (RSVM) (Herbrich et al., 2000; Joachims, 2002) — one of the typical pairwise algorithms for this.

Let's assume there are m queries in the data set of source domain, and for each query q_i there are $\ell(q_i)$ number of meaningful document pairs that can be constructed based on the ground truth rank labels. Given ranking function f, the objective of RSVM is presented as follows:

$$\min \frac{1}{2}||\vec{w}||^2 + C \sum_{i=1}^{m} \sum_{j=1}^{\ell(q_i)} \xi_{ij} \quad (1)$$
$$\text{subject to } z_{ij} * f(\vec{w}, \vec{x}_{q_i}^{j(1)} - \vec{x}_{q_i}^{j(2)}) \geq 1 - \xi_{ij}$$
$$\xi_{ij} \geq 0, i = 1, \ldots, m; j = 1, \ldots, \ell(q_i)$$

where $\vec{x}_{q_i}^{j(1)}$ and $\vec{x}_{q_i}^{j(2)}$ are two documents with different rank label, and $z_{ij} = +1$ if $\vec{x}_{q_i}^{j(1)}$ is labeled more relevant than $\vec{x}_{q_i}^{j(2)}$; or $z_{ij} = -1$ otherwise.

Let $\lambda = \frac{1}{2C}$ and replace ξ_{ij} with Hinge Loss function $(.)^+$, Equation 1 can be turned to the following form:

$$\min \lambda||\vec{w}||^2 + \sum_{i=1}^{m} \sum_{j=1}^{\ell(q_i)} \left(1 - z_{ij} * f(\vec{w}, \vec{x}_{q_i}^{j(1)} - \vec{x}_{q_i}^{j(2)})\right)^+ \quad (2)$$

Let $IW(q_i)$ represent the importance weight of source query q_i. Equation 2 is extended for integrating the query weight into the loss function in a

straightforward way:

$$\min \lambda||\vec{w}||^2 + \sum_{i=1}^{m} IW(q_i) * \sum_{j=1}^{\ell(q_i)} \left(1 - z_{ij} * f(\vec{w}, \vec{x}_{q_i}^{j(1)} - \vec{x}_{q_i}^{j(2)})\right)^+$$

where $IW(.)$ takes any one of the weighting schemes given by algorithm 1 and algorithm 2.

5 Evaluation

We evaluated the proposed two query weighting methods on TREC-2003 and TREC-2004 web track datasets, which were released through LETOR3.0 as a benchmark collection for learning to rank by (Qin et al., 2010). Originally, different query tasks were defined on different parts of data in the collection, which can be considered as different domains for us. Adaptation takes place when ranking tasks are performed by using the models trained on the domains in which they were originally defined to rank the documents in other domains. Our goal is to demonstrate that query weighting can be more effective than the state-of-the-art document instance weighting.

5.1 Datasets and Setup

Three query tasks were defined in TREC-2003 and TREC-2004 web track, which are home page finding (HP), named page finding (NP) and topic distillation (TD) (Voorhees, 2003; Voorhees, 2004). In this dataset, each document instance is represented by 64 features, including low-level features such as term frequency, inverse document frequency and document length, and high-level features such as BM25, language-modeling, PageRank and HITS. The number of queries of each task is given in Table 1.

The *baseline* ranking model is an RSVM directly trained on the source domain without using any weighting methods, denoted as *no-weight*. We implemented two weighting measures based on domain separator and Kullback-Leibler divergence, referred to *DS* and *KL*, respectively. In *DS* measure, three document instance weighting methods based on probability principle (Gao et al., 2010) were implemented for comparison, denoted as *doc-pair*, *doc-avg* and *doc-comb* (see Section 2). In *KL* measure, there is no probabilistic meaning for KL weight and the *doc-comb* based on KL is not interpretable, and we only present the results of *doc-pair* and *doc-avg* for KL measure. Our proposed query weighting methods are denoted by *query-aggr* and *query-comp*, corresponding to document feature aggregation in query and query comparison across domains, respectively. All ranking models above were trained only on source domain training data and the labeled data of target domain was just used for testing.

Query Task	TREC 2003	TREC 2004
Topic Distillation	50	75
Home Page finding	150	75
Named Page finding	150	75

Table 1: The number of queries in TREC-2003 and TREC-2004 web track

For training the models efficiently, we implemented RSVM with Stochastic Gradient Descent (SGD) optimizer (Shalev-Shwartz et al., 2007). The reported performance is obtained by five-fold cross validation.

5.2 Experimental Results

The task of HP and NP are more similar to each other whereas HP/NP is rather different from TD (Voorhees, 2003; Voorhees, 2004). Thus, we carried out HP/NP to TD and TD to HP/NP ranking adaptation tasks. Mean Average Precision (MAP) (Baeza-Yates and Ribeiro-Neto, 1999) is used as the ranking performance measure.

5.2.1 Adaptation from HP/NP to TD

The first set of experiments performed adaptation from HP to TD and NP to TD. The results of MAP are shown in Table 2.

For the DS-based measure, as shown in the table, *query-aggr* works mostly better than *no-weight*,*doc-pair*, *doc-avg* and *doc-comb*, and *query-comp* performs the best among the five weighting methods. T-test on MAP indicates that the improvement of *query-aggr* over *no-weight* is statistically significant on two adaptation tasks while the improvement of document instance weighting over *no-weight* is statistically significant only on one task. All of the improvement of *query-comp* over *no-weight*, *doc-pair*,*doc-avg* and *doc-comb* are statistically significant. This demonstrates the effectiveness of query

Model	Weighting method	HP03 to TD03	HP04 to TD04	NP03 to TD03	NP04 to TD04
	no-weight	0.2508	0.2086	0.1936	0.1756
DS	*doc-pair*	0.2505	0.2042	$0.1982^{\dagger}$	0.1708
	doc-avg	0.2514	0.2019	$0.2122^{\dagger\ddagger}$	0.1716
	doc-comb	0.2562	0.2051	$0.2224^{\dagger\ddagger\sharp}$	0.1793
	query-aggr	0.2573	$\mathbf{0.2106}^{\dagger\ddagger\sharp}$	0.2088	$\mathbf{0.1808}^{\dagger\ddagger\sharp}$
	query-comp	$\mathbf{0.2816}^{\dagger\ddagger\sharp}$	$\mathbf{0.2147}^{\dagger\ddagger\sharp}$	$\mathbf{0.2392}^{\dagger\ddagger\sharp}$	$\mathbf{0.1861}^{\dagger\ddagger\sharp}$
KL	*doc-pair*	0.2521	0.2048	0.1901	0.1761
	doc-avg	0.2534	$0.2127^{\dagger}$	0.1904	0.1777
	doc-comb	-	-	-	-
	query-aggr	0.1890	0.1901	0.1870	0.1643
	query-comp	$0.2548^{\dagger}$	$0.2142^{\dagger}$	$0.2313^{\dagger\ddagger\sharp}$	$0.1807^{\dagger}$

Table 2: Results of MAP for HP/NP to TD adaptation. †, ‡, ♯ and boldface indicate significantly better than *no-weight*, *doc-pair*, *doc-avg* and *doc-comb*, respectively. Confidence level is set at 95%

weighting compared to document instance weighting.

Furthermore, *query-comp* can perform better than *query-aggr*. The reason is that although document feature aggregation might be a reasonable representation for a set of document instances, it is possible that some information could be lost or distorted in the process of compression. By contrast, more accurate query weights can be achieved by the more fine-grained similarity measure between the source query and all target queries in algorithm 2.

For the KL-based measure, similar observation can be obtained. However, it's obvious that DS-based models can work better than the KL-based. The reason is that KL conducts weighting by density function ratio which is sensitive to the data scale. Specifically, after document feature aggregation, the number of query feature vectors in all adaptation tasks is no more than 150 in source and target domains. It renders the density estimation in *query-aggr* is very inaccurate since the set of samples is too small. As each query contains 1000 documents, they seemed to provide *query-comp* enough samples for achieving reasonable estimation of the density functions in both domains.

5.2.2 Adaptation from TD to HP/NP

To further validate the effectiveness of query weighting, we also conducted adaptation from TD to HP and TD to NP . MAP results with significant test are shown in Table 3.

We can see that document instance weighting schemes including *doc-pair*, *doc-avg* and *doc-comb* can not outperform *no-weight* based on MAP measure. The reason is that each query in TD has 1000 retrieved documents in which 10-15 documents are relevant whereas each query in HP or NP only consists 1-2 relevant documents. Thus, when TD serves as the source domain, it leads to the problem that too many document pairs were generated for training the RSVM model. In this case, a small number of documents that were weighted inaccurately can make significant impact on many number of document pairs. Since query weighting method directly estimates the query importance instead of document instance importance, both *query-aggr* and *query-comp* can avoid such kind of negative influence that is inevitable in the three document instance weighting methods.

5.2.3 The Analysis on Source Query Weights

An interesting problem is which queries in the source domain are assigned high weights and why it's the case. Query weighting assigns each source query with a weight value. Note that it's not meaningful to directly compare absolute weight values between *query-aggr* and *query-comp* because source query weights from distinct weighting methods have different range and scale. However, it is feasible to compare the weights with the same weighting method. Intuitively, if the ranking model learned from a source query can work well in target domain, it should get high weight. According to this intuition, if ranking models $f_{q_s^1}$ and $f_{q_s^2}$ are learned

model	weighting scheme	TD03 to HP03	TD04 to HP04	TD03 to NP03	TD04 to NP04
	no-weight	0.6986	0.6158	0.5053	0.5427
DS	*doc-pair*	0.6588	$0.6235^{\dagger}$	0.4878	0.5212
	doc-avg	0.6654	0.6200	0.4736	0.5035
	doc-comb	0.6932	$0.6214^{\dagger}$	0.4974	0.5077
	query-aggr	$\mathbf{0.7179}^{\dagger\ddagger\sharp}$	$0.6292^{\dagger\ddagger\sharp}$	$\mathbf{0.5198}^{\dagger\ddagger\sharp}$	$\mathbf{0.5551}^{\dagger\ddagger\sharp}$
	query-comp	$\mathbf{0.7297}^{\dagger\ddagger\sharp}$	$\mathbf{0.6499}^{\dagger\ddagger\sharp}$	$\mathbf{0.5203}^{\dagger\ddagger\sharp}$	$\mathbf{0.6541}^{\dagger\ddagger\sharp}$
KL	*doc-pair*	0.6480	0.6107	0.4633	0.5413
	doc-avg	0.6472	0.6132	0.4626	0.5406
	doc-comb	–	–	–	–
	query-aggr	0.6263	0.5929	0.4597	0.4673
	query-comp	$0.6530^{\ddagger\sharp}$	$0.6358^{\dagger\ddagger\sharp}$	0.4726	$0.5559^{\dagger\ddagger\sharp}$

Table 3: Results of MAP for TD to HP/NP adaptation. †, ‡, ♯ and boldface indicate significantly better than *no-weight*, *doc-pair*, *doc-avg* and *doc-comb*, respectively. Confidence level is set as 95%.

from queries q_s^1 and q_s^2 respectively, and $f_{q_s^1}$ performs better than $f_{q_s^2}$, then the source query weight of q_s^1 should be higher than that of q_s^2.

For further analysis, we compare the weight values between each source query pair, for which we trained RSVM on each source query and evaluated the learned model on test data from target domain. Then, the source queries are ranked according to the MAP values obtained by their corresponding ranking models. The order is denoted as R_{map}. Meanwhile, the source queries are also ranked with respect to their weights estimated by DS-based measure, and the order is denoted as R_{weight}. We hope R_{weight} is correlated as positively as possible with R_{map}. For comparison, we also ranked these queries according to randomly generated query weights, which is denoted as *query-rand* in addition to *query-aggr* and *query-comp*. The Kendall's $\tau = \frac{P-Q}{P+Q}$ is used to measure the correlation (Kendall, 1970), where P is the number of concordant query pairs and Q is the number of discordant pairs. It's noted that τ's range is from -1 to 1, and the larger value means the two ranking is better correlated. The Kendall's τ by different weighting methods are given in Table 4 and 5.

We find that R_{weight} produced by *query-aggr* and *query-comp* are all positively correlated with R_{map} and clearly the orders generated by *query-comp* are more positive than those by *query-aggr*. This is another explanation why *query-comp* outperforms *query-aggr*. Furthermore, both are far better than *query-rand* because the R_{weight} by *query-rand* is actually independent of R_{map}.

weighting	TD03 to HP03	TD04 to HP04
doc-pair	28,835 secs	21,640 secs
query-aggr	182 secs	123 secs
query-comp	15,056 secs	10,081 secs

Table 6: The efficiency of weighting in seconds.

5.2.4 Efficiency

In the situation where there are large scale data in source and target domains, how to efficiently weight a source query is another interesting problem. Without the loss of generality, we reported the weighting time of *doc-pair*, *query-aggr* and *query-comp* from adaptation from TD to HP using DS measure. As *doc-avg* and *doc-comb* are derived from *doc-pair*, their efficiency is equivalent to *doc-pair*.

As shown in table 6, *query-aggr* can efficiently weight query using query feature vector. The reason is two-fold: one is the operation of query document aggregation can be done very fast, and the other is there are 1000 documents in each query of TD or HP, which means that the compression ratio is 1000:1. Thus, the domain separator can be found quickly. In addition, *query-comp* is more efficient than *doc-pair* because *doc-pair* needs too much time to find the separator using all instances from source and target domain. And *query-comp* uses a divide-and-conquer method to measure the similarity of source query to each target query, and then efficiently combine these

Weighting method	HP03 to TD03	HP04 to TD04	NP03 to TD03	NP04 to TD04
query-aggr	0.0906	0.0280	0.0247	0.0525
query-comp	0.1001	0.0804	0.0711	0.1737
query-rand	0.0041	0.0008	-0.0127	0.0163

Table 4: The Kendall's τ of R_{weight} and R_{map} in HP/NP to TD adaptation.

Weighting method	TD03 to HP03	TD04 to HP04	TD03 to NP03	TD04 to NP04
query-aggr	0.1172	0.0121	0.0574	0.0464
query-comp	0.1304	0.1393	0.1586	0.0545
query-rand	-0.0291	0.0022	0.0161	-0.0262

Table 5: The Kendall's τ of R_{weight} and R_{map} in TD to HP/NP adaptation.

fine-grained similarity values.

6 Related Work

Cross-domain knowledge transfer has became an important topic in machine learning and natural language processing (Ben-David et al., 2010; Jiang and Zhai, 2007; Blitzer et al., 2006; Daumé III and Marcu, 2006). (Blitzer et al., 2006) proposed model adaptation using pivot features to build structural feature correspondence in two domains. (Pan et al., 2009) proposed to seek a common features space to reduce the distribution difference between the source and target domain. (Daumé III and Marcu, 2006) assumed training instances were generated from source domain, target domain and cross-domain distributions, and estimated the parameter for the mixture distribution.

Recently, domain adaptation in learning to rank received more and more attentions due to the lack of training data in new search domains. Existing ranking adaptation approaches can be grouped into feature-based (Geng et al., 2009; Chen et al., 2008b; Wang et al., 2009; Gao et al., 2009) and instance-based (Chen et al., 2010; Chen et al., 2008a; Gao et al., 2010) approaches. In (Geng et al., 2009; Chen et al., 2008b), the parameters of ranking model trained on the source domain was adjusted with the small set of labeled data in the target domain. (Wang et al., 2009) aimed at ranking adaptation in heterogeneous domains. (Gao et al., 2009) learned ranking models on the source and target domains independently, and then constructed a stronger model by interpolating the two models. (Chen et al., 2010; Chen et al., 2008a) weighted source instances by using small amount of labeled data in the target domain. (Gao et al., 2010) studied instance weighting based on domain separator for learning to rank by only using training data from source domain. In this work, we propose to directly measure the query importance instead of document instance importance by considering information at both levels.

7 Conclusion

We introduced two simple yet effective query weighting methods for ranking model adaptation. The first represents a set of document instances within the same query as a query feature vector, and then directly measure the source query importance to the target domain. The second measures the similarity between a source query and each target query, and then combine the fine-grained similarity values to estimate its importance to target domain. We evaluated our approaches on LETOR3.0 dataset for ranking adaptation and found that: (1) the first method efficiently estimate query weights, and can outperform the document instance weighting but some information is lost during the aggregation; (2) the second method consistently and significantly outperforms document instance weighting.

8 Acknowledgement

P. Cai and A. Zhou are supported by NSFC (No. 60925008) and 973 program (No. 2010CB731402). W. Gao and K.-F. Wong are supported by national 863 program (No. 2009AA01Z150). We also thank anonymous reviewers for their helpful comments.

References

Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*.

Somnath Banerjee, Avinava Dubey, Jinesh Machchhar, and Soumen Chakrabarti. 2009. Efficient and accurate local learning for ranking. In *SIGIR workshop : Learning to rank for information retrieval*, pages 1–8.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175.

John Blitzer, Ryan Mcdonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of EMNLP*.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of ICML*, pages 89–96.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of ICML*, pages 129 – 136.

Depin Chen, Jun Yan, Gang Wang, Yan Xiong, Weiguo Fan, and Zheng Chen. 2008a. Transrank: A novel algorithm for transfer of rank learning. In *Proceedings of ICDM Workshops*, pages 106–115.

Keke Chen, Rongqing Lu, C.K. Wong, Gordon Sun, Larry Heck, and Belle Tseng. 2008b. Trada: Tree based ranking function adaptation. In *Proceedings of CIKM*.

Depin Chen, Yan Xiong, Jun Yan, Gui-Rong Xue, Gang Wang, and Zheng Chen. 2010. Knowledge transfer for cross domain learning to rank. *Information Retrieval*, 13(3):236–253.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26(1):101–126.

Y. Freund, R. Iyer, R. Schapire, and Y. Singer. 2004. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969.

Jianfeng Gao, Qiang Wu, Chris Burges, Krysta Svore, Yi Su, Nazan Khan, Shalin Shah, and Hongyan Zhou. 2009. Model adaptation via model interpolation and boosting for web search ranking. In *Proceedings of EMNLP*.

Wei Gao, Peng Cai, Kam Fai Wong, and Aoying Zhou. 2010. Learning to rank only using training data from related domain. In *Proceedings of SIGIR*, pages 162–169.

Xiubo Geng, Tie-Yan Liu, Tao Qin, Andrew Arnold, Hang Li, and Heung-Yeung Shum. 2008. Query dependent ranking using k-nearest neighbor. In *Proceedings of SIGIR*, pages 115–122.

Bo Geng, Linjun Yang, Chao Xu, and Xian-Sheng Hua. 2009. Ranking model adaptation for domain-specific search. In *Proceedings of CIKM*.

R. Herbrich, T. Graepel, and K. Obermayer. 2000. *Large Margin Rank Boundaries for Ordinal Regression*. MIT Press, Cambridge.

Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Schölkopf. 2007. Correcting sample selection bias by unlabeled data. In *Proceedings of NIPS*, pages 601–608.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of ACL*.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of SIGKDD*, pages 133–142.

Maurice Kendall. 1970. *Rank Correlation Methods*. Griffin.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. 2009. Domain adaptation via transfer component analysis. In *Proceedings of IJCAI*, pages 1187–1192.

John C. Platt and John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press.

Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. Letor: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13(4):346–374.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. 2007. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814.

Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90:227–244.

Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. 2008. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Proceedings of NIPS*, pages 1433–1440.

Ellen M. Voorhees. 2003. Overview of trec 2003. In *Proceedings of TREC-2003*, pages 1–13.

Ellen M. Voorhees. 2004. Overview of trec 2004. In *Proceedings of TREC-2004*, pages 1–12.

Bo Wang, Jie Tang, Wei Fan, Songcan Chen, Zi Yang, and Yanzhu Liu. 2009. Heterogeneous cross domain ranking in latent space. In *Proceedings of CIKM*.

Y. Yue, T. Finley, F. Radlinski, and T. Joachims. 2007. A support vector method for optimizing average precision. In *Proceedings of SIGIR*, pages 271–278.

Bianca Zadrozny Zadrozny. 2004. Learning and evaluating classifiers under sample selection bias. In *Proceedings of ICML*, pages 325–332.

Automatically Extracting Polarity-Bearing Topics for Cross-Domain Sentiment Classification

Yulan He **Chenghua Lin**[†] **Harith Alani**
Knowledge Media Institute, The Open University
Milton Keynes MK7 6AA, UK
{y.he,h.alani}@open.ac.uk
[†] School of Engineering, Computing and Mathematics
University of Exeter, Exeter EX4 4QF, UK
cl322@exeter.ac.uk

Abstract

Joint sentiment-topic (JST) model was previously proposed to detect sentiment and topic simultaneously from text. The only supervision required by JST model learning is domain-independent polarity word priors. In this paper, we modify the JST model by incorporating word polarity priors through modifying the topic-word Dirichlet priors. We study the polarity-bearing topics extracted by JST and show that by augmenting the original feature space with polarity-bearing topics, the in-domain supervised classifiers learned from augmented feature representation achieve the state-of-the-art performance of 95% on the movie review data and an average of 90% on the multi-domain sentiment dataset. Furthermore, using feature augmentation and selection according to the information gain criteria for cross-domain sentiment classification, our proposed approach performs either better or comparably compared to previous approaches. Nevertheless, our approach is much simpler and does not require difficult parameter tuning.

1 Introduction

Given a piece of text, sentiment classification aims to determine whether the semantic orientation of the text is positive, negative or neutral. Machine learning approaches to this problem (**?**; **?**; **?**; **?**; **?**; **?**) typically assume that classification models are trained and tested using data drawn from some fixed distribution. However, in many practical cases, we may have plentiful labeled examples in the *source* domain, but very few or no labeled examples in the *target* domain with a different distribution. For example, we may have many labeled books reviews, but we are interested in detecting the polarity of electronics reviews. Reviews for different produces might have widely different vocabularies, thus classifiers trained on one domain often fail to produce satisfactory results when shifting to another domain. This has motivated much research on sentiment transfer learning which transfers knowledge from a source task or domain to a different but related task or domain (**?**; **?**; **?**; **?**).

Joint sentiment-topic (JST) model (**?**; **?**) was extended from the latent Dirichlet allocation (LDA) model (**?**) to detect sentiment and topic simultaneously from text. The only supervision required by JST learning is domain-independent polarity word prior information. With prior polarity words extracted from both the MPQA subjectivity lexicon[1] and the appraisal lexicon[2], the JST model achieves a sentiment classification accuracy of 74% on the movie review data[3] and 71% on the multi-domain sentiment dataset[4]. Moreover, it is also able to extract coherent and informative topics grouped under different sentiment. The fact that the JST model does not required any labeled documents for training makes it desirable for domain adaptation in sentiment classification. Many existing approaches solve the sentiment transfer problem by associating words

[1] http://www.cs.pitt.edu/mpqa/

[2] http://lingcog.iit.edu/arc/appraisal_lexicon_2007b.tar.gz

[3] http://www.cs.cornell.edu/people/pabo/movie-review-data

[4] http://www.cs.jhu.edu/~mdredze/datasets/sentiment/index2.html

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 123–131,
Portland, Oregon, June 19-24, 2011.

from different domains which indicate the same sentiment (**?**; **?**). Such an association mapping problem can be naturally solved by the posterior inference in the JST model. Indeed, the polarity-bearing topics extracted by JST essentially capture sentiment associations among words from different domains which effectively overcome the data distribution difference between source and target domains.

The previously proposed JST model uses the sentiment prior information in the Gibbs sampling inference step that a sentiment label will only be sampled if the current word token has no prior sentiment as defined in a sentiment lexicon. This in fact implies a different generative process where many of the word prior sentiment labels are observed. The model is no longer "latent". We propose an alternative approach by incorporating word prior polarity information through modifying the topic-word Dirichlet priors. This essentially creates an informed prior distribution for the sentiment labels and would allow the model to actually be latent and would be consistent with the generative story.

We study the polarity-bearing topics extracted by the JST model and show that by augmenting the original feature space with polarity-bearing topics, the performance of in-domain supervised classifiers learned from augmented feature representation improves substantially, reaching the state-of-the-art results of 95% on the movie review data and an average of 90% on the multi-domain sentiment dataset. Furthermore, using simple feature augmentation, our proposed approach outperforms the structural correspondence learning (SCL) (**?**) algorithm and achieves comparable results to the recently proposed spectral feature alignment (SFA) method (**?**). Nevertheless, our approach is much simpler and does not require difficult parameter tuning.

We proceed with a review of related work on sentiment domain adaptation. We then briefly describe the JST model and present another approach to incorporate word prior polarity information into JST learning. We subsequently show that words from different domains can indeed be grouped under the same polarity-bearing topic through an illustration of example topic words extracted by JST before proposing a domain adaptation approach based on JST. We verify our proposed approach by conducting experiments on both the movie review data and the multi-domain sentiment dataset. Finally, we conclude our work and outline future directions.

2 Related Work

There has been significant amount of work on algorithms for domain adaptation in NLP. Earlier work treats the source domain data as "prior knowledge" and uses maximum a posterior (MAP) estimation to learn a model for the target domain data under this prior distribution (**?**). Chelba and Acero (**?**) also uses the source domain data to estimate prior distribution but in the context of a maximum entropy (ME) model. The ME model has later been studied in (**?**) for domain adaptation where a mixture model is defined to learn differences between domains.

Other approaches rely on unlabeled data in the target domain to overcome feature distribution differences between domains. Motivated by the alternating structural optimization (ASO) algorithm (**?**) for multi-task learning, Blitzer et al. (**?**) proposed structural correspondence learning (SCL) for domain adaptation in sentiment classification. Given labeled data from a source domain and unlabeled data from target domain, SCL selects a set of pivot features to link the source and target domains where pivots are selected based on their common frequency in both domains and also their mutual information with the source labels.

There has also been research in exploring careful structuring of features for domain adaptation. Daumé (**?**) proposed a kernel-mapping function which maps both source and target domains data to a high-dimensional feature space so that data points from the same domain are twice as similar as those from different domains. Dai et al.(**?**) proposed translated learning which uses a language model to link the class labels to the features in the source spaces, which in turn is translated to the features in the target spaces. Dai et al. (**?**) further proposed using spectral learning theory to learn an eigen feature representation from a task graph representing features, instances and class labels. In a similar vein, Pan et al. (**?**) proposed the spectral feature alignment (SFA) algorithm where some domain-independent words are used as a bridge to construct a bipartite graph to model the co-occurrence relationship between domain-specific words and domain-independent words. Feature clusters are

generated by co-align domain-specific and domain-independent words.

Graph-based approach has also been studied in (**?**) where a graph is built with nodes denoting documents and edges denoting content similarity between documents. The sentiment score of each unlabeled documents is recursively calculated until convergence from its neighbors the actual labels of source domain documents and pseudo-labels of target document documents. This approach was later extended by simultaneously considering relations between documents and words from both source and target domains (**?**).

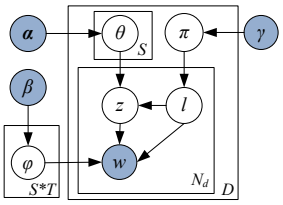
More recently, Seah et al. (**?**) addressed the issue when the predictive distribution of class label given input data of the domains differs and proposed Predictive Distribution Matching SVM learn a robust classifier in the target domain by leveraging the labeled data from only the relevant regions of multiple sources.

3 Joint Sentiment-Topic (JST) Model

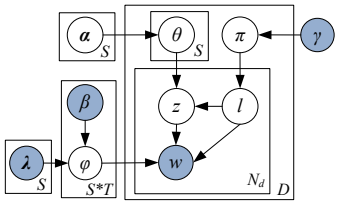
Assume that we have a corpus with a collection of D documents denoted by $C = \{d_1, d_2, ..., d_D\}$; each document in the corpus is a sequence of N_d words denoted by $d = (w_1, w_2, ..., w_{N_d})$, and each word in the document is an item from a vocabulary index with V distinct terms denoted by $\{1, 2, ..., V\}$. Also, let S be the number of distinct sentiment labels, and T be the total number of topics. The generative process in JST which corresponds to the graphical model shown in Figure **??**(a) is as follows:

- For each document d, choose a distribution $\pi_d \sim \text{Dir}(\gamma)$.
- For each sentiment label l under document d, choose a distribution $\theta_{d,l} \sim \text{Dir}(\alpha)$.
- For each word w_i in document d
  - choose a sentiment label $l_i \sim \text{Mult}(\pi_d)$,
  - choose a topic $z_i \sim \text{Mult}(\theta_{d,l_i})$,
  - choose a word w_i from $\varphi^{l_i}_{z_i}$, a Multinomial distribution over words conditioned on topic z_i and sentiment label l_i.

Gibbs sampling was used to estimate the posterior distribution by sequentially sampling each variable of interest, z_t and l_t here, from the distribution over



(a) JST model.



(b) Modified JST model.

Figure 1: JST model and its modified version.

that variable given the current values of all other variables and data. Letting the superscript $-t$ denote a quantity that excludes data from t^{th} position, the conditional posterior for z_t and l_t by marginalizing out the random variables φ, θ, and π is

$$P(z_t = j, l_t = k | \mathbf{w}, \mathbf{z^{-t}}, \mathbf{l^{-t}}, \alpha, \beta, \gamma) \propto \frac{N^{-t}_{w_t,j,k} + \beta}{N^{-t}_{j,k} + V\beta} \cdot \frac{N^{-t}_{j,k,d} + \alpha_{j,k}}{N^{-t}_{k,d} + \sum_j \alpha_{j,k}} \cdot \frac{N^{-t}_{k,d} + \gamma}{N^{-t}_{d} + S\gamma}. \quad (1)$$

where $N_{w_t,j,k}$ is the number of times word w_t appeared in topic j and with sentiment label k, $N_{j,k}$ is the number of times words assigned to topic j and sentiment label k, $N_{j,k,d}$ is the number of times a word from document d has been associated with topic j and sentiment label k, $N_{k,d}$ is the number of times sentiment label k has been assigned to some word tokens in document d, and N_d is the total number of words in the document collection.

In the modified JST model as shown in Figure **??**(b), we add an additional dependency link of $\boldsymbol{\varphi}$ on the matrix $\boldsymbol{\lambda}$ of size $S \times V$ which we use to encode word prior sentiment information into the JST model. For each word $w \in \{1, ..., V\}$, if w is found in the sentiment lexicon, for each $l \in \{1, ..., S\}$, the element λ_{lw} is updated as follows

$$\lambda_{lw} = \begin{cases} 1 & \text{if } S(w) = l \\ 0 & \text{otherwise} \end{cases}, \quad (2)$$

where the function $S(w)$ returns the prior sentiment label of w in a sentiment lexicon, i.e. neutral, posi-

	Book	DVD	Book	Elec.	Book	Kitch.	DVD	Elec.	DVD	Kitch.	Elec.	Kitch.
Pos.	recommend	funni	interest	pictur	interest	qualiti	concert	sound	movi	recommend	sound	pleas
	highli	cool	topic	clear	success	easili	rock	listen	stori	highli	excel	look
	easi	entertain	knowledg	paper	polit	servic	favorit	bass	classic	perfect	satisfi	worth
	depth	awesom	follow	color	clearli	stainless	sing	amaz	fun	great	perform	materi
	strong	worth	easi	accur	popular	safe	talent	acoust	charact	qulati	comfort	profession
Neg.	mysteri	cop	abus	problem	bore	return	bore	poorli	horror	cabinet	tomtom	elimin
	fbi	shock	question	poor	tediou	heavi	plot	low	alien	break	region	regardless
	investig	prison	mislead	design	cheat	stick	stupid	replac	scari	install	error	cheapli
	death	escap	point	case	crazi	defect	stori	avoid	evil	drop	code	plain
	report	dirti	disagre	flaw	hell	mess	terribl	crap	dead	gap	dumb	incorrect

Table 1: Extracted polarity words by JST on the combined data sets.

tive or negative.

The matrix $\boldsymbol{\lambda}$ can be considered as a transformation matrix which modifies the Dirichlet priors $\boldsymbol{\beta}$ of size $S \times T \times V$, so that the word prior polarity can be captured. For example, the word "*excellent*" with index i in the vocabulary has a positive polarity. The corresponding row vector in $\boldsymbol{\lambda}$ is $[0, 1, 0]$ with its elements representing neutral, positive, and negative. For each topic j, multiplying λ_{li} with β_{lji}, only the value of $\beta_{l_{pos}ji}$ is retained, and $\beta_{l_{neu}ji}$ and $\beta_{l_{neg}ji}$ are set to 0. Thus, the word "*excellent*" can only be drawn from the positive topic word distributions generated from a Dirichlet distribution with parameter $\boldsymbol{\beta}_{l_{pos}}$.

4 Polarity Words Extracted by JST

The JST model allows clustering different terms which share similar sentiment. In this section, we study the polarity-bearing topics extracted by JST. We combined reviews from the source and target domains and discarded document labels in both domains. There are a total of six different combinations. We then run JST on the combined data sets and listed some of the topic words extracted as shown in Table **??**. Words in each cell are grouped under one topic and the upper half of the table shows topic words under the positive sentiment label while the lower half shows topic words under the negative sentiment label.

We can see that JST appears to better capture sentiment association distribution in the source and target domains. For example, in the DVD+Elec. set, words from the DVD domain describe a rock concert DVD while words from the Electronics domain are likely relevant to stereo amplifiers and receivers, and yet they are grouped under the same topic by the JST model. Checking the word coverage in each domain reveals that for example "bass" seldom appears in the DVD domain, but appears more often in the Electronics domain. Likewise, in the Book+Kitch. set, "stainless" rarely appears in the Book domain and "interest" does not occur often in the Kitchen domain and they are grouped under the same topic. These observations motivate us to explore polarity-bearing topics extracted by JST for cross-domain sentiment classification since grouping words from different domains but bearing similar sentiment has the effect of overcoming the data distribution difference of two domains.

5 Domain Adaptation using JST

Given input data x and a class label y, labeled patterns of one domain can be drawn from the joint distribution $P(x, y) = P(y|x)P(x)$. Domain adaptation usually assume that data distribution are different in source and target domains, i.e., $P_s(x) \neq P_t(x)$. The task of domain adaptation is to predict the label y_i^t corresponding to x_i^t in the target domain.

We assume that we are given two sets of training data, $\mathcal{D}^s$ and $\mathcal{D}^t$, the *source domain* and *target domain* data sets, respectively. In the multiclass classification problem, the source domain data consist of labeled instances, $\mathcal{D}^s = \{(x_n^s; y_n^s) \in \mathcal{X} \times \mathcal{Y} : 1 \leq n \leq N^s\}$, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is a finite set of class labels. No class label is given in the target domain, $\mathcal{D}^t = \{x_n^t \in \mathcal{X} : 1 \leq n \leq N^t, N^t \gg N^s\}$. Algorithm **??** shows how to perform domain adaptation using the JST model. The source and target domain data are first merged with document labels discarded. A JST model is then

learned from the merged corpus to generate polarity-bearing topics for each document. The original documents in the source domain are augmented with those polarity-bearing topics as shown in Step 4 of Algorithm **??**, where $l_i\_z_i$ denotes a combination of sentiment label l_i and topic z_i for word w_i. Finally, feature selection is performed according to the information gain criteria and a classifier is then trained from the source domain using the new document representations. The target domain documents are also encoded in a similar way with polarity-bearing topics added into their feature representations.

Algorithm 1 Domain adaptation using JST.

Input: The source domain data $\mathcal{D}^s = \{(x_n^s; y_n^s) \in \mathcal{X} \times \mathcal{Y} : 1 \leq n \leq N^s\}$, the target domain data, $\mathcal{D}^t = \{x_n^t \in \mathcal{X} : 1 \leq n \leq N^t, N^t \gg N^s\}$

Output: A sentiment classifier for the target domain $\mathcal{D}^t$

1: Merge $\mathcal{D}^s$ and $\mathcal{D}^t$ with document labels discarded, $\mathcal{D} = \{(x_n^s, 1 \leq n \leq N^s; x_n^t, 1 \leq n \leq N^t\}$
2: Train a JST model on $\mathcal{D}$
3: **for** each document $x_n^s = (w_1, w_2, ..., w_m) \in \mathcal{D}^s$ **do**
4: Augment document with polarity-bearing topics generated from JST, $x_n^{s'} = (w_1, w_2, ..., w_m, l_1\_z_1, l_2\_z_2, ..., l_m\_z_m)$
5: Add $\{x_n^{s'}; y_n^s\}$ into a document pool $\mathcal{B}$
6: **end for**
7: Perform feature selection using IG on $\mathcal{B}$
8: Return a classifier, trained on $\mathcal{B}$

As discussed in Section **??** that the JST model directly models $P(\mathbf{l}|d)$, the probability of sentiment label given document, and hence document polarity can be classified accordingly. Since JST model learning does not require the availability of document labels, it is possible to augment the source domain data by adding most confident pseudo-labeled documents from the target domain by the JST model as shown in Algorithm **??**.

6 Experiments

We evaluate our proposed approach on the two datasets, the movie review (MR) data and the multi-domain sentiment (MDS) dataset. The movie review data consist of 1000 positive and 1000 negative movie reviews drawn from the IMDB movie archive while the multi-domain sentiment dataset contains four different types of product reviews extracted from Amazon.com including Book, DVD, Electronics and Kitchen appliances. Each category

Algorithm 2 Adding pseudo-labeled documents.

Input: The target domain data, $\mathcal{D}^t = \{x_n^t \in \mathcal{X} : 1 \leq n \leq N^t, N^t \gg N^s\}$, document sentiment classification threshold τ

Output: A labeled document pool $\mathcal{B}$

1: Train a JST model parameterized by Λ on $\mathcal{D}^t$
2: **for** each document $x_n^t \in \mathcal{D}^t$ **do**
3: Infer its sentiment class label from JST as $l_n = \arg\max_s P(l|x_n^t; \Lambda)$
4: **if** $P(l_n|x_n^t; \Lambda) > \tau$ **then**
5: Add labeled sample (x_n^t, l_n) into a document pool $\mathcal{B}$
6: **end if**
7: **end for**

of product reviews comprises of 1000 positive and 1000 negative reviews and is considered as a domain. Preprocessing was performed on both of the datasets by removing punctuation, numbers, non-alphabet characters and stopwords. The MPQA subjectivity lexicon is used as a sentiment lexicon in our experiments.

6.1 Experimental Setup

While the original JST model can produce reasonable results with a simple symmetric Dirichlet prior, here we use asymmetric prior $\boldsymbol{\alpha}$ over the topic proportions which is learned directly from data using a fixed-point iteration method (**?**).

In our experiment, $\boldsymbol{\alpha}$ was updated every 25 iterations during the Gibbs sampling procedure. In terms of other priors, we set symmetric prior $\beta = 0.01$ and $\gamma = (0.05 \times L)/S$, where L is the average document length, and the value of 0.05 on average allocates 5% of probability mass for mixing.

6.2 Supervised Sentiment Classification

We performed 5-fold cross validation for the performance evaluation of supervised sentiment classification. Results reported in this section are averaged over 10 such runs. We have tested several classifiers including Naïve Bayes (NB) and support vector machines (SVMs) from WEKA[5], and maximum entropy (ME) from MALLET[6]. All parameters are set to their default values except the Gaussian

[5] http://www.cs.waikato.ac.nz/ml/weka/
[6] http://mallet.cs.umass.edu/

prior variance is set to 0.1 for the ME model training. The results show that ME consistently outperforms NB and SVM on average. Thus, we only report results from ME trained on document vectors with each term weighted according to its frequency.

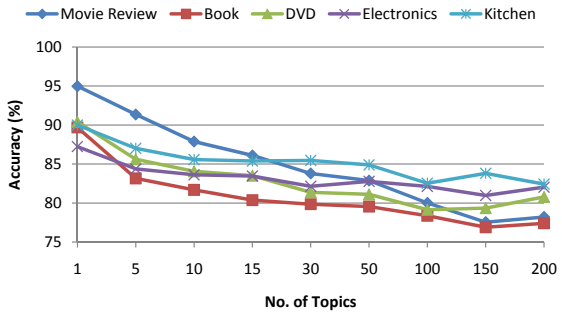


Figure 2: Classification accuracy vs. no. of topics.

The only parameter we need to set is the number of topics T. It has to be noted that the actual number of feature clusters is $3 \times T$. For example, when T is set to 5, there are 5 topic groups under each of the positive, negative, or neutral sentiment labels and hence there are altogether 15 feature clusters. The generated topics for each document from the JST model were simply added into its bag-of-words (BOW) feature representation prior to model training. Figure **??** shows the classification results on the five different domains by varying the number of topics from 1 to 200. It can be observed that the best classification accuracy is obtained when the number of topics is set to 1 (or 3 feature clusters). Increasing the number of topics results in the decrease of accuracy though it stabilizes after 15 topics. Nevertheless, when the number of topics is set to 15, using JST feature augmentation still outperforms ME without feature augmentation (the baseline model) in all of the domains. It is worth pointing out that the JST model with single topic becomes the standard LDA model with only three sentiment topics. Nevertheless, we have proposed an effective way to incorporate domain-independent word polarity prior information into model learning. As will be shown later in Table **??** that the JST model with word polarity priors incorporated performs significantly better than the LDA model without incorporating such prior information.

For comparison purpose, we also run the LDA model and augmented the BOW features with the generated topics in a similar way. The best accuracy was obtained when the number of topics is set to 15 in the LDA model. Table **??** shows the classification accuracy results with or without feature augmentation. We have performed significance test and found that LDA performs statistically significant better than Baseline according to a paired t-test with $p < 0.005$ for the Kitchen domain and with $p < 0.001$ for all the other domains. JST performs statistically significant better than both Baseline and LDA with $p < 0.001$.

Method	*MR*	*MDS*			
		Book	*DVD*	*Elec.*	*Kitch.*
Baseline	82.53	79.96	81.32	83.61	85.82
LDA	83.76	84.32	85.62	85.4	87.68
JST	**94.98**	**89.95**	**91.7**	**88.25**	**89.85**
[YE10]	91.78	82.75	82.85	84.55	87.9
[LI10]	-	79.49	81.65	83.64	85.65

Table 2: Supervised sentiment classification accuracy.

We also compare our method with other recently proposed approaches. Yessenalina et al. (**?**) explored different methods to automatically generate annotator rationales to improve sentiment classification accuracy. Our method using JST feature augmentation consistently performs better than their approach (denoted as [YE10] in Table **??**). They further proposed a two-level structured model (**?**) for document-level sentiment classification. The best accuracy obtained on the MR data is 93.22% with the model being initialized with sentence-level human annotations, which is still worse than ours. Li et al. (**?**) adopted a two-stage process by first classifying sentences as personal views and impersonal views and then using an ensemble method to perform sentiment classification. Their method (denoted as [LI10] in Table **??**) performs worse than either LDA or JST feature augmentation. To the best of our knowledge, the results achieved using JST feature augmentation are the state-of-the-art for both the MR and the MDS datasets.

6.3 Domain Adaptation

We conducted domain adaptation experiments on the MDS dataset comprising of four different domains, Book (B), DVD (D), Electronics (E), and Kitchen appliances (K). We randomly split each do-

main data into a training set of 1,600 instances and a test set of 400 instances. A classifier trained on the training set of one domain is tested on the test set of a different domain. We preformed 5 random splits and report the results averaged over 5 such runs.

Comparison with Baseline Models

We compare our proposed approaches with two baseline models. The first one (denoted as "Base" in Table **??**) is an ME classifier trained without adaptation. LDA results were generated from an ME classifier trained on document vectors augmented with topics generated from the LDA model. The number of topics was set to 15. JST results were obtained in a similar way except that we used the polarity-bearing topics generated from the JST model. We also tested with adding pseudo-labeled examples from the JST model into the source domain for ME classifier training (following Algorithm **??**), denoted as "JST-PL" in Table **??**. The document sentiment classification probability threshold τ was set to 0.8. Finally, we performed feature selection by selecting the top 2000 features according to the information gain criteria ("JST-IG")[7].

There are altogether 12 cross-domain sentiment classification tasks. We showed the adaptation loss results in Table **??** where the result for each domain and for each method is averaged over all three possible adaptation tasks by varying the source domain. The adaptation loss is calculated with respect to the in-domain gold standard classification result. For example, the in-domain goal standard for the Book domain is 79.96%. For adapting from DVD to Book, baseline achieves 72.25% and JST gives 76.45%. The adaptation loss is 7.71 for baseline and 3.51 for JST.

It can be observed from Table **??** that LDA only improves slightly compared to the baseline with an error reduction of 11%. JST further reduces the error due to transfer by 27%. Adding pseudo-labeled examples gives a slightly better performance compared to JST with an error reduction of 36%. With feature selection, JST-IG outperforms all the other approaches with a relative error reduction of 53%.

[7] Both values of 0.8 and 2000 were set arbitrarily after an initial run on some held-out data; they were not tuned to optimize test performance.

Domain	Base	LDA	JST	JST-PL	JST-IG
Book	10.8	9.4	7.2	6.3	**5.2**
DVD	8.3	6.1	4.8	4.4	**2.9**
Electr.	7.9	7.7	6.3	5.4	**3.9**
Kitch.	7.6	7.6	6.9	6.1	**4.4**
Average	8.6	7.7	6.3	5.5	**4.1**

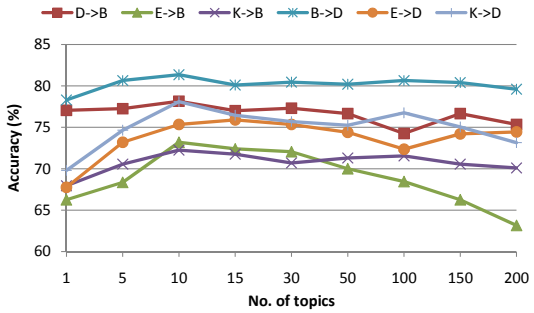
Table 3: Adaptation loss with respect to the in-domain gold standard. The last row shows the average loss over all the four domains.

Parameter Sensitivity

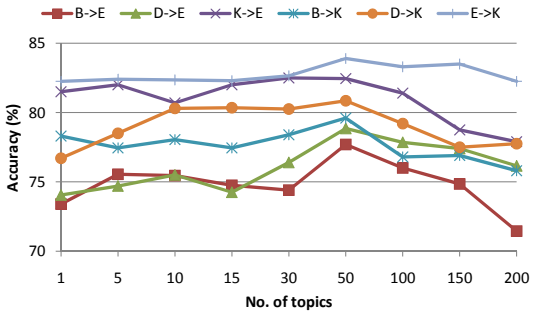
There is only one parameters to be set in the JST-IG approach, the number of topics. We plot the classification accuracy versus different topic numbers in Figure **??** with the number of topics varying between 1 and 200, corresponding to feature clusters varying between 3 and 600. It can be observed that for the relatively larger Book and DVD data sets, the accuracies peaked at topic number 10, whereas for the relatively smaller Electronics and Kitchen data sets, the best performance was obtained at topic number 50. Increasing topic numbers results in the decrease of classification accuracy. Manually examining the extracted polarity topics from JST reveals that when the topic number is small, each topic cluster contains well-mixed words from different domains. However, when the topic number is large, words under each topic cluster tend to be dominated by a single domain.

Comparison with Existing Approaches

We compare in Figure **??** our proposed approach with two other domain adaptation algorithms for sentiment classification, SCL and SFA. Each set of bars represent a cross-domain sentiment classification task. The thick horizontal lines are in-domain sentiment classification accuracies. It is worth noting that our in-domain results are slightly different from those reported in (**?**; **?**) due to different random splits. Our proposed JST-IG approach outperforms SCL in average and achieves comparable results to SFA. While SCL requires the construction of a reasonable number of auxiliary tasks that are useful to model "pivots" and "non-pivots", SFA relies on a good selection of domain-independent features for the construction of bipartite feature graph before running spectral clustering to derive feature clusters.

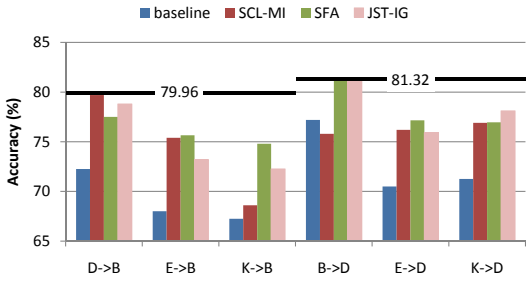


(a) Adapted to Book and DVD data sets.

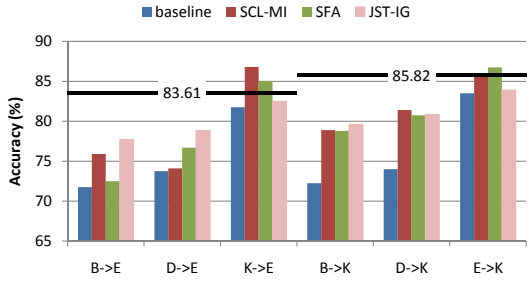


(b) Adapted to Electronics and Kitchen data sets.

Figure 3: Classification accuracy vs. no. of topics.



(a) Adapted to Book and DVD data sets.



(b) Adapted to Electronics and Kitchen data sets.

Figure 4: Comparison with existing approaches.

On the contrary, our proposed approach based on the JST model is much simpler and yet still achieves comparable results.

7 Conclusions

In this paper, we have studied polarity-bearing topics generated from the JST model and shown that by augmenting the original feature space with polarity-bearing topics, the in-domain supervised classifiers learned from augmented feature representation achieve the state-of-the-art performance on both the movie review data and the multi-domain sentiment dataset. Furthermore, using feature augmentation and selection according to the information gain criteria for cross-domain sentiment classification, our proposed approach outperforms SCL and gives similar results as SFA. Nevertheless, our approach is much simpler and does not require difficult parameter tuning.

There are several directions we would like to explore in the future. First, polarity-bearing topics generated by the JST model were simply added into the original feature space of documents, it is worth investigating attaching different weight to each topic maybe in proportional to the posterior probability of sentiment label and topic given a word estimated by the JST model. Second, it might be interesting to study the effect of introducing a tradeoff parameter to balance the effect of original and new features. Finally, our experimental results show that adding pseudo-labeled examples by the JST model does not appear to be effective. We could possibly explore instance weight strategies (**?**) on both pseudo-labeled examples and source domain training examples in order to improve the adaptation performance.

Acknowledgements

This work was supported in part by the EC-FP7 projects ROBUST (grant number 257859).

References

R.K. Ando and T. Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, 6:1817–1853.

A. Aue and M. Gamon. 2005. Customizing sentiment classifiers to new domains: a case study. In *Proceedings of Recent Advances in Natural Language Processing (RANLP)*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan.

2003. Latent Dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

J. Blitzer, M. Dredze, and F. Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, page 440–447.

C. Chelba and A. Acero. 2004. Adaptation of maximum entropy classifier: Little data can help a lot. In *EMNLP*.

W. Dai, Y. Chen, G.R. Xue, Q. Yang, and Y. Yu. 2008. Translated learning: Transfer learning across different feature spaces. In *NIPS*, pages 353–360.

W. Dai, O. Jin, G.R. Xue, Q. Yang, and Y. Yu. 2009. Eigentransfer: a unified framework for transfer learning. In *ICML*, pages 193–200.

H. Daumé III and D. Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26(1):101–126.

H. Daumé. 2007. Frustratingly easy domain adaptation. In *ACL*, pages 256–263.

J. Jiang and C.X. Zhai. 2007. Instance weighting for domain adaptation in NLP. In *ACL*, pages 264–271.

A. Kennedy and D. Inkpen. 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22(2):110–125.

S. Li, C.R. Huang, G. Zhou, and S.Y.M. Lee. 2010. Employing personal/impersonal views in supervised and semi-supervised sentiment classification. In *ACL*, pages 414–423.

C. Lin and Y. He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM international conference on Information and knowledge management (CIKM)*, pages 375–384.

C. Lin, Y. He, and R. Everson. 2010. A Comparative Study of Bayesian Models for Unsupervised Sentiment Detection. In *Proceedings of the 14th Conference on Computational Natural Language Learning (CoNLL)*, pages 144–152.

Ryan McDonald, Kerry Hannan, Tyler Neylon, Mike Wells, and Jeff Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. In *ACL*, pages 432–439.

T. Minka. 2003. Estimating a Dirichlet distribution. Technical report.

S.J. Pan, X. Ni, J.T. Sun, Q. Yang, and Z. Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World Wide Web (WWW)*, pages 751–760.

Bo Pang and Lillian Lee. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*, page 271–278.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *EMNLP*, pages 79–86.

B. Roark and M. Bacchiani. 2003. Supervised and unsupervised PCFG adaptation to novel domains. In *NAACL-HLT*, pages 126–133.

C.W. Seah, I. Tsang, Y.S. Ong, and K.K. Lee. 2010. Predictive Distribution Matching SVM for Multi-domain Learning. In *ECML-PKDD*, pages 231–247.

Casey Whitelaw, Navendu Garg, and Shlomo Argamon. 2005. Using appraisal groups for sentiment analysis. In *Proceedings of the ACM international conference on Information and Knowledge Management (CIKM)*, pages 625–631.

Q. Wu, S. Tan, and X. Cheng. 2009. Graph ranking for sentiment transfer. In *ACL-IJCNLP*, pages 317–320.

Q. Wu, S. Tan, X. Cheng, and M. Duan. 2010. MIEA: a Mutual Iterative Enhancement Approach for Cross-Domain Sentiment Classification. In *COLING*, page 1327-1335.

A. Yessenalina, Y. Choi, and C. Cardie. 2010a. Automatically generating annotator rationales to improve sentiment classification. In *ACL*, pages 336–341.

A. Yessenalina, Y. Yue, and C. Cardie. 2010b. Multi-Level Structured Models for Document-Level Sentiment Classification. In *EMNLP*, pages 1046–1056.

Jun Zhao, Kang Liu, and Gen Wang. 2008. Adding redundant features for CRFs-based sentence sentiment classification. In *EMNLP*, pages 117–126.

Using Multiple Sources to Construct a Sentiment Sensitive Thesaurus for Cross-Domain Sentiment Classification

Danushka Bollegala
The University of Tokyo
7-3-1, Hongo, Tokyo,
113-8656, Japan
danushka@
iba.t.u-tokyo.ac.jp

David Weir
School of Informatics
University of Sussex
Falmer, Brighton,
BN1 9QJ, UK
d.j.weir@
sussex.ac.uk

John Carroll
School of Informatics
University of Sussex
Falmer, Brighton,
BN1 9QJ, UK
j.a.carroll@
sussex.ac.uk

Abstract

We describe a sentiment classification method that is applicable when we do not have any labeled data for a *target* domain but have some labeled data for multiple other domains, designated as the *source* domains. We automatically create a *sentiment sensitive* thesaurus using both labeled and unlabeled data from multiple source domains to find the association between words that express similar sentiments in different domains. The created thesaurus is then used to expand feature vectors to train a binary classifier. Unlike previous cross-domain sentiment classification methods, our method can efficiently learn from *multiple source* domains. Our method significantly outperforms numerous baselines and returns results that are better than or comparable to previous cross-domain sentiment classification methods on a benchmark dataset containing Amazon user reviews for different types of products.

1 Introduction

Users express opinions about products or services they consume in blog posts, shopping sites, or review sites. It is useful for both consumers as well as for producers to know what general public think about a particular product or service. Automatic document level sentiment classification (Pang et al., 2002; Turney, 2002) is the task of classifying a given review with respect to the sentiment expressed by the author of the review. For example, a sentiment classifier might classify a user review about a movie as *positive* or *negative* depending on the sentiment expressed in the review. Sentiment classification has been applied in numerous tasks such as opinion mining (Pang and Lee, 2008), opinion summarization (Lu et al., 2009), contextual advertising (Fan and Chang, 2010), and market analysis (Hu and Liu, 2004).

Supervised learning algorithms that require labeled data have been successfully used to build sentiment classifiers for a specific domain (Pang et al., 2002). However, sentiment is expressed differently in different domains, and it is costly to annotate data for each new domain in which we would like to apply a sentiment classifier. For example, in the domain of reviews about *electronics* products, the words "durable" and "light" are used to express positive sentiment, whereas "expensive" and "short battery life" often indicate negative sentiment. On the other hand, if we consider the *books* domain the words "exciting" and "thriller" express positive sentiment, whereas the words "boring" and "lengthy" usually express negative sentiment. A classifier trained on one domain might not perform well on a different domain because it would fail to learn the sentiment of the unseen words.

Work in *cross-domain sentiment classification* (Blitzer et al., 2007) focuses on the challenge of training a classifier from one or more domains (source domains) and applying the trained classifier in a different domain (target domain). A cross-domain sentiment classification system must overcome two main challenges. First, it must identify which source domain features are related to which target domain features. Second, it requires a learning framework to incorporate the information re-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 132–141,
Portland, Oregon, June 19-24, 2011.

garding the relatedness of source and target domain features. Following previous work, we define cross-domain sentiment classification as the problem of learning a binary classifier (i.e. positive or negative sentiment) given a small set of labeled data for the source domain, and unlabeled data for both source and target domains. In particular, no labeled data is provided for the target domain.

In this paper, we describe a cross-domain sentiment classification method using an automatically created sentiment sensitive thesaurus. We use labeled data from multiple source domains and unlabeled data from source and target domains to represent the distribution of features. We represent a *lexical element* (i.e. a unigram or a bigram of word lemma) in a review using a feature vector. Next, for each lexical element we measure its relatedness to other lexical elements and group related lexical elements to create a thesaurus. The thesaurus captures the relatedness among lexical elements that appear in source and target domains based on the contexts in which the lexical elements appear (their distributional context). A distinctive aspect of our approach is that, in addition to the usual co-occurrence features typically used in characterizing a word's distributional context, we make use, where possible, of the sentiment label of a document: i.e. sentiment labels form part of our context features. This is what makes the distributional thesaurus sensitive to sentiment. Unlabeled data is cheaper to collect compared to labeled data and is often available in large quantities. The use of unlabeled data enables us to accurately estimate the distribution of words in source and target domains. Our method can learn from a large amount of unlabeled data to leverage a robust cross-domain sentiment classifier.

We model the cross-domain sentiment classification problem as one of *feature expansion*, where we append additional related features to feature vectors that represent source and target domain reviews in order to reduce the mismatch of features between the two domains. Methods that use related features have been successfully used in numerous tasks such as query expansion (Fang, 2008), and document classification (Shen et al., 2009). However, feature expansion techniques have not previously been applied to the task of cross-domain sentiment classification.

In our method, we use the automatically created thesaurus to *expand* feature vectors in a binary classifier at train and test times by introducing related lexical elements from the thesaurus. We use L1 regularized logistic regression as the classification algorithm. (However, the method is agnostic to the properties of the classifier and can be used to expand feature vectors for any binary classifier). L1 regularization enables us to select a small subset of features for the classifier. Unlike previous work which attempts to learn a cross-domain classifier using a single source domain, we leverage data from multiple source domains to learn a robust classifier that generalizes across multiple domains. Our contributions can be summarized as follows.

- We describe a fully automatic method to create a thesaurus that is sensitive to the sentiment of words expressed in different domains.
- We describe a method to use the created thesaurus to expand feature vectors at train and test times in a binary classifier.

2 A Motivating Example

To explain the problem of cross-domain sentiment classification, consider the reviews shown in Table 1 for the domains *books* and *kitchen appliances*. Table 1 shows two positive and one negative review from each domain. We have emphasized in boldface the words that express the sentiment of the authors of the reviews. We see that the words **excellent**, **broad**, **high quality**, **interesting**, and **well researched** are used to express positive sentiment in the books domain, whereas the word **disappointed** indicates negative sentiment. On the other hand, in the kitchen appliances domain the words **thrilled**, **high quality**, **professional**, **energy saving**, **lean**, and **delicious** express positive sentiment, whereas the words **rust** and **disappointed** express negative sentiment. Although **high quality** would express positive sentiment in both domains, and **disappointed** negative sentiment, it is unlikely that we would encounter **well researched** in kitchen appliances reviews, or **rust** or **delicious** in book reviews. Therefore, a model that is trained only using book reviews might not have any weights learnt for **delicious** or **rust**, which would make it difficult for this model to accurately classify reviews of kitchen appliances.

	books	*kitchen appliances*
+	**Excellent** and **broad** survey of the development of civilization with all the punch of **high quality** fiction.	I was so **thrilled** when I unpack my processor. It is so **high quality** and **professional** in both looks and performance.
+	This is an **interesting** and **well researched** book.	**Energy saving** grill. My husband loves the burgers that I make from this grill. They are **lean** and **delicious**.
-	Whenever a new book by Philippa Gregory comes out, I buy it hoping to have the same experience, and lately have been sorely **disappointed**.	These knives are already showing spots of **rust** despite washing by hand and drying. Very **disappointed**.

Table 1: Positive (+) and negative (-) sentiment reviews in two different domains.

sentence	Excellent and broad survey of the development of civilization.
POS tags	Excellent/JJ and/CC broad/JJ survey/NN1 of/IO the/AT development/NN1 of/IO civilization/NN1
lexical elements (unigrams)	excellent, broad, survey, development, civilization
lexical elements (bigrams)	excellent+broad, broad+survey, survey+development, development+civilization
sentiment features (lemma)	excellent*P, broad*P, survey*P, excellent+broad*P, broad+survey*P
sentiment features (POS)	JJ*P, NN1*P, JJ+NN1*P

Table 2: Generating lexical elements and sentiment features from a positive review sentence.

3 Sentiment Sensitive Thesaurus

One solution to the feature mismatch problem outlined above is to use a thesaurus that groups different words that express the same sentiment. For example, if we know that both *excellent* and *delicious* are positive sentiment words, then we can use this knowledge to *expand* a feature vector that contains the word *delicious* using the word *excellent*, thereby reducing the mismatch between features in a test instance and a trained model. Below we describe a method to construct a sentiment sensitive thesaurus for feature expansion.

Given a labeled or an unlabeled review, we first split the review into individual sentences. We carry out part-of-speech (POS) tagging and lemmatization on each review sentence using the RASP system (Briscoe et al., 2006). Lemmatization reduces the data sparseness and has been shown to be effective in text classification tasks (Joachims, 1998). We then apply a simple word filter based on POS tags to select content words (nouns, verbs, adjectives, and adverbs). In particular, previous work has identified adjectives as good indicators of sentiment (Hatzivassiloglou and McKeown, 1997; Wiebe, 2000). Following previous work in cross-domain sentiment classification, we model a review as a bag of words. We select unigrams and bigrams from each sentence. For the remainder of this paper, we will refer to unigrams and bigrams collectively as *lexical elements*. Previous work on sentiment classification has shown that both unigrams and bigrams are useful for training a sentiment classifier (Blitzer et al., 2007). We note that it is possible to create lexical elements both from source domain labeled reviews as well as from unlabeled reviews in source and target domains.

Next, we represent each lexical element u using a set of features as follows. First, we select other lexical elements that co-occur with u in a review sentence as features. Second, from each source domain labeled review sentence in which u occurs, we create *sentiment features* by appending the label of the review to each lexical element we generate from that review. For example, consider the sentence selected from a positive review of a book shown in Table 2. In Table 2, we use the notation "*P" to indicate positive sentiment features and "*N" to indicate negative sentiment features. The example sentence shown in Table 2 is selected from a positively labeled review, and generates positive sentiment features as shown in Table 2. In addition to word-level sentiment features, we replace words with their POS tags to create

POS-level sentiment features. POS tags generalize the word-level sentiment features, thereby reducing feature sparseness.

Let us denote the value of a feature w in the feature vector $\boldsymbol{u}$ representing a lexical element u by $f(\boldsymbol{u}, w)$. The vector $\boldsymbol{u}$ can be seen as a compact representation of the distribution of a lexical element u over the set of features that co-occur with u in the reviews. From the construction of the feature vector $\boldsymbol{u}$ described in the previous paragraph, it follows that w can be either a sentiment feature or another lexical element that co-occurs with u in some review sentence. The distributional hypothesis (Harris, 1954) states that words that have similar distributions are semantically similar. We compute $f(\boldsymbol{u}, w)$ as the pointwise mutual information between a lexical element u and a feature w as follows:

$$f(\boldsymbol{u}, w) = \log\left(\frac{\frac{c(u,w)}{N}}{\frac{\sum_{i=1}^{n} c(i,w)}{N} \times \frac{\sum_{j=1}^{m} c(u,j)}{N}}\right) \quad (1)$$

Here, $c(u, w)$ denotes the number of review sentences in which a lexical element u and a feature w co-occur, n and m respectively denote the total number of lexical elements and the total number of features, and $N = \sum_{i=1}^{n}\sum_{j=1}^{m} c(i, j)$. Pointwise mutual information is known to be biased towards infrequent elements and features. We follow the discounting approach of Pantel & Ravichandran (2004) to overcome this bias.

Next, for two lexical elements u and v (represented by feature vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, respectively), we compute the relatedness $\tau(v, u)$ of the feature v to the feature u as follows,

$$\tau(v, u) = \frac{\sum_{w \in \{x | f(\boldsymbol{v}, x) > 0\}} f(\boldsymbol{u}, w)}{\sum_{w \in \{x | f(\boldsymbol{u}, x) > 0\}} f(\boldsymbol{u}, w)}. \quad (2)$$

Here, we use the set notation $\{x | f(\boldsymbol{v}, x) > 0\}$ to denote the set of features that co-occur with v. Relatedness of a lexical element u to another lexical element v is the fraction of feature weights in the feature vector for the element u that also co-occur with the features in the feature vector for the element v. If there are no features that co-occur with both u and v, then the relatedness reaches its minimum value of 0. On the other hand if all features that co-occur with u also co-occur with v, then the relatedness , $\tau(v, u)$, reaches its maximum value of 1. Note that relatedness is an asymmetric measure by the definition given in Equation 2, and the relatedness $\tau(v, u)$ of an element v to another element u is not necessarily equal to $\tau(u, v)$, the relatedness of u to v.

We use the relatedness measure defined in Equation 2 to construct a *sentiment sensitive* thesaurus in which, for each lexical element u we list lexical elements v that co-occur with u (i.e. $f(\boldsymbol{u}, v) > 0$) in descending order of relatedness values $\tau(v, u)$. In the remainder of the paper, we use the term *base entry* to refer to a lexical element u for which its related lexical elements v (referred to as the *neighbors* of u) are listed in the thesaurus. Note that relatedness values computed according to Equation 2 are sensitive to sentiment labels assigned to reviews in the source domain, because co-occurrences are computed over both lexical and sentiment elements extracted from reviews. In other words, the relatedness of an element u to another element v depends upon the sentiment labels assigned to the reviews that generate u and v. This is an important fact that differentiates our sentiment-sensitive thesaurus from other distributional thesauri which do not consider sentiment information.

Moreover, we only need to retain lexical elements in the sentiment sensitive thesaurus because when predicting the sentiment label for target reviews (at test time) we cannot generate sentiment elements from those (unlabeled) reviews, therefore we are not required to find expansion candidates for sentiment elements. However, we emphasize the fact that the relatedness values between the lexical elements listed in the sentiment-sensitive thesaurus are computed using co-occurrences with both lexical and sentiment features, and therefore the expansion candidates selected for the lexical elements in the target domain reviews are sensitive to sentiment labels assigned to reviews in the source domain. Using a sparse matrix format and approximate similarity matching techniques (Sarawagi and Kirpal, 2004), we can efficiently create a thesaurus from a large set of reviews.

4 Feature Expansion

Our *feature expansion* phase augments a feature vector with additional related features selected from the

sentiment-sensitive thesaurus created in Section 3 to overcome the feature mismatch problem. First, following the bag-of-words model, we model a review d using the set $\{w_1, \ldots, w_N\}$, where the elements w_i are either unigrams or bigrams that appear in the review d. We then represent a review d by a real-valued term-frequency vector $\boldsymbol{d} \in \mathbb{R}^N$, where the value of the j-th element d_j is set to the total number of occurrences of the unigram or bigram w_j in the review d. To find the suitable candidates to expand a vector $\boldsymbol{d}$ for the review d, we define a ranking score $\text{score}(u_i, \boldsymbol{d})$ for each base entry in the thesaurus as follows:

$$\text{score}(u_i, \boldsymbol{d}) = \frac{\sum_{j=1}^{N} d_j \tau(w_j, u_i)}{\sum_{l=1}^{N} d_l} \quad (3)$$

According to this definition, given a review d, a base entry u_i will have a high ranking score if there are many words w_j in the review d that are also listed as neighbors for the base entry u_i in the sentiment-sensitive thesaurus. Moreover, we weight the relatedness scores for each word w_j by its normalized term-frequency to emphasize the salient unigrams and bigrams in a review. Recall that relatedness is defined as an asymmetric measure in Equation 2, and we use $\tau(w_j, u_i)$ in the computation of $\text{score}(u_i, \boldsymbol{d})$ in Equation 3. This is particularly important because we would like to score base entries u_i considering *all* the unigrams and bigrams that appear in a review d, instead of considering each unigram or bigram individually.

To expand a vector, $\boldsymbol{d}$, for a review d, we first rank the base entries, u_i using the ranking score in Equation 3 and select the top k ranked base entries. Let us denote the r-th ranked ($1 \leq r \leq k$) base entry for a review d by v_d^r. We then extend the original set of unigrams and bigrams $\{w_1, \ldots, w_N\}$ by the base entries $v_d^1, \ldots, v_d^k$ to create a new vector $\boldsymbol{d'} \in \mathbb{R}^{(N+k)}$ with dimensions corresponding to $w_1, \ldots, w_N, v_d^1, \ldots, v_d^k$ for a review d. The values of the extended vector $\boldsymbol{d'}$ are set as follows. The values of the first N dimensions that correspond to unigrams and bigrams w_i that occur in the review d are set to d_i, their frequency in d. The subsequent k dimensions that correspond to the top ranked based entries for the review d are weighted according to their ranking score. Specifically, we set the value of the r-th ranked base entry v_d^r to $1/r$. Alternatively, one could use the ranking score, $\text{score}(v_d^r, d)$, itself as the value of the appended base entries. However, both relatedness scores as well as normalized term-frequencies can be small in practice, which leads to very small absolute ranking scores. By using the inverse rank, we only take into account the relative ranking of base entries and ignore their absolute scores.

Note that the score of a base entry depends on a review d. Therefore, we select different base entries as additional features for expanding different reviews. Furthermore, we do *not* expand each w_i individually when expanding a vector $\boldsymbol{d}$ for a review. Instead, we consider *all* unigrams and bigrams in d when selecting the base entries for expansion. One can think of the feature expansion process as a lower dimensional latent mapping of features onto the space spanned by the base entries in the sentiment-sensitive thesaurus. The asymmetric property of the relatedness (Equation 2) implicitly prefers common words that co-occur with numerous other words as expansion candidates. Such words act as domain independent pivots and enable us to transfer the information regarding sentiment from one domain to another.

Using the extended vectors $\boldsymbol{d'}$ to represent reviews, we train a binary classifier from the source domain labeled reviews to predict positive and negative sentiment in reviews. We differentiate the appended base entries v_d^r from w_i that existed in the original vector $\boldsymbol{d}$ (prior to expansion) by assigning different feature identifiers to the appended base entries. For example, a unigram *excellent* in a feature vector is differentiated from the base entry *excellent* by assigning the feature id, "BASE=*excellent*" to the latter. This enables us to learn different weights for base entries depending on whether they are useful for expanding a feature vector. We use L1 regularized logistic regression as the classification algorithm (Ng, 2004), which produces a sparse model in which most irrelevant features are assigned a zero weight. This enables us to select useful features for classification in a systematic way without having to preselect features using heuristic approaches. The regularization parameter is set to its default value of 1 for all the experiments described in this paper.

5 Experiments

5.1 Dataset

To evaluate our method we use the cross-domain sentiment classification dataset prepared by Blitzer et al. (2007). This dataset consists of Amazon product reviews for four different product types: books (**B**), DVDs (**D**), electronics (**E**) and kitchen appliances (**K**). There are 1000 positive and 1000 negative labeled reviews for each domain. Moreover, the dataset contains some unlabeled reviews (on average $17,547$) for each domain. This benchmark dataset has been used in much previous work on cross-domain sentiment classification and by evaluating on it we can directly compare our method against existing approaches.

Following previous work, we randomly select 800 positive and 800 negative labeled reviews from each domain as training instances (i.e. $1600 \times 4 = 6400$); the remainder is used for testing (i.e. $400 \times 4 = 1600$). In our experiments, we select each domain in turn as the target domain, with one or more other domains as sources. Note that when we combine more than one source domain we limit the total number of source domain labeled reviews to 1600, balanced between the domains. For example, if we combine two source domains, then we select 400 positive and 400 negative labeled reviews from each domain giving $(400 + 400) \times 2 = 1600$. This enables us to perform a fair evaluation when combining multiple source domains. The evaluation metric is classification accuracy on a target domain, computed as the percentage of correctly classified target domain reviews out of the total number of reviews in the target domain.

5.2 Effect of Feature Expansion

To study the effect of feature expansion at train time compared to test time, we used Amazon reviews for two further domains, *music* and *video*, which were also collected by Blitzer et al. (2007) but are not part of the benchmark dataset. Each validation domain has 1000 positive and 1000 negative labeled reviews, and 15000 unlabeled reviews. Using the validation domains as targets, we vary the number of top k ranked base entries (Equation 3) used for feature expansion during training (Train_k) and testing (Test_k), and measure the average classification accuracy. Figure 1 illustrates the results using a *heat map*, where dark colors indicate low accuracy values and light colors indicate high accuracy values. We see that expanding features only at test time (the left-most column) does not work well because we have not learned proper weights for the additional features. Similarly, expanding features only at train time (the bottom-most row) also does not perform well because the expanded features are not used during testing. The maximum classification accuracy is obtained when $\text{Test}_k = 400$ and $\text{Train}_k = 800$, and we use these values for the remainder of the experiments described in the paper.

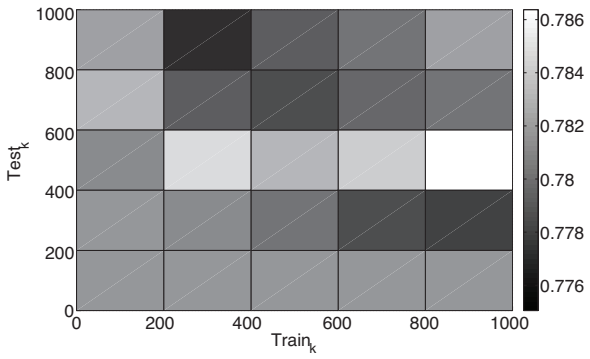


Figure 1: Feature expansion at train vs. test times.

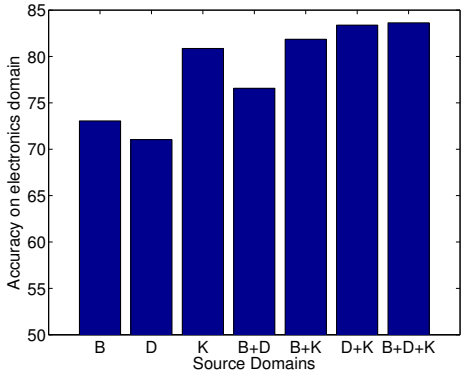


Figure 2: Effect of using multiple source domains.

5.3 Combining Multiple Sources

Figure 2 shows the effect of combining multiple source domains to build a sentiment classifier for the electronics domain. We see that the kitchen domain is the single best source domain when adapting to the electronics target domain. This behavior

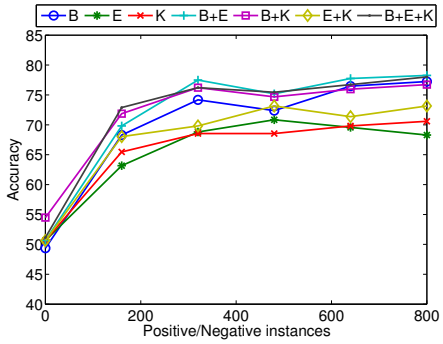


Figure 3: Effect of source domain labeled data.

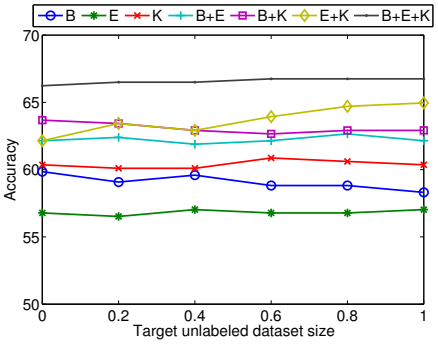


Figure 5: Effect of target domain unlabeled data.

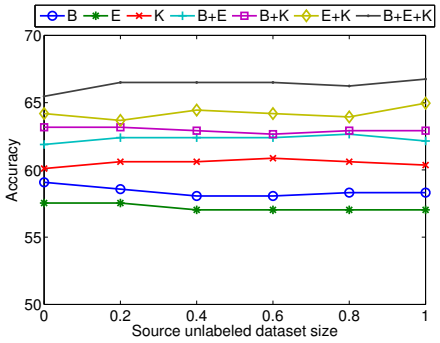


Figure 4: Effect of source domain unlabeled data.

is explained by the fact that in general kitchen appliances and electronic items have similar aspects. But a more interesting observation is that the accuracy that we obtain when we use two source domains is always greater than the accuracy if we use those domains individually. The highest accuracy is achieved when we use all three source domains. Although not shown here for space limitations, we observed similar trends with other domains in the benchmark dataset.

To investigate the impact of the quantity of source domain labeled data on our method, we vary the amount of data from zero to 800 reviews, with equal amounts of positive and negative labeled data. Figure 3 shows the accuracy with the DVD domain as the target. Note that source domain labeled data is used both to create the sentiment sensitive thesaurus as well as to train the sentiment classifier. When there are multiple source domains we limit and balance the number of labeled instances as outlined in Section 5.1. The amount of unlabeled data is held constant, so that any change in classification accuracy is directly attributable to the source domain labeled instances. Because this is a binary classification task (i.e. positive vs. negative sentiment), a random classifier that does not utilize any labeled data would report a 50% classification accuracy. From Figure 3, we see that when we increase the amount of source domain labeled data the accuracy increases quickly. In fact, by selecting only 400 (i.e. 50% of the total 800) labeled instances per class, we achieve the maximum performance in most of the cases.

To study the effect of source and target domain unlabeled data on the performance of our method, we create sentiment sensitive thesauri using different proportions of unlabeled data. The amount of labeled data is held constant and is balanced across multiple domains as outlined in Section 5.1, so any changes in classification accuracy can be directly attributed to the contribution of unlabeled data. Figure 4 shows classification accuracy on the DVD target domain when we vary the proportion of source domain unlabeled data (target domain's unlabeled data is fixed).

Likewise, Figure 5 shows the classification accuracy on the DVD target domain when we vary the proportion of the target domain's unlabeled data (source domains' unlabeled data is fixed). From Figures 4 and 5, we see that irrespective of the amount being used, there is a clear performance gain when we use unlabeled data from multiple source domains compared to using a single source domain. However, we could not observe a clear gain in performance when we increase the amount of the unlabeled data used to create the sentiment sensitive thesaurus.

Method	K	D	E	B
No Thesaurus	72.61	68.97	70.53	62.72
SCL	80.83	74.56	78.43	72.76
SCL-MI	82.06	76.30	78.93	74.56
SFA	81.48	76.31	75.30	**77.73**
LSA	79.00	73.50	77.66	70.83
FALSA	80.83	76.33	77.33	73.33
NSS	77.50	73.50	75.50	71.46
Proposed	**85.18**	**78.77**	**83.63**	76.32
Within-Domain	*87.70*	*82.40*	*84.40*	*80.40*

Table 3: Cross-domain sentiment classification accuracy.

5.4 Cross-Domain Sentiment Classification

Table 3 compares our method against a number of baselines and previous cross-domain sentiment classification techniques using the benchmark dataset. For all previous techniques we give the results reported in the original papers. The **No Thesaurus** baseline simulates the effect of not performing any feature expansion. We simply train a binary classifier using unigrams and bigrams as features from the labeled reviews in the source domains and apply the trained classifier on the target domain. This can be considered to be a lower bound that does not perform domain adaptation. **SCL** is the structural correspondence learning technique of Blitzer et al. (2006). In **SCL-MI**, features are selected using the mutual information between a feature (unigram or bigram) and a domain label. After selecting salient features, the SCL algorithm is used to train a binary classifier. **SFA** is the spectral feature alignment technique of Pan et al. (2010). Both the **LSA** and **FALSA** techniques are based on latent semantic analysis (Pan et al., 2010). For the **Within-Domain** baseline, we train a binary classifier using the labeled data from the target domain. This upper baseline represents the classification accuracy we could hope to obtain if we were to have labeled data for the target domain. Note that this is not a cross-domain classification setting. To evaluate the benefit of using sentiment features on our method, we give a **NSS** (non-sentiment sensitive) baseline in which we create a thesaurus without using any sentiment features. **Proposed** is our method.

From Table 3, we see that our **proposed** method returns the best cross-domain sentiment classification accuracy (shown in boldface) for the three domains kitchen appliances, DVDs, and electronics. For the books domain, the best results are returned by **SFA**. The books domain has the lowest number of unlabeled reviews (around 5000) in the dataset. Because our method relies upon the availability of unlabeled data for the construction of a sentiment sensitive thesaurus, we believe that this accounts for our lack of performance on the books domain. However, given that it is much cheaper to obtain unlabeled than labeled data for a target domain, there is strong potential for improving the performance of our method in this domain. The analysis of variance (ANOVA) and Tukey's honestly significant differences (HSD) tests on the classification accuracies for the four domains show that our method is statistically significantly better than both the **No Thesaurus** and **NSS** baselines, at confidence level 0.05. We therefore conclude that using the sentiment sensitive thesaurus for feature expansion is useful for cross-domain sentiment classification. The results returned by our method are comparable to state-of-the-art techniques such as **SCL-MI** and **SFA**. In particular, the differences between those techniques and our method are not statistically significant.

6 Related Work

Compared to single-domain sentiment classification, which has been studied extensively in previous work (Pang and Lee, 2008; Turney, 2002), cross-domain sentiment classification has only recently received attention in response to advances in the area of domain adaptation. Aue and Gammon (2005) report a number of empirical tests into domain adaptation of sentiment classifiers using an ensemble of classifiers. However, most of these tests were unable to outperform a simple baseline classifier that is trained using all labeled data for all domains.

Blitzer et al. (2007) apply the structural correspondence learning (SCL) algorithm to train a cross-domain sentiment classifier. They first chooses a set of *pivot features* using pointwise mutual information between a feature and a domain label. Next, linear predictors are learnt to predict the occurrences of those pivots. Finally, they use singular value decomposition (SVD) to construct a lower-dimensional feature space in which a binary classi-

fier is trained. The selection of pivots is vital to the performance of SCL and heuristically selected pivot features might not guarantee the best performance on target domains. In contrast, our method uses all features when creating the thesaurus and selects a subset of features during training using L1 regularization. Moreover, we do not require SVD, which has cubic time complexity so can be computationally expensive for large datasets.

Pan et al. (2010) use structural feature alignment (SFA) to find an alignment between domain specific and domain independent features. The mutual information of a feature with domain labels is used to classify domain specific and domain independent features. Next, spectral clustering is performed on a bipartite graph that represents the relationship between the two sets of features. Finally, the top eigenvectors are selected to construct a lower-dimensional projection. However, not all words can be cleanly classified into domain specific or domain independent, and this process is conducted prior to training a classifier. In contrast, our method lets a particular lexical entry to be listed as a neighour for multiple base entries. Moreover, we expand each feature vector individually and do not require any clustering. Furthermore, unlike SCL and SFA, which consider a single source domain, our method can efficiently adapt from multiple source domains.

7 Conclusions

We have described and evaluated a method to construct a sentiment-sensitive thesaurus to bridge the gap between source and target domains in cross-domain sentiment classification using multiple source domains. Experimental results using a benchmark dataset for cross-domain sentiment classification show that our proposed method can improve classification accuracy in a sentiment classifier. In future, we intend to apply the proposed method to other domain adaptation tasks.

Acknowledgements

This research was conducted while the first author was a visiting research fellow at Sussex university under the postdoctoral fellowship of the Japan Society for the Promotion of Science (JSPS).

References

Anthony Aue and Michael Gamon. 2005. Customizing sentiment classifiers to new domains: a case study. Technical report, Microsoft Research.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP 2006*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL 2007*, pages 440–447.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the rasp system. In *COLING/ACL 2006 Interactive Presentation Sessions*.

Teng-Kai Fan and Chia-Hui Chang. 2010. Sentiment-oriented contextual advertising. *Knowledge and Information Systems*, 23(3):321–344.

Hui Fang. 2008. A re-examination of query expansion using lexical resources. In *ACL 2008*, pages 139–147.

Z. Harris. 1954. Distributional structure. *Word*, 10:146–162.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *ACL 1997*, pages 174–181.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD 2004*, pages 168–177.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *ECML 1998*, pages 137–142.

Yue Lu, ChengXiang Zhai, and Neel Sundaresan. 2009. Rated aspect summarization of short comments. In *WWW 2009*, pages 131–140.

Andrew Y. Ng. 2004. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *ICML 2004*.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *WWW 2010*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *EMNLP 2002*, pages 79–86.

Patrick Pantel and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In *NAACL-HLT'04*, pages 321 – 328.

Sunita Sarawagi and Alok Kirpal. 2004. Efficient set joins on similarity predicates. In *SIGMOD '04*, pages 743–754.

Dou Shen, Jianmin Wu, Bin Cao, Jian-Tao Sun, Qiang Yang, Zheng Chen, and Ying Li. 2009. Exploiting term relationship to boost text classification. In *CIKM'09*, pages 1637 – 1640.

Peter D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *ACL 2002*, pages 417–424.

Janyce M. Wiebe. 2000. Learning subjective adjective from corpora. In *AAAI 2000*, pages 735–740.

Learning Word Vectors for Sentiment Analysis

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang,
Andrew Y. Ng, and **Christopher Potts**
Stanford University
Stanford, CA 94305
[amaas, rdaly, ptpham, yuze, ang, cgpotts]@stanford.edu

Abstract

Unsupervised vector-based approaches to semantics can model rich lexical meanings, but they largely fail to capture sentiment information that is central to many word meanings and important for a wide range of NLP tasks. We present a model that uses a mix of unsupervised and supervised techniques to learn word vectors capturing semantic term–document information as well as rich sentiment content. The proposed model can leverage both continuous and multi-dimensional sentiment information as well as non-sentiment annotations. We instantiate the model to utilize the document-level sentiment polarity annotations present in many online documents (e.g. star ratings). We evaluate the model using small, widely used sentiment and subjectivity corpora and find it out-performs several previously introduced methods for sentiment classification. We also introduce a large dataset of movie reviews to serve as a more robust benchmark for work in this area.

1 Introduction

Word representations are a critical component of many natural language processing systems. It is common to represent words as indices in a vocabulary, but this fails to capture the rich relational structure of the lexicon. Vector-based models do much better in this regard. They encode continuous similarities between words as distance or angle between word vectors in a high-dimensional space. The general approach has proven useful in tasks such as word sense disambiguation, named entity recognition, part of speech tagging, and document retrieval (Turney and Pantel, 2010; Collobert and Weston, 2008; Turian et al., 2010).

In this paper, we present a model to capture both semantic and sentiment similarities among words. The semantic component of our model learns word vectors via an unsupervised probabilistic model of documents. However, in keeping with linguistic and cognitive research arguing that expressive content and descriptive semantic content are distinct (Kaplan, 1999; Jay, 2000; Potts, 2007), we find that this basic model misses crucial sentiment information. For example, while it learns that *wonderful* and *amazing* are semantically close, it doesn't capture the fact that these are both very strong positive sentiment words, at the opposite end of the spectrum from *terrible* and *awful*.

Thus, we extend the model with a supervised sentiment component that is capable of embracing many social and attitudinal aspects of meaning (Wilson et al., 2004; Alm et al., 2005; Andreevskaia and Bergler, 2006; Pang and Lee, 2005; Goldberg and Zhu, 2006; Snyder and Barzilay, 2007). This component of the model uses the vector representation of words to predict the sentiment annotations on contexts in which the words appear. This causes words expressing similar sentiment to have similar vector representations. The full objective function of the model thus learns semantic vectors that are imbued with nuanced sentiment information. In our experiments, we show how the model can leverage document-level sentiment annotations of a sort that are abundant online in the form of consumer reviews for movies, products, etc. The technique is suffi-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 142–150,
Portland, Oregon, June 19-24, 2011.

ciently general to work also with continuous and multi-dimensional notions of sentiment as well as non-sentiment annotations (e.g., political affiliation, speaker commitment).

After presenting the model in detail, we provide illustrative examples of the vectors it learns, and then we systematically evaluate the approach on document-level and sentence-level classification tasks. Our experiments involve the small, widely used sentiment and subjectivity corpora of Pang and Lee (2004), which permits us to make comparisons with a number of related approaches and published results. We also show that this dataset contains many correlations between examples in the training and testing sets. This leads us to evaluate on, and make publicly available, a large dataset of informal movie reviews from the Internet Movie Database (IMDB).

2 Related work

The model we present in the next section draws inspiration from prior work on both probabilistic topic modeling and vector-spaced models for word meanings.

Latent Dirichlet Allocation (LDA; (Blei et al., 2003)) is a probabilistic document model that assumes each document is a mixture of latent topics. For each latent topic T, the model learns a conditional distribution $p(w|T)$ for the probability that word w occurs in T. One can obtain a k-dimensional vector representation of words by first training a k-topic model and then filling the matrix with the $p(w|T)$ values (normalized to unit length). The result is a word–topic matrix in which the rows are taken to represent word meanings. However, because the emphasis in LDA is on modeling topics, not word meanings, there is no guarantee that the row (word) vectors are sensible as points in a k-dimensional space. Indeed, we show in section 4 that using LDA in this way does not deliver robust word vectors. The semantic component of our model shares its probabilistic foundation with LDA, but is factored in a manner designed to discover word vectors rather than latent topics. Some recent work introduces extensions of LDA to capture sentiment in addition to topical information (Li et al., 2010; Lin and He, 2009; Boyd-Graber and Resnik, 2010). Like LDA, these methods focus on modeling sentiment-imbued topics rather than embedding words in a vector space.

Vector space models (VSMs) seek to model words directly (Turney and Pantel, 2010). Latent Semantic Analysis (LSA), perhaps the best known VSM, explicitly learns semantic word vectors by applying singular value decomposition (SVD) to factor a term–document co-occurrence matrix. It is typical to weight and normalize the matrix values prior to SVD. To obtain a k-dimensional representation for a given word, only the entries corresponding to the k largest singular values are taken from the word's basis in the factored matrix. Such matrix factorization-based approaches are extremely successful in practice, but they force the researcher to make a number of design choices (weighting, normalization, dimensionality reduction algorithm) with little theoretical guidance to suggest which to prefer.

Using term frequency (tf) and inverse document frequency (idf) weighting to transform the values in a VSM often increases the performance of retrieval and categorization systems. Delta idf weighting (Martineau and Finin, 2009) is a supervised variant of idf weighting in which the idf calculation is done for each document class and then one value is subtracted from the other. Martineau and Finin present evidence that this weighting helps with sentiment classification, and Paltoglou and Thelwall (2010) systematically explore a number of weighting schemes in the context of sentiment analysis. The success of delta idf weighting in previous work suggests that incorporating sentiment information into VSM values via supervised methods is helpful for sentiment analysis. We adopt this insight, but we are able to incorporate it directly into our model's objective function. (Section 4 compares our approach with a representative sample of such weighting schemes.)

3 Our Model

To capture semantic similarities among words, we derive a probabilistic model of documents which learns word representations. This component does not require labeled data, and shares its foundation with probabilistic topic models such as LDA. The sentiment component of our model uses sentiment annotations to constrain words expressing similar

sentiment to have similar representations. We can efficiently learn parameters for the joint objective function using alternating maximization.

3.1 Capturing Semantic Similarities

We build a probabilistic model of a document using a continuous mixture distribution over words indexed by a multi-dimensional random variable θ. We assume words in a document are conditionally independent given the mixture variable θ. We assign a probability to a document d using a joint distribution over the document and θ. The model assumes each word $w_i \in d$ is conditionally independent of the other words given θ. The probability of a document is thus

$$p(d) = \int p(d, \theta) d\theta = \int p(\theta) \prod_{i=1}^{N} p(w_i|\theta) d\theta. \quad (1)$$

Where N is the number of words in d and w_i is the i^{th} word in d. We use a Gaussian prior on θ.

We define the conditional distribution $p(w_i|\theta)$ using a log-linear model with parameters R and b. The energy function uses a word representation matrix $R \in \mathbb{R}^{(\beta \times |V|)}$ where each word w (represented as a one-on vector) in the vocabulary V has a β-dimensional vector representation $\phi_w = Rw$ corresponding to that word's column in R. The random variable θ is also a β-dimensional vector, $\theta \in \mathbb{R}^{\beta}$ which weights each of the β dimensions of words' representation vectors. We additionally introduce a bias b_w for each word to capture differences in overall word frequencies. The energy assigned to a word w given these model parameters is

$$E(w; \theta, \phi_w, b_w) = -\theta^T \phi_w - b_w. \quad (2)$$

To obtain the distribution $p(w|\theta)$ we use a softmax,

$$p(w|\theta; R, b) = \frac{\exp(-E(w; \theta, \phi_w, b_w))}{\sum_{w' \in V} \exp(-E(w'; \theta, \phi_{w'}, b_{w'}))} \quad (3)$$

$$= \frac{\exp(\theta^T \phi_w + b_w)}{\sum_{w' \in V} \exp(\theta^T \phi_{w'} + b_{w'})}. \quad (4)$$

The number of terms in the denominator's summation grows linearly in $|V|$, making exact computation of the distribution possible. For a given θ, a word w's occurrence probability is related to how closely its representation vector ϕ_w matches the scaling direction of θ. This idea is similar to the word vector inner product used in the log-bilinear language model of Mnih and Hinton (2007).

Equation 1 resembles the probabilistic model of LDA (Blei et al., 2003), which models documents as mixtures of latent topics. One could view the entries of a word vector ϕ as that word's association strength with respect to each latent topic dimension. The random variable θ then defines a weighting over topics. However, our model does not attempt to model individual topics, but instead directly models word probabilities conditioned on the topic mixture variable θ. Because of the log-linear formulation of the conditional distribution, θ is a vector in $\mathbb{R}^{\beta}$ and not restricted to the unit simplex as it is in LDA.

We now derive maximum likelihood learning for this model when given a set of unlabeled documents D. In maximum likelihood learning we maximize the probability of the observed data given the model parameters. We assume documents $d_k \in D$ are i.i.d. samples. Thus the learning problem becomes

$$\max_{R,b} p(D; R, b) = \prod_{d_k \in D} \int p(\theta) \prod_{i=1}^{N_k} p(w_i|\theta; R, b) d\theta. \quad (5)$$

Using *maximum a posteriori* (MAP) estimates for θ, we approximate this learning problem as

$$\max_{R,b} \prod_{d_k \in D} p(\hat{\theta}_k) \prod_{i=1}^{N_k} p(w_i|\hat{\theta}_k; R, b), \quad (6)$$

where $\hat{\theta}_k$ denotes the MAP estimate of θ for d_k. We introduce a Frobenious norm regularization term for the word representation matrix R. The word biases b are not regularized reflecting the fact that we want the biases to capture whatever overall word frequency statistics are present in the data. By taking the logarithm and simplifying we obtain the final objective,

$$\nu ||R||_F^2 + \sum_{d_k \in D} \lambda ||\hat{\theta}_k||_2^2 + \sum_{i=1}^{N_k} \log p(w_i|\hat{\theta}_k; R, b), \quad (7)$$

which is maximized with respect to R and b. The hyper-parameters in the model are the regularization

weights (λ and ν), and the word vector dimensionality β.

3.2 Capturing Word Sentiment

The model presented so far does not explicitly capture sentiment information. Applying this algorithm to documents will produce representations where words that occur together in documents have similar representations. However, this unsupervised approach has no explicit way of capturing which words are predictive of sentiment as opposed to content-related. Much previous work in natural language processing achieves better representations by learning from multiple tasks (Collobert and Weston, 2008; Finkel and Manning, 2009). Following this theme we introduce a second task to utilize labeled documents to improve our model's word representations.

Sentiment is a complex, multi-dimensional concept. Depending on which aspects of sentiment we wish to capture, we can give some body of text a sentiment label s which can be categorical, continuous, or multi-dimensional. To leverage such labels, we introduce an objective that the word vectors of our model should predict the sentiment label using some appropriate predictor,

$$\hat{s} = f(\phi_w). \tag{8}$$

Using an appropriate predictor function $f(x)$ we map a word vector ϕ_w to a predicted sentiment label $\hat{s}$. We can then improve our word vector ϕ_w to better predict the sentiment labels of contexts in which that word occurs.

For simplicity we consider the case where the sentiment label s is a scalar continuous value representing sentiment polarity of a document. This captures the case of many online reviews where documents are associated with a label on a star rating scale. We linearly map such star values to the interval $s \in [0, 1]$ and treat them as a probability of positive sentiment polarity. Using this formulation, we employ a logistic regression as our predictor $f(x)$. We use w's vector representation ϕ_w and regression weights ψ to express this as

$$p(s = 1|w; R, \psi) = \sigma(\psi^T \phi_w + b_c), \tag{9}$$

where $\sigma(x)$ is the logistic function and $\psi \in \mathbb{R}^\beta$ is the logistic regression weight vector. We additionally introduce a scalar bias b_c for the classifier.

The logistic regression weights ψ and b_c define a linear hyperplane in the word vector space where a word vector's positive sentiment probability depends on where it lies with respect to this hyperplane. Learning over a collection of documents results in words residing different distances from this hyperplane based on the average polarity of documents in which the words occur.

Given a set of labeled documents D where s_k is the sentiment label for document d_k, we wish to maximize the probability of document labels given the documents. We assume documents in the collection and words within a document are i.i.d. samples. By maximizing the log-objective we obtain,

$$\max_{R,\psi,b_c} \sum_{k=1}^{|D|} \sum_{i=1}^{N_k} \log p(s_k|w_i; R, \psi, b_c). \tag{10}$$

The conditional probability $p(s_k|w_i; R, \psi, b_c)$ is easily obtained from equation 9.

3.3 Learning

The full learning objective maximizes a sum of the two objectives presented. This produces a final objective function of,

$$\nu||R||_F^2 + \sum_{k=1}^{|D|} \lambda||\hat{\theta}_k||_2^2 + \sum_{i=1}^{N_k} \log p(w_i|\hat{\theta}_k; R, b) + \sum_{k=1}^{|D|} \frac{1}{|S_k|} \sum_{i=1}^{N_k} \log p(s_k|w_i; R, \psi, b_c). \tag{11}$$

$|S_k|$ denotes the number of documents in the dataset with the same rounded value of s_k (i.e. $s_k < 0.5$ and $s_k \geq 0.5$). We introduce the weighting $\frac{1}{|S_k|}$ to combat the well-known imbalance in ratings present in review collections. This weighting prevents the overall distribution of document ratings from affecting the estimate of document ratings in which a particular word occurs. The hyper-parameters of the model are the regularization weights (λ and ν), and the word vector dimensionality β.

Maximizing the objective function with respect to R, b, ψ, and b_c is a non-convex problem. We use alternating maximization, which first optimizes the

word representations (R, b, ψ, and b_c) while leaving the MAP estimates ($\hat{\theta}$) fixed. Then we find the new MAP estimate for each document while leaving the word representations fixed, and continue this process until convergence. The optimization algorithm quickly finds a global solution for each $\hat{\theta}_k$ because we have a low-dimensional, convex problem in each $\hat{\theta}_k$. Because the MAP estimation problems for different documents are independent, we can solve them on separate machines in parallel. This facilitates scaling the model to document collections with hundreds of thousands of documents.

4 Experiments

We evaluate our model with document-level and sentence-level categorization tasks in the domain of online movie reviews. For document categorization, we compare our method to previously published results on a standard dataset, and introduce a new dataset for the task. In both tasks we compare our model's word representations with several bag of words weighting methods, and alternative approaches to word vector induction.

4.1 Word Representation Learning

We induce word representations with our model using 25,000 movie reviews from IMDB. Because some movies receive substantially more reviews than others, we limited ourselves to including at most 30 reviews from any movie in the collection. We build a fixed dictionary of the 5,000 most frequent tokens, but ignore the 50 most frequent terms from the original full vocabulary. Traditional stop word removal was not used because certain stop words (e.g. negating words) are indicative of sentiment. Stemming was not applied because the model learns similar representations for words of the same stem when the data suggests it. Additionally, because certain non-word tokens (e.g. "!" and ":-)") are indicative of sentiment, we allow them in our vocabulary. Ratings on IMDB are given as star values ($\in \{1, 2, ..., 10\}$), which we linearly map to $[0, 1]$ to use as document labels when training our model.

The semantic component of our model does not require document labels. We train a variant of our model which uses 50,000 unlabeled reviews in addition to the labeled set of 25,000 reviews. The unlabeled set of reviews contains neutral reviews as well as those which are polarized as found in the labeled set. Training the model with additional unlabeled data captures a common scenario where the amount of labeled data is small relative to the amount of unlabeled data available. For all word vector models, we use 50-dimensional vectors.

As a qualitative assessment of word representations, we visualize the words most similar to a query word using vector similarity of the learned representations. Given a query word w and another word w' we obtain their vector representations ϕ_w and $\phi_{w'}$, and evaluate their cosine similarity as $S(\phi_w, \phi_{w'}) = \frac{\phi_w^T \phi_{w'}}{||\phi_w|| \cdot ||\phi_{w'}||}$. By assessing the similarity of w with all other words w', we can find the words deemed most similar by the model.

Table 1 shows the most similar words to given query words using our model's word representations as well as those of LSA. All of these vectors capture broad semantic similarities. However, both versions of our model seem to do better than LSA in avoiding accidental distributional similarities (e.g., *screwball* and *grant* as similar to *romantic*) A comparison of the two versions of our model also begins to highlight the importance of adding sentiment information. In general, words indicative of sentiment tend to have high similarity with words of the same sentiment polarity, so even the purely unsupervised model's results look promising. However, they also show more genre and content effects. For example, the sentiment enriched vectors for *ghastly* are truly semantic alternatives to that word, whereas the vectors without sentiment also contain some content words that tend to have *ghastly* predicated of them. Of course, this is only an impressionistic analysis of a few cases, but it is helpful in understanding why the sentiment-enriched model proves superior at the sentiment classification results we report next.

4.2 Other Word Representations

For comparison, we implemented several alternative vector space models that are conceptually similar to our own, as discussed in section 2:

Latent Semantic Analysis (LSA; Deerwester et al., 1990) We apply truncated SVD to a tf.idf weighted, cosine normalized count matrix, which is a standard weighting and smoothing scheme for

	Our model Sentiment + Semantic	Our model Semantic only	LSA
melancholy	bittersweet	thoughtful	poetic
	heartbreaking	warmth	lyrical
	happiness	layer	poetry
	tenderness	gentle	profound
	compassionate	loneliness	vivid
ghastly	embarrassingly	predators	hideous
	trite	hideous	inept
	laughably	tube	severely
	atrocious	baffled	grotesque
	appalling	smack	unsuspecting
lackluster	lame	passable	uninspired
	laughable	unconvincing	flat
	unimaginative	amateurish	bland
	uninspired	clichéd	forgettable
	awful	insipid	mediocre
romantic	romance	romance	romance
	love	charming	screwball
	sweet	delightful	grant
	beautiful	sweet	comedies
	relationship	chemistry	comedy

Table 1: Similarity of learned word vectors. Each target word is given with its five most similar words using cosine similarity of the vectors determined by each model. The full version of our model (left) captures both lexical similarity as well as similarity of sentiment strength and orientation. Our unsupervised semantic component (center) and LSA (right) capture semantic relations.

VSM induction (Turney and Pantel, 2010).

Latent Dirichlet Allocation (LDA; Blei et al., 2003) We use the method described in section 2 for inducing word representations from the topic matrix. To train the 50-topic LDA model we use code released by Blei et al. (2003). We use the same 5,000 term vocabulary for LDA as is used for training word vector models. We leave the LDA hyperparameters at their default values, though some work suggests optimizing over priors for LDA is important (Wallach et al., 2009).

Weighting Variants We evaluate both binary (b) term frequency weighting with smoothed delta idf (Δt') and no idf (n) because these variants worked well in previous experiments in sentiment (Martineau and Finin, 2009; Pang et al., 2002). In all cases, we use cosine normalization (c). Paltoglou and Thelwall (2010) perform an extensive analysis of such weighting variants for sentiment tasks.

4.3 Document Polarity Classification

Our first evaluation task is document-level sentiment polarity classification. A classifier must predict whether a given review is positive or negative given the review text.

Given a document's bag of words vector v, we obtain features from our model using a matrix-vector product Rv, where v can have arbitrary tf.idf weighting. We do not cosine normalize v, instead applying cosine normalization to the final feature vector Rv. This procedure is also used to obtain features from the LDA and LSA word vectors. In preliminary experiments, we found 'bnn' weighting to work best for v when generating document features via the product Rv. In all experiments, we use this weighting to get multi-word representations

Features	PL04	Our Dataset	Subjectivity
Bag of Words (bnc)	85.45	87.80	87.77
Bag of Words (bΔt'c)	85.80	88.23	85.65
LDA	66.70	67.42	66.65
LSA	84.55	83.96	82.82
Our Semantic Only	87.10	87.30	86.65
Our Full	84.65	87.44	86.19
Our Full, Additional Unlabeled	87.05	87.99	87.22
Our Semantic + Bag of Words (bnc)	88.30	88.28	88.58
Our Full + Bag of Words (bnc)	87.85	88.33	88.45
Our Full, Add'l Unlabeled + Bag of Words (bnc)	88.90	88.89	88.13
Bag of Words SVM (Pang and Lee, 2004)	87.15	N/A	90.00
Contextual Valence Shifters (Kennedy and Inkpen, 2006)	86.20	N/A	N/A
tf.Δidf Weighting (Martineau and Finin, 2009)	88.10	N/A	N/A
Appraisal Taxonomy (Whitelaw et al., 2005)	90.20	N/A	N/A

Table 2: Classification accuracy on three tasks. From left to right the datasets are: A collection of 2,000 movie reviews often used as a benchmark of sentiment classification (Pang and Lee, 2004), 50,000 reviews we gathered from IMDB, and the sentence subjectivity dataset also released by (Pang and Lee, 2004). All tasks are balanced two-class problems.

from word vectors.

4.3.1 Pang and Lee Movie Review Dataset

The polarity dataset version 2.0 introduced by Pang and Lee (2004) [1] consists of 2,000 movie reviews, where each is associated with a binary sentiment polarity label. We report 10-fold cross validation results using the authors' published folds to make our results comparable with others in the literature. We use a linear support vector machine (SVM) classifier trained with LIBLINEAR (Fan et al., 2008), and set the SVM regularization parameter to the same value used by Pang and Lee (2004).

Table 2 shows the classification performance of our method, other VSMs we implemented, and previously reported results from the literature. Bag of words vectors are denoted by their weighting notation. Features from word vector learner are denoted by the learner name. As a control, we trained versions of our model with only the unsupervised semantic component, and the full model (semantic and sentiment). We also include results for a version of our full model trained with 50,000 additional unlabeled examples. Finally, to test whether our models' representations complement a standard bag of words, we evaluate performance of the two feature representations concatenated.

Our method's features clearly outperform those of other VSMs, and perform best when combined with the original bag of words representation. The variant of our model trained with additional unlabeled data performed best, suggesting the model can effectively utilize large amounts of unlabeled data along with labeled examples. Our method performs competitively with previously reported results in spite of our restriction to a vocabulary of only 5,000 words.

We extracted the movie title associated with each review and found that 1,299 of the 2,000 reviews in the dataset have at least one other review of the same movie in the dataset. Of 406 movies with multiple reviews, 249 have the same polarity label for all of their reviews. Overall, these facts suggest that, relative to the size of the dataset, there are highly correlated examples with correlated labels. This is a natural and expected property of this kind of document collection, but it can have a substantial impact on performance in datasets of this scale. In the random folds distributed by the authors, approximately 50% of reviews in each validation fold's test set have a review of the same movie with the same label in the training set. Because the dataset is small, a learner may perform well by memorizing the association between label and words unique to a particular movie (e.g., character names or plot terms).

We introduce a substantially larger dataset, which

[1] http://www.cs.cornell.edu/people/pabo/movie-review-data

uses disjoint sets of movies for training and testing. These steps minimize the ability of a learner to rely on idiosyncratic word–class associations, thereby focusing attention on genuine sentiment features.

4.3.2 IMDB Review Dataset

We constructed a collection of 50,000 reviews from IMDB, allowing no more than 30 reviews per movie. The constructed dataset contains an even number of positive and negative reviews, so randomly guessing yields 50% accuracy. Following previous work on polarity classification, we consider only highly polarized reviews. A negative review has a score ≤ 4 out of 10, and a positive review has a score ≥ 7 out of 10. Neutral reviews are not included in the dataset. In the interest of providing a benchmark for future work in this area, we release this dataset to the public.[2]

We evenly divided the dataset into training and test sets. The training set is the same 25,000 labeled reviews used to induce word vectors with our model. We evaluate classifier performance after cross-validating classifier parameters on the training set, again using a linear SVM in all cases. Table 2 shows classification performance on our subset of IMDB reviews. Our model showed superior performance to other approaches, and performed best when concatenated with bag of words representation. Again the variant of our model which utilized extra unlabeled data during training performed best.

Differences in accuracy are small, but, because our test set contains 25,000 examples, the variance of the performance estimate is quite low. For example, an accuracy increase of 0.1% corresponds to correctly classifying an additional 25 reviews.

4.4 Subjectivity Detection

As a second evaluation task, we performed sentence-level subjectivity classification. In this task, a classifier is trained to decide whether a given sentence is subjective, expressing the writer's opinions, or objective, expressing purely facts. We used the dataset of Pang and Lee (2004), which contains subjective sentences from movie review summaries and objective sentences from movie plot summaries. This task is substantially different from the review classification task because it uses sentences as opposed to entire documents and the target concept is subjectivity instead of opinion polarity. We randomly split the 10,000 examples into 10 folds and report 10-fold cross validation accuracy using the SVM training protocol of Pang and Lee (2004).

Table 2 shows classification accuracies from the sentence subjectivity experiment. Our model again provided superior features when compared against other VSMs. Improvement over the bag-of-words baseline is obtained by concatenating the two feature vectors.

5 Discussion

We presented a vector space model that learns word representations captuing semantic and sentiment information. The model's probabilistic foundation gives a theoretically justified technique for word vector induction as an alternative to the overwhelming number of matrix factorization-based techniques commonly used. Our model is parametrized as a log-bilinear model following recent success in using similar techniques for language models (Bengio et al., 2003; Collobert and Weston, 2008; Mnih and Hinton, 2007), and it is related to probabilistic latent topic models (Blei et al., 2003; Steyvers and Griffiths, 2006). We parametrize the topical component of our model in a manner that aims to capture word representations instead of latent topics. In our experiments, our method performed better than LDA, which models latent topics directly.

We extended the unsupervised model to incorporate sentiment information and showed how this extended model can leverage the abundance of sentiment-labeled texts available online to yield word representations that capture both sentiment and semantic relations. We demonstrated the utility of such representations on two tasks of sentiment classification, using existing datasets as well as a larger one that we release for future research. These tasks involve relatively simple sentiment information, but the model is highly flexible in this regard; it can be used to characterize a wide variety of annotations, and thus is broadly applicable in the growing areas of sentiment analysis and retrieval.

[2]Dataset and further details are available online at: `http://www.andrew-maas.net/data/sentiment`

Acknowledgments

This work is supported by the DARPA Deep Learning program under contract number FA8650-10-C-7020, an NSF Graduate Fellowship awarded to AM, and ONR grant No. N00014-10-1-0109 to CP.

References

C. O. Alm, D. Roth, and R. Sproat. 2005. Emotions from text: machine learning for text-based emotion prediction. In *Proceedings of HLT/EMNLP*, pages 579–586.

A. Andreevskaia and S. Bergler. 2006. Mining WordNet for fuzzy sentiment: sentiment tag extraction from WordNet glosses. In *Proceedings of the European ACL*, pages 209–216.

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. 2003. a neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, August.

D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, May.

J. Boyd-Graber and P. Resnik. 2010. Holistic sentiment analysis across languages: multilingual supervised latent Dirichlet allocation. In *Proceedings of EMNLP*, pages 45–55.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing. In *Proceedings of the ICML*, pages 160–167.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, September.

R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. 2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, August.

J. R. Finkel and C. D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of NAACL*, pages 326–334.

A. B. Goldberg and J. Zhu. 2006. Seeing stars when there aren't many stars: graph-based semi-supervised learning for sentiment categorization. In *TextGraphs: HLT/NAACL Workshop on Graph-based Algorithms for Natural Language Processing*, pages 45–52.

T. Jay. 2000. *Why We Curse: A Neuro-Psycho-Social Theory of Speech*. John Benjamins, Philadelphia/Amsterdam.

D. Kaplan. 1999. What is meaning? Explorations in the theory of *Meaning as Use*. Brief version — draft 1. Ms., UCLA.

A. Kennedy and D. Inkpen. 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22:110–125, May.

F. Li, M. Huang, and X. Zhu. 2010. Sentiment analysis with global topics and local dependency. In *Proceedings of AAAI*, pages 1371–1376.

C. Lin and Y. He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceeding of the 18th ACM Conference on Information and Knowledge Management*, pages 375–384.

J. Martineau and T. Finin. 2009. Delta tfidf: an improved feature space for sentiment analysis. In *Proceedings of the 3rd AAAI International Conference on Weblogs and Social Media*, pages 258–261.

A. Mnih and G. E. Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the ICML*, pages 641–648.

G. Paltoglou and M. Thelwall. 2010. A study of information retrieval weighting schemes for sentiment analysis. In *Proceedings of the ACL*, pages 1386–1395.

B. Pang and L. Lee. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, pages 271–278.

B. Pang and L. Lee. 2005. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*, pages 115–124.

B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86.

C. Potts. 2007. The expressive dimension. *Theoretical Linguistics*, 33:165–197.

B. Snyder and R. Barzilay. 2007. Multiple aspect ranking using the good grief algorithm. In *Proceedings of NAACL*, pages 300–307.

M. Steyvers and T. L. Griffiths. 2006. Probabilistic topic models. In T. Landauer, D McNamara, S. Dennis, and W. Kintsch, editors, *Latent Semantic Analysis: A Road to Meaning*.

J. Turian, L. Ratinov, and Y. Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the ACL*, page 384394.

P. D. Turney and P. Pantel. 2010. From frequency to meaning: vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

H. Wallach, D. Mimno, and A. McCallum. 2009. Rethinking LDA: why priors matter. In *Proceedings of NIPS*, pages 1973–1981.

C. Whitelaw, N. Garg, and S. Argamon. 2005. Using appraisal groups for sentiment analysis. In *Proceedings of CIKM*, pages 625–631.

T. Wilson, J. Wiebe, and R. Hwa. 2004. Just how mad are you? Finding strong and weak opinion clauses. In *Proceedings of AAAI*, pages 761–769.

Target-dependent Twitter Sentiment Classification

Long Jiang[1] **Mo Yu**[2] **Ming Zhou**[1] **Xiaohua Liu**[1] **Tiejun Zhao**[2]

1 Microsoft Research Asia
Beijing, China
{longj,mingzhou,xiaoliu}@microsoft.com

2 School of Computer Science & Technology
Harbin Institute of Technology
Harbin, China
{yumo,tjzhao}@mtlab.hit.edu.cn

Abstract

Sentiment analysis on Twitter data has attracted much attention recently. In this paper, we focus on target-dependent Twitter sentiment classification; namely, given a query, we classify the sentiments of the tweets as positive, negative or neutral according to whether they contain positive, negative or neutral sentiments about that query. Here the query serves as the target of the sentiments. The state-of-the-art approaches for solving this problem always adopt the target-independent strategy, which may assign irrelevant sentiments to the given target. Moreover, the state-of-the-art approaches only take the tweet to be classified into consideration when classifying the sentiment; they ignore its context (i.e., related tweets). However, because tweets are usually short and more ambiguous, sometimes it is not enough to consider only the current tweet for sentiment classification. In this paper, we propose to improve target-dependent Twitter sentiment classification by 1) incorporating target-dependent features; and 2) taking related tweets into consideration. According to the experimental results, our approach greatly improves the performance of target-dependent sentiment classification.

1 Introduction

Twitter, as a micro-blogging system, allows users to publish tweets of up to 140 characters in length to tell others what they are doing, what they are thinking, or what is happening around them. Over the past few years, Twitter has become very popular. According to the latest Twitter entry in Wikipedia, the number of Twitter users has climbed to 190 million and the number of tweets published on Twitter every day is over 65 million[1].

As a result of the rapidly increasing number of tweets, mining people's sentiments expressed in tweets has attracted more and more attention. In fact, there are already many web sites built on the Internet providing a Twitter sentiment search service, such as Tweetfeel[2], Twendz[3], and Twitter Sentiment[4]. In those web sites, the user can input a sentiment target as a query, and search for tweets containing positive or negative sentiments towards the target. The problem needing to be addressed can be formally named as Target-dependent Sentiment Classification of Tweets; namely, given a query, classifying the sentiments of the tweets as positive, negative or neutral according to whether they contain positive, negative or neutral sentiments about that query. Here the query serves as the target of the sentiments.

The state-of-the-art approaches for solving this problem, such as (Go et al., 2009[5]; Barbosa and Feng, 2010), basically follow (Pang et al., 2002), who utilize machine learning based classifiers for the sentiment classification of texts. However, their classifiers actually work in a target-independent way: all the features used in the classifiers are independent of the target, so the sentiment is decided no matter what the target is. Since (Pang et al., 2002) (or later research on sentiment classification

[1] http://en.wikipedia.org/wiki/Twitter
[2] http://www.tweetfeel.com/
[3] http://twendz.waggeneredstrom.com/
[4] http://twittersentiment.appspot.com/
[5] The algorithm used in Twitter Sentiment

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 151–160,
Portland, Oregon, June 19-24, 2011.

of product reviews) aim to classify the polarities of movie (or product) reviews and each movie (or product) review is assumed to express sentiments only about the target movie (or product), it is reasonable for them to adopt the target-independent approach. However, for target-dependent sentiment classification of tweets, it is not suitable to exactly adopt that approach. Because people may mention multiple targets in one tweet or comment on a target in a tweet while saying many other unrelated things in the same tweet, target-independent approaches are likely to yield unsatisfactory results:

1. Tweets that do not express any sentiments to the given target but express sentiments to other things will be considered as being opinionated about the target. For example, the following tweet expresses no sentiment to *Bill Gates* but is very likely to be classified as positive about *Bill Gates* by target-independent approaches.

   "People everywhere love Windows & vista. Bill Gates"

2. The polarities of some tweets towards the given target are misclassified because of the interference from sentiments towards other targets in the tweets. For example, the following tweet expresses a positive sentiment to *Windows 7* and a negative sentiment to *Vista*. However, with target-independent sentiment classification, both of the targets would get positive polarity.

   "Windows 7 is much better than Vista!"

In fact, it is easy to find many such cases by looking at the output of Twitter Sentiment or other Twitter sentiment analysis web sites. Based on our manual evaluation of Twitter Sentiment output, about 40% of errors are because of this (see Section 6.1 for more details).

In addition, tweets are usually shorter and more ambiguous than other sentiment data commonly used for sentiment analysis, such as reviews and blogs. Consequently, it is more difficult to classify the sentiment of a tweet only based on its content. For instance, for the following tweet, which contains only three words, it is difficult for any existing approaches to classify its sentiment correctly.

"First game: Lakers!"

However, relations between individual tweets are more common than those in other sentiment data. We can easily find many related tweets of a given tweet, such as the tweets published by the same person, the tweets replying to or replied by the given tweet, and retweets of the given tweet. These related tweets provide rich information about what the given tweet expresses and should definitely be taken into consideration for classifying the sentiment of the given tweet.

In this paper, we propose to improve target-dependent sentiment classification of tweets by using both target-dependent and context-aware approaches. Specifically, the target-dependent approach refers to incorporating syntactic features generated using words syntactically connected with the given target in the tweet to decide whether or not the sentiment is about the given target. For instance, in the second example, using syntactic parsing, we know that "*Windows 7*" is connected to "*better*" by a copula, while "*Vista*" is connected to "*better*" by a preposition. By learning from training data, we can probably predict that "*Windows 7*" should get a positive sentiment and "*Vista*" should get a negative sentiment.

In addition, we also propose to incorporate the contexts of tweets into classification, which we call a context-aware approach. By considering the sentiment labels of the related tweets, we can further boost the performance of the sentiment classification, especially for very short and ambiguous tweets. For example, in the third example we mentioned above, if we find that the previous and following tweets published by the same person are both positive about the *Lakers*, we can confidently classify this tweet as positive.

The remainder of this paper is structured as follows. In Section 2, we briefly summarize related work. Section 3 gives an overview of our approach. We explain the target-dependent and context-aware approaches in detail in Sections 4 and 5 respectively. Experimental results are reported in Section 6 and Section 7 concludes our work.

2 Related Work

In recent years, sentiment analysis (SA) has become a hot topic in the NLP research community. A lot of papers have been published on this topic.

2.1 Target-independent SA

Specifically, Turney (2002) proposes an unsupervised method for classifying product or movie reviews as positive or negative. In this method, sentimental phrases are first selected from the reviews according to predefined part-of-speech patterns. Then the semantic orientation score of each phrase is calculated according to the mutual information values between the phrase and two predefined seed words. Finally, a review is classified based on the average semantic orientation of the sentimental phrases in the review.

In contrast, (Pang et al., 2002) treat the sentiment classification of movie reviews simply as a special case of a topic-based text categorization problem and investigate three classification algorithms: Naive Bayes, Maximum Entropy, and Support Vector Machines. According to the experimental results, machine learning based classifiers outperform the unsupervised approach, where the best performance is achieved by the SVM classifier with unigram presences as features.

2.2 Target-dependent SA

Besides the above mentioned work for target-independent sentiment classification, there are also several approaches proposed for target-dependent classification, such as (Nasukawa and Yi, 2003; Hu and Liu, 2004; Ding and Liu, 2007). (Nasukawa and Yi, 2003) adopt a rule based approach, where rules are created by humans for adjectives, verbs, nouns, and so on. Given a sentiment target and its context, part-of-speech tagging and dependency parsing are first performed on the context. Then predefined rules are matched in the context to determine the sentiment about the target. In (Hu and Liu, 2004), opinions are extracted from product reviews, where the features of the product are considered opinion targets. The sentiment about each target in each sentence of the review is determined based on the dominant orientation of the opinion words appearing in the sentence.

As mentioned in Section 1, target-dependent sentiment classification of review sentences is quite different from that of tweets. In reviews, if any sentiment is expressed in a sentence containing a feature, it is very likely that the sentiment is about the feature. However, the assumption does not hold in tweets.

2.3 SA of Tweets

As Twitter becomes more popular, sentiment analysis on Twitter data becomes more attractive. (Go et al., 2009; Parikh and Movassate, 2009; Barbosa and Feng, 2010; Davidiv et al., 2010) all follow the machine learning based approach for sentiment classification of tweets. Specifically, (Davidiv et al., 2010) propose to classify tweets into multiple sentiment types using hashtags and smileys as labels. In their approach, a supervised KNN-like classifier is used. In contrast, (Barbosa and Feng, 2010) propose a two-step approach to classify the sentiments of tweets using SVM classifiers with abstract features. The training data is collected from the outputs of three existing Twitter sentiment classification web sites. As mentioned above, these approaches work in a target-independent way, and so need to be adapted for target-dependent sentiment classification.

3 Approach Overview

The problem we address in this paper is target-dependent sentiment classification of tweets. So the input of our task is a collection of tweets containing the target and the output is labels assigned to each of the tweets. Inspired by (Barbosa and Feng, 2010; Pang and Lee, 2004), we design a three-step approach in this paper:

1. Subjectivity classification as the first step to decide if the tweet is subjective or neutral about the target;
2. Polarity classification as the second step to decide if the tweet is positive or negative about the target if it is classified as subjective in Step 1;
3. Graph-based optimization as the third step to further boost the performance by taking the related tweets into consideration.

In each of the first two steps, a binary SVM classifier is built to perform the classification. To train the classifiers, we use SVM-Light[6] with a linear kernel; the default setting is adopted in all experiments.

[6] http://svmlight.joachims.org/

3.1 Preprocessing

In our approach, rich feature representations are used to distinguish between sentiments expressed towards different targets. In order to generate such features, much NLP work has to be done beforehand, such as tweet normalization, POS tagging, word stemming, and syntactic parsing.

In our experiments, POS tagging is performed by the OpenNLP POS tagger[7]. Word stemming is performed by using a word stem mapping table consisting of about 20,000 entries. We also built a simple rule-based model for tweet normalization which can correct simple spelling errors and variations into normal form, such as "gooood" to "good" and "luve" to "love". For syntactic parsing we use a Maximum Spanning Tree dependency parser (McDonald et al., 2005).

3.2 Target-independent Features

Previous work (Barbosa and Feng, 2010; Davidiv et al., 2010) has discovered many effective features for sentiment analysis of tweets, such as emoticons, punctuation, prior subjectivity and polarity of a word. In our classifiers, most of these features are also used. Since these features are all generated without considering the target, we call them target-independent features. In both the subjectivity classifier and polarity classifier, the same target-independent feature set is used. Specifically, we use two kinds of target-independent features:

1. Content features, including words, punctuation, emoticons, and hashtags (hashtags are provided by the author to indicate the topic of the tweet).
2. Sentiment lexicon features, indicating how many positive or negative words are included in the tweet according to a predefined lexicon. In our experiments, we use the lexicon downloaded from General Inquirer[8].

4 Target-dependent Sentiment Classification

Besides target-independent features, we also incorporate target-dependent features in both the subjectivity classifier and polarity classifier. We will explain them in detail below.

4.1 Extended Targets

It is quite common that people express their sentiments about a target by commenting not on the target itself but on some related things of the target. For example, one may express a sentiment about a company by commenting on its products or technologies. To express a sentiment about a product, one may choose to comment on the features or functionalities of the product. It is assumed that readers or audiences can clearly infer the sentiment about the target based on those sentiments about the related things. As shown in the tweet below, the author expresses a positive sentiment about "Microsoft" by expressing a positive sentiment directly about "Microsoft technologies".

"I am passionate about Microsoft technologies especially Silverlight."

In this paper, we define those aforementioned related things as Extended Targets. Tweets expressing positive or negative sentiments towards the extended targets are also regarded as positive or negative about the target. Therefore, for target-dependent sentiment classification of tweets, the first thing is identifying all extended targets in the input tweet collection.

In this paper, we first regard all noun phrases, including the target, as extended targets for simplicity. However, it would be interesting to know under what circumstances the sentiment towards the target is truly consistent with that towards its extended targets. For example, a sentiment about someone's behavior usually means a sentiment about the person, while a sentiment about someone's colleague usually has nothing to do with the person. This could be a future work direction for target-dependent sentiment classification.

In addition to the noun phrases including the target, we further expand the extended target set with the following three methods:

1. Adding mentions co-referring to the target as new extended targets. It is common that people use definite or demonstrative noun phrases or pronouns referring to the target in a tweet and express sentiments directly on them. For instance, in "Oh, Jon Stewart. How I love you so.", the author expresses

[7] http://opennlp.sourceforge.net/projects.html
[8] http://www.wjh.harvard.edu/~inquirer/

a positive sentiment to "you" which actually refers to "Jon Stewart". By using a simple co-reference resolution tool adapted from (Soon et al., 2001), we add all the mentions referring to the target into the extended target set.

2. Identifying the top K nouns and noun phrases which have the strongest association with the target. Here, we use Pointwise Mutual Information (PMI) to measure the association.

$$PMI(w,t) = \log \frac{p(w,t)}{p(w)p(t)}$$

Where *p(w,t)*, *p(w)*, and *p(t)* are probabilities of *w* and *t* co-occurring, *w* appearing, and *t* appearing in a tweet respectively. In the experiments, we estimate them on a tweet corpus containing 20 million tweets. We set $K = 20$ in the experiments based on empirical observations.

3. Extracting head nouns of all extended targets, whose PMI values with the target are above some predefined threshold, as new extended targets. For instance, suppose we have found *"Microsoft Technologies"* as the extended target, we will further add *"technologies"* into the extended target set if the PMI value for *"technologies"* and *"Microsoft"* is above the threshold. Similarly, we can find *"price"* as the extended targets for *"iPhone"* from *"the price of iPhone"* and *"LoveGame"* for *"Lady Gaga"* from *"LoveGame by Lady Gaga"*.

4.2 Target-dependent Features

Target-dependent sentiment classification needs to distinguish the expressions describing the target from other expressions. In this paper, we rely on the syntactic parse tree to satisfy this need. Specifically, for any word stem w_i in a tweet which has one of the following relations with the given target *T* or any from the extended target set, we generate corresponding target-dependent features with the following rules:

- w_i is a transitive verb and *T* (or any of the extended target) is its object; we generate a feature w_i *_arg2*. "arg" is short for "argument". For example, for the target iPhone in *"I love iPhone"*, we generate *"love_arg2"* as a feature.
- w_i is a transitive verb and *T* (or any of the extended target) is its subject; we generate a feature w_i*_arg1* similar to Rule 1.
- w_i is a intransitive verb and *T* (or any of the extended target) is its subject; we generate a feature w_i*_it_arg1*.
- w_i is an adjective or noun and *T* (or any of the extended target) is its head; we generate a feature w_i*_arg1*.
- w_i is an adjective or noun and it (or its head) is connected by a copula with *T* (or any of the extended target); we generate a feature w_i*_cp_arg1*.
- w_i is an adjective or intransitive verb appearing alone as a sentence and *T* (or any of the extended target) appears in the previous sentence; we generate a feature w_i*_arg*. For example, in "*John did that. Great!*", "*Great*" appears alone as a sentence, so we generate "*great_arg*" for the target "*John*".
- w_i is an adverb, and the verb it modifies has *T* (or any of the extended target) as its subject; we generate a feature *arg1_v_*w_i. For example, for the target *iPhone* in the tweet "*iPhone works better with the CellBand",* we will generate the feature "*arg1_v_well*".

Moreover, if any word included in the generated target-dependent features is modified by a negation[9], then we will add a prefix "*neg-*" to it in the generated features. For example, for the target iPhone in the tweet "*iPhone does not work better with the CellBand*", we will generate the features "*arg1_v_neg-well*" and "*neg-work_it_arg1*".

To overcome the sparsity of target-dependent features mentioned above, we design a special binary feature indicating whether or not the tweet contains at least one of the above target-dependent features. Target-dependent features are binary features, each of which corresponds to the presence of the feature in the tweet. If the feature is present, the entry will be 1; otherwise it will be 0.

[9] Seven negations are used in the experiments: *not, no, never, n't, neither, seldom, hardly*.

5 Graph-based Sentiment Optimization

As we mentioned in Section 1, since tweets are usually shorter and more ambiguous, it would be useful to take their contexts into consideration when classifying the sentiments. In this paper, we regard the following three kinds of related tweets as context for a tweet.

1. Retweets. Retweeting in Twitter is essentially the forwarding of a previous message. People usually do not change the content of the original tweet when retweeting. So retweets usually have the same sentiment as the original tweets.
2. Tweets containing the target and published by the same person. Intuitively, the tweets published by the same person within a short timeframe should have a consistent sentiment about the same target.
3. Tweets replying to or replied by the tweet to be classified.

Based on these three kinds of relations, we can construct a graph using the input tweet collection of a given target. As illustrated in Figure 1, each circle in the graph indicates a tweet. The three kinds of edges indicate being published by the same person (solid line), retweeting (dash line), and replying relations (round dotted line) respectively.

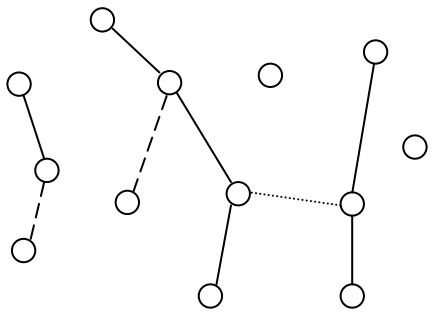


Figure 1. An example graph of tweets about a target

If we consider that the sentiment of a tweet only depends on its content and immediate neighbors, we can leverage a graph-based method for sentiment classification of tweets. Specifically, the probability of a tweet belonging to a specific sentiment class can be computed with the following formula:

$$p(c \mid \tau, G) = p(c \mid \tau) \sum_{N(d)} p(c \mid N(d)) p(N(d))$$

Where c is the sentiment label of a tweet which belongs to {positive, negative, neutral}, G is the tweet graph, $N(d)$ is a specific assignment of sentiment labels to all immediate neighbors of the tweet, and τ is the content of the tweet.

We can convert the output scores of a tweet by the subjectivity and polarity classifiers into probabilistic form and use them to approximate $p(c \mid \tau)$. Then a relaxation labeling algorithm described in (Angelova and Weikum, 2006) can be used on the graph to iteratively estimate $p(c \mid \tau, G)$ for all tweets. After the iteration ends, for any tweet in the graph, the sentiment label that has the maximum $p(c \mid \tau, G)$ is considered the final label.

6 Experiments

Because there is no annotated tweet corpus publicly available for evaluation of target-dependent Twitter sentiment classification, we have to create our own. Since people are most interested in sentiments towards celebrities, companies and products, we selected 5 popular queries of these kinds: {*Obama*, *Google*, *iPad*, *Lakers*, *Lady Gaga*}. For each of those queries, we downloaded 400 English tweets[10] containing the query using the Twitter API.

We manually classify each tweet as positive, negative or neutral towards the query with which it is downloaded. After removing duplicate tweets, we finally obtain 459 positive, 268 negative and 1,212 neutral tweets.

Among the tweets, 100 are labeled by two human annotators for inter-annotator study. The results show that for 86% of them, both annotators gave identical labels. Among the 14 tweets which the two annotators disagree on, only 1 case is a positive-negative disagreement (one annotator considers it positive while the other negative), and the other 13 are all neutral-subjective disagreement. This probably indicates that it is harder for humans to decide if a tweet is neutral or subjective than to decide if it is positive or negative.

[10] In this paper, we use sentiment classification of English tweets as a case study; however, our approach is applicable to other languages as well.

6.1 Error Analysis of Twitter Sentiment Output

We first analyze the output of Twitter Sentiment (TS) using the five test queries. For each query, we randomly select 20 tweets labeled as positive or negative by TS. We also manually classify each tweet as positive, negative or neutral about the corresponding query. Then, we analyze those tweets that get different labels from TS and humans. Finally we find two major types of error: 1) Tweets which are totally neutral (for any target) are classified as subjective by TS; 2) sentiments in some tweets are classified correctly but the sentiments are not truly about the query. The two types take up about 35% and 40% of the total errors, respectively.

The second type is actually what we want to resolve in this paper. After further checking those tweets of the second type, we found that most of them are actually neutral for the target, which means that the dominant error in Twitter Sentiment is classifying neutral tweets as subjective. Below are several examples of the second type where the bolded words are the targets.

*"No debate needed, heat can't beat **lakers** or celtics"* (negative by TS but positive by human)

*"why am i getting spams from weird people asking me if i want to chat with **lady gaga**"* (positive by TS but neutral by human)

*"Bringing iPhone and **iPad** apps into cars? http://www.speakwithme.com/ will be out soon and alpha is awesome in my car."* (positive by TS but neutral by human)

*"Here's a great article about Monte Veronese cheese. It's in Italian so just put the url into **Google** translate and enjoy http://ow.ly/3oQ77"* (positive by TS but neutral by human)

6.2 Evaluation of Subjectivity Classification

We conduct several experiments to evaluate subjectivity classifiers using different features. In the experiments, we consider the positive and negative tweets annotated by humans as subjective tweets (i.e., positive instances in the SVM classifiers), which amount to 727 tweets. Following (Pang et al., 2002), we balance the evaluation data set by randomly selecting 727 tweets from all neutral tweets annotated by humans and consider them as objective tweets (i.e., negative instances in the classifiers). We perform 10-fold cross-validations on the selected data. Following (Go et al., 2009; Pang et al., 2002), we use accuracy as a metric in our experiments. The results are listed below.

Features	Accuracy (%)
Content features	61.1
+ Sentiment lexicon features	63.8
+ Target-dependent features	**68.2**
Re-implementation of (Barbosa and Feng, 2010)	60.3

Table 1. Evaluation of subjectivity classifiers.

As shown in Table 1, the classifier using only the content features achieves an accuracy of 61.1%. Adding sentiment lexicon features improves the accuracy to 63.8%. Finally, the best performance (68.2%) is achieved by combining target-dependent features and other features (t-test: $p < 0.005$). This clearly shows that target-dependent features do help remove many sentiments not truly about the target. We also re-implemented the method proposed in (Barbosa and Feng, 2010) for comparison. From Table 1, we can see that all our systems perform better than (Barbosa and Feng, 2010) on our data set. One possible reason is that (Barbosa and Feng, 2010) use only abstract features while our systems use more lexical features.

To further evaluate the contribution of target extension, we compare the system using the exact target and all extended targets with that using only the exact target. We also eliminate the extended targets generated by each of the three target extension methods and reevaluate the performances.

Target	Accuracy (%)
Exact target	65.6
+ all extended targets	**68.2**
- co-references	68.0
- targets found by PMI	67.8
- head nouns	67.3

Table 2. Evaluation of target extension methods.

As shown in Table 2, without extended targets, the accuracy is 65.6%, which is still higher than those using only target-independent features. After adding all extended targets, the accuracy is improved significantly to 68.2% ($p < 0.005$), which suggests that target extension does help find indi-

rectly expressed sentiments about the target. In addition, all of the three methods contribute to the overall improvement, with the head noun method contributing most. However, the other two methods do not contribute significantly.

6.3 Evaluation of Polarity Classification

Similarly, we conduct several experiments on positive and negative tweets to compare the polarity classifiers with different features, where we use 268 negative and 268 randomly selected positive tweets. The results are listed below.

Features	Accuracy (%)
Content features	78.8
+ Sentiment lexicon features	84.2
+ Target-dependent features	**85.6**
Re-implementation of (Barbosa and Feng, 2010)	83.9

Table 3. Evaluation of polarity classifiers.

From Table 3, we can see that the classifier using only the content features achieves the worst accuracy (78.8%). Sentiment lexicon features are shown to be very helpful for improving the performance. Similarly, we re-implemented the method proposed by (Barbosa and Feng, 2010) in this experiment. The results show that our system using both content features and sentiment lexicon features performs slightly better than (Barbosa and Feng, 2010). The reason may be same as that we explained above.

Again, the classifier using all features achieves the best performance. Both the classifiers with all features and with the combination of content and sentiment lexicon features are significantly better than that with only the content features ($p < 0.01$). However, the classifier with all features does not significantly outperform that using the combination of content and sentiment lexicon features. We also note that the improvement by target-dependent features here is not as large as that in subjectivity classification. Both of these indicate that target-dependent features are more useful for improving subjectivity classification than for polarity classification. This is consistent with our observation in Subsection 6.2 that most errors caused by incorrect target association are made in subjectivity classification. We also note that all numbers in Table 3 are much bigger than those in Table 1, which suggests that subjectivity classification of tweets is more difficult than polarity classification.

Similarly, we evaluated the contribution of target extension for polarity classification. According to the results, adding all extended targets improves the accuracy by about 1 point. However, the contributions from the three individual methods are not statistically significant.

6.4 Evaluation of Graph-based Optimization

As seen in Figure 1, there are several tweets which are not connected with any other tweets. For these tweets, our graph-based optimization approach will have no effect. The following table shows the percentages of the tweets in our evaluation data set which have at least one related tweet according to various relation types.

Relation type	Percentage
Published by the same person[11]	41.6
Retweet	23.0
Reply	21.0
All	**66.2**

Table 4. Percentages of tweets having at least one related tweet according to various relation types.

According to Table 4, for 66.2% of the tweets concerning the test queries, we can find at least one related tweet. That means our context-aware approach is potentially useful for most of the tweets.

To evaluate the effectiveness of our context-aware approach, we compared the systems with and without considering the context.

System	Accuracy	F1-score (%)		
		pos	neu	neg
Target-dependent sentiment classifier	66.0	57.5	70.1	66.1
+Graph-based optimization	**68.3**	**63.5**	**71.0**	**68.5**

Table 5. Effectiveness of the context-aware approach.

As shown in Table 5, the overall accuracy of the target-dependent classifiers over three classes is 66.0%. The graph-based optimization improves the performance by over 2 points ($p < 0.005$), which clearly shows that the context information is very

[11] We limit the time frame from one week before to one week after the post time of the current tweet.

useful for classifying the sentiments of tweets. From the detailed improvement for each sentiment class, we find that the context-aware approach is especially helpful for positive and negative classes.

Relation type	Accuracy (%)
Published by the same person	**67.8**
Retweet	66.0
Reply	67.0

Table 6. Contribution comparison between relations.

We further compared the three types of relations for context-aware sentiment classification; the results are reported in Table 6. Clearly, being published by the same person is the most useful relation for sentiment classification, which is consistent with the percentage distribution of the tweets over relation types; using retweet only does not help. One possible reason for this is that the retweets and their original tweets are nearly the same, so it is very likely that they have already got the same labels in previous classifications.

7 Conclusions and Future Work

Twitter sentiment analysis has attracted much attention recently. In this paper, we address target-dependent sentiment classification of tweets. Different from previous work using target-independent classification, we propose to incorporate syntactic features to distinguish texts used for expressing sentiments towards different targets in a tweet. According to the experimental results, the classifiers incorporating target-dependent features significantly outperform the previous target-independent classifiers.

In addition, different from previous work using only information on the current tweet for sentiment classification, we propose to take the related tweets of the current tweet into consideration by utilizing graph-based optimization. According to the experimental results, the graph-based optimization significantly improves the performance.

As mentioned in Section 4.1, in future we would like to explore the relations between a target and any of its extended targets. We are also interested in exploring relations between Twitter accounts for classifying the sentiments of the tweets published by them.

Acknowledgments

We would like to thank Matt Callcut for refining the language of this paper, and thank Yuki Arase and the anonymous reviewers for many valuable comments and helpful suggestions. We would also thank Furu Wei and Xiaolong Wang for their help with some of the experiments and the preparation of the camera-ready version of the paper.

References

Ralitsa Angelova, Gerhard Weikum. 2006. Graph-based text classification: learn from your neighbors. SIGIR 2006: 485-492

Luciano Barbosa and Junlan Feng. 2010. Robust Sentiment Detection on Twitter from Biased and Noisy Data. Coling 2010.

Christopher Burges. 1998. A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery, 2(2):121-167.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan S. 2005. Identifying sources of opinions with conditional random fields and extraction patterns. In Proc. of the 2005 Human Language Technology Conf. and Conf. on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005). pp. 355-362

Dmitry Davidiv, Oren Tsur and Ari Rappoport. 2010. Enhanced Sentiment Learning Using Twitter Hashtags and Smileys. Coling 2010.

Xiaowen Ding and Bing Liu. 2007. The Utility of Linguistic Rules in Opinion Mining. SIGIR-2007 (poster paper), 23-27 July 2007, Amsterdam.

Alec Go, Richa Bhayani, Lei Huang. 2009. Twitter Sentiment Classification using Distant Supervision.

Vasileios Hatzivassiloglou and Kathleen.R. McKeown. 2002. Predicting the semantic orientation of adjectives. In Proceedings of the 35th ACL and the 8th Conference of the European Chapter of the ACL.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2004, full paper), Seattle, Washington, USA, Aug 22-25, 2004.

Thorsten Joachims. Making Large-scale Support Vector Machine Learning Practical. In B. SchÄolkopf, C. J. C. Burges, and A. J. Smola, editors, Advances in kernel methods: support vector learning, pages 169-184. MIT Press, Cambridge, MA, USA, 1999.

Soo-Min Kim and Eduard Hovy 2006. Extracting opinions, opinion holders, and topics expressed in online news media text, In Proc. of ACL Workshop on Sentiment and Subjectivity in Text, pp.1-8, Sydney, Australia.

Ryan McDonald, F. Pereira, K. Ribarov, and J. Haji˘c. 2005. Non-projective dependency parsing using spanning tree algorithms. In Proc. HLT/EMNLP.

Tetsuya Nasukawa, Jeonghee Yi. 2003. Sentiment analysis: capturing favorability using natural language processing. In Proceedings of K-CAP.

Bo Pang, Lillian Lee. 2004. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In Proceedings of ACL 2004.

Bo Pang, Lillian Lee, Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques.

Ravi Parikh and Matin Movassate. 2009. Sentiment Analysis of User-Generated Twitter Updates using Various Classification Techniques.

Wee. M. Soon, Hwee. T. Ng, and Danial. C. Y. Lim. 2001. A Machine Learning Approach to Coreference Resolution of Noun Phrases. Computational Linguistics, 27(4):521–544.

Peter D. Turney. 2002. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In proceedings of ACL 2002.

Janyce Wiebe. 2000. Learning subjective adjectives from corpora. In Proceedings of AAAI-2000.

Theresa Wilson, Janyce Wiebe, Paul Hoffmann. 2005. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In Proceedings of NAACL 2005.

A Comprehensive Dictionary of Multiword Expressions

Kosho Shudo[1], **Akira Kurahone**[2], **and Toshifumi Tanabe**[1]
[1]Fukuoka University, Nanakuma, Jonan-ku, Fukuoka, 814-0180, JAPAN
{shudo,tanabe}@fukuoka-u.ac.jp
[2]TechTran Ltd., Ikebukuro, Naka-ku, Yokohama, 231-0834, JAPAN
kurahone@opentech.co.jp

Abstract

It has been widely recognized that one of the most difficult and intriguing problems in natural language processing (NLP) is how to cope with idiosyncratic multiword expressions. This paper presents an overview of the comprehensive dictionary (JDMWE) of Japanese multiword expressions. The JDMWE is characterized by a large notational, syntactic, and semantic diversity of contained expressions as well as a detailed description of their syntactic functions, structures, and flexibilities. The dictionary contains about 104,000 expressions, potentially 750,000 expressions. This paper shows that the JDMWE's validity can be supported by comparing the dictionary with a large-scale Japanese N-gram frequency dataset, namely the LDC2009T08, generated by Google Inc. (Kudo et al. 2009).

1 Introduction

Linguistically idiosyncratic multiword expressions occur in authentic sentences with an unexpectedly high frequency. Since (Sag et al. 2002), we have become aware that a proper solution of idiosyncratic multiword expressions (MWEs) is one of the most difficult and intriguing problems in NLP. In principle, the nature of the idiosyncrasy of MWEs is twofold: one is idiomaticity, i.e., non-compositionality of meaning; the other is the strong probabilistic affinity between component words. Many attempts have been made to extract these expressions from corpora, mainly using automated methods that exploit statistical means. However, to our knowledge, no reliable, extensive solution has yet been made available, presumably because of the difficulty of extracting correctly without any human insight. Recognizing the crucial importance of such expressions, one of the authors of the current paper began in the 1970s to construct a Japanese electronic dictionary with comprehensive inclusion of idioms, idiom-like expressions, and probabilistically idiosyncratic expressions for general use. In this paper, we begin with an overview of the JDMWE (Japanese Dictionary of Multi-Word Expressions). It has approximately 104,000 dictionary entries and covers potentially at least 750,000 expressions. The most important features of the JDMWE are:

1. A large notational, syntactic, and semantic diversity of contained expressions
2. A detailed description of syntactic function and structure for each entry expression
3. An indication of the syntactic flexibility of entry expressions (i.e., possibility of internal modification of constituent words) of entry expressions.

In section 2, we outline the main features of the present study, first presenting a brief summary of significant previous work on this topic. In section 3, we propose and describe the criteria for selecting MWEs and introduce a number of classes of multiword expressions. In section 4, we outline the format and contents of the JDMWE, discussing the information on notational variants, syntactic functions, syntactic structures, and the syntactic flexibility of MWEs. In section 5, we describe and explain the contextual conditions stipulated in the JDMWE. In section 6, we illustrate some important statistical properties of the JDMWE by comparing the dictionary with a large-scale Japanese N-gram frequency dataset, the LDC2009T08, generated by Google Inc. (Kudo et al. 2009). The paper ends with concluding remarks in section 7.

2 Related Work

Gross (1986) analyzed French compound adverbs and compound verbs. According to his estimate, the lexical stock of such words in French would be respectively 3.3 and 1.7 times greater than that of single-word adverbs and single-word verbs. Jackendoff (1997) notes that an English speaker's lexicon would contain as many MWEs as single words. Sag et al. (2002) pointed out that 41% of the entries of WordNet 1.7 (Fellbaum 1999) are multiword; and Uchiyama et al. (2003) reported that 44% of Japanese verbs are VV-type compounds. These and other similar observations underscore the great need for a well-designed, extensive MWE lexicon for practical natural language processing.

In the past, attempts have been made to produce an MWE dictionary. Examples include the following: Gross (1986) reported on a dictionary of French verbal MWEs with description of 22 syntactic structures; Kuiper et al. (2003) constructed a database of 13,000 English idioms tagged with syntactic structures; Villavicencio (2004) attempted to compile lexicons of English idioms and verb-particle constructions (VPCs) by augmenting existing single-word dictionaries with specific tables; Baptista et al. (2004) reported on a dictionary of 3,500 Portuguese verbal MWEs with ten syntactic structures; Fellbaum et al. (2006) reported corpus-based studies in developing German verb phrase idiom resources; and recently, Laporte et al. (2008) have reported on a dictionary of 6,800 French adverbial MWEs annotated with 15 syntactic structures.

Our JDMWE approach differs from these studies in that it can treat more comprehensive types of MWEs. Our system can handle almost all types of MWEs except compositional compounds, named entities, acronyms, blends, politeness expressions, and functional expressions; in contrast, the types of MWEs that most of the other studies can deal with are limited to verb-object idioms, VPCs, verbal MWEs, support-verb constructions (SVCs) and so forth.

Many attempts have been made to extract MWEs automatically using statistical corpus-based methods. For example, Pantel et al. (2001) sought to extract Chinese compounds using mutual information and the log-likelihood measure. Fazly et al. (2006) attempted to extract English verb-object type idioms by recognizing their structural fixedness in terms of mutual information and relative entropy. Bannard (2007) tried to extract English syntactically fixed verb-noun combinations using pointwise mutual information, and so on.

In spite of these and many similar efforts, it is still difficult to adequately extract MWEs from corpora using a statistical approach, because regarding the types of multiword expressions, realistically speaking, the corpus-wide distribution can be far from exhaustive. Paradoxically, to compile an MWE lexicon we need a reliable standard MWE lexicon, as it is impossible to evaluate the automatic extraction by recall rate without such a reference. The conventional idiom dictionaries published for human readers have been occasionally used for the evaluation of automatic extraction methods in some past studies. However, no conventional Japanese dictionary of idioms would suffice for an MWE lexicon for the practical NLP because they lack entries related to the diverse MWE objects we frequently encounter in common textual materials, such as quasi-idioms, quasi-clichés, metaphoric fixed or partly fixed expressions. In addition, they provide no systematic information on the notational variants, syntactic functions, or syntactic structures of the entry expressions. The JDMWE is intended to circumvent these problems.

In past Japanese MWE studies, Shudo et al. (1980) compiled a lexicon of 3,500 functional multiword expressions and used the lexicon for a morphological analysis of Japanese. Koyama et al. (1998) made a seven-point increase in the precision rate of kana-to-kanji conversion for a commercial Japanese word processor by using a prototype of the JDMWE with 65,000 MWEs. Baldwin et al. (2003) discussed the treatment of Japanese MWEs in the framework of Sag et al. (2002). Shudo et al. (2004) pointed out the importance of the auxiliary-verbal MWEs and their non-propositional meanings (i.e., modality in a generalized sense). Hashimoto et al. (2009) studied a disambiguation method of semantically ambiguous idioms using 146 basic idioms.

3 MWEs Selected for the JDMWE

The human deliberate judgment is indispensable for the correct, extensive extraction of MWEs. In

view of this, we have manually extracted multiword expressions that have definite syntactic, semantic, or communicative functions and are linguistically idiosyncratic from a variety of publications, such as newspaper articles, journals, magazines, novels, and dictionaries. In principle, the idiosyncrasy of MWEs is twofold: first, the semantic non-compositionality (i.e., idiomaticity); second, the strong probabilistic affinity between component words. Here we have treated them differently.

The number of words included in a MWE ranges from two to eighteen. The length distribution is shown in Figure 1.

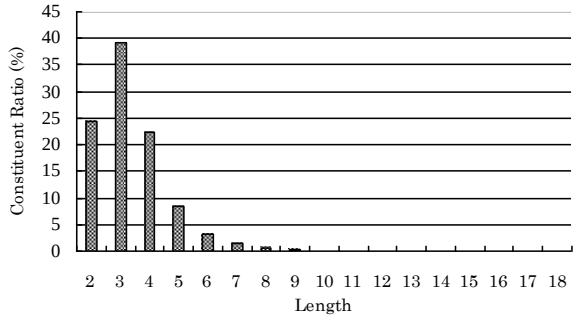


Figure 1: Length distribution of MWEs

type	example
Idiom: Semantically Non-Compositional Expression	赤−の−他人 *aka-no-tanin* (lit. red stranger) “complete stranger”
Morphologically or Syntactically Non-Compositional Expression, Cranberry-Type Expression	と−は−いえ *to-ha-ie* “however”
SVC: Support-Verb Construction	批判−を−加える *hihan-wo-kuwaeru* (lit. add criticism) “criticize”
Compound Noun; Compound Verb; Compound Adjective; Compound Adjective-Verb	打ち−拉がれる *uti-hisigareru* (lit. be hit and smashed) “become depressed”
Four-Character-Idiom	支離−滅裂 *siri-meturetu* “incoherence”
Metaphorical Expression	命−の−限り *inoti-no-kagiri* (lit. limit of life) “at the risk of life”
Quasi-Idiom	辞書−を−引く *jisho-wo-hiku* (lit. pull dictionary) “look up in a dictionary”

Table 1: Non-Compositional Expressions

3.1 Non-Compositional MWEs

In our approach, we use non-substitutability criterion to define a word string as an MWE, the logic being that an MWE expression is usually fixed in its form and the substitution of one of its constituent words would yield a meaningless expression or an expression with a meaning that is completely different from that of the original MWE expression. Formally, a word string $w_1w_2\cdots w_i\cdots w_n$ ($2\leq n\leq 18$) is an MWE if it has a definite syntactic, semantic, or communicative function of its own, and if $w_1w_2 \cdots w_i' \cdots w_n$ is either meaningless or has a meaning completely different from that of $w_1w_2\cdots w_i\cdots w_n$ for some i, where w_i' is any synonym or synonymous phrase of w_i. For example, 赤(w_1)-の-他人 *aka-no-tanin* (lit. “red stranger”) is selected because it has a definite nominal meaning of “complete stranger” and neither 真紅(w_1')-の-他人 *sinku-no-tanin* nor レッド(w_1')-の-他人 *reddo-no-tanin* means “complete stranger”. The evaluation of semantic relevance of MWEs was carried out by human judges entirely. It is just too difficult to judge the semantic relevance automatically and correctly. Table 1 shows a number of MWEs of this type.[1]

3.2 Probabilistically Idiosyncratic MWEs

An MWE must form a linguistic unit of its own. This and the following transition probability condition constitute another criterion that we adopt to define what an MWE is. Formally, a word string $w_1w_2\cdots w_i\cdots w_n$ ($2\leq n\leq 18$) is an MWE if it has a definite syntactic, semantic, or communicative function of its own, and if its forward or backward transition probability $\boldsymbol{p}_f(w_{i+1}|w_1\cdots w_i)$ or $\boldsymbol{p}_b(w_i|w_{i+1}\cdots w_n)$, respectively is judged to be in the relatively high range for some i. With this definition, for example, 手-を-拱く *te-wo-komaneku* “fold arms” is selected as an MWE because it is a well-formed verb phrase and $\boldsymbol{p}_b$(手|を-拱く) is judged empirically to be very high. No general probabilistic threshold value can be fixed a priori because the value is expression-dependent. Although the probabilistic judgment was performed, for each expression in turn, on the basis of the developer’s empirical language model, the resulting dataset is consistent with this criterion on

[1] These classes are not necessarily disjoint.

the whole as shown in section 6.1. Table 2 lists some MWEs of this type.[2]

type	example
Cliché, Stereotyped, Hackneyed, or Set Expression	風前−の−灯 *fuuzen-no-tomosibi* (lit. light in front of the wind) "candle flickering in the wind"
Proverb, Old-Saying	急が−ば−回れ *isoga-ba-maware* (lit. make a detour when in a hurry) "more haste, less speed"
Onomatopoeic or Mimetic Expression	ノロノロ−と−歩く *noronoro-to-aruku* (lit. slouchingly walk) "walk slowly"
Quasi-Cliché, Institutionalized Phrase	肩−の−荷−を−下ろす *kata-no-ni-wo-orosu* (lit. lower lord from the shoulder) "take a big load off one's mind"

Table 2: Probabilistically Idiosyncratic Expressions

With entries like these, an NLP system can use the JDMWE as a reliable reference while effectively disambiguating the structures in the syntactic analysis process.

Of the MWEs in the JDMWE, approximately 38% and 92% of them were judged to meet criterion 3.1 and criterion 3.2, respectively. These are illustrated in Figure 2.

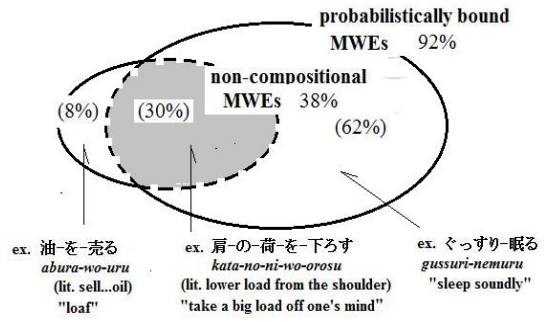


Figure 2: Approximate constituent ratio of non-compositional MWEs and probabilistically bound MWEs

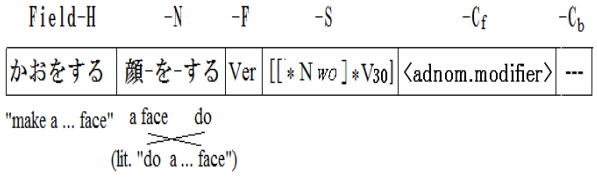


Figure 3: Example JDMWE entry

[2] These classes are not necessarily disjoint.

4 Contents of the JDMWE

The JDMWE has approximately 104,000 entries, one for each MWE, composed of six fields, namely, Field-H, -N, -F, -S, -C_f, and -C_b. The dictionary entry form of an MWE is stated in Field-H in the form of a non-segmented hira-kana (phonetic character) string. An example is given in Figure 3.

4.1 Notational Information (Field-N)

Japanese has three notational options: hira-kana, kata-kana, and kanji. The two kanas are phonological syllabaries. Kanji are originally Chinese idiographic characters. As we have many kanji characters that are both homophonic and synonymous, sentences can contain kanji replaceable by others. In addition, the inflectional suffix of some verbs can be absent in some contexts. The JDMWE has flexible conventions to cope with these characteristics. It uses brackets to indicate an optional word (or a series of interchangeable words marked off by the slash "/") in the Field-N description. Therefore, the entry whose Field-H (the first field) is きのいいやつ *ki-no-ii-yatu* (lit. "a guy who has a good spirit") "good-natured guy", can have (き/気)-の-(い/良/好/善)い-(やつ/奴/ヤツ) in its Field-N. The dash "-" is used as a word boundary indicator. This example can stand for twenty-four combinatorial variants, i.e., きのいいやつ,…, 気の良い奴,…, 気の善いヤツ.

If fully expanded with this information, the JDMWE's total number of MWEs can exceed 750,000.

4.2 Functional Information (Field-F)

Linguistic functions of MWEs can be simply classified by means of codes, as shown in Tables 3 and 4. Field-F is filled with one of those codes which corresponds to a root node label in the syntactic tree representation of a MWE.

code	function	size	example
Cdis	Discourse-Connective	1,000	言い−換えれ−ば *ii-kaere-ba* (lit. if (I) paraphrase) "in other words"
Adv	Adverbial	6,000	不思議−と *fusigi-to* "strangely enough"
Pren	Prenominal-Adjectival	13,700	確−たる *kaku-taru* "definite"

Nom	Nominal	12,000	灰汁−の−強さ *aku-no-tuyosa* (lit. strong taste of lye) “strong harshness”
Nd	Nominal/ Dynamic	4,700	一目−惚れ *hitome-bore* “love at first sight”
Nk	Nominal/State-describing	5,400	二−枚−舌 *ni-mai-jita* “being double-tongued”
Ver	Verbal	49,000	油−を−売る *abura-wo-uru* (lit. sell oil) “idle away”
Adj	Adjectival	4,600	眼−に−入れ−ても−痛く−ない *me-ni-ire-temo-itaku-nai* (lit. have no pain even if put into eyes) “an apple in ones eye”
K	Adjective-Verbal	3,500	経験−豊か *keiken-yutaka* “abundant in experience”
Ono	Onomatopoeic or Mimetic Expression	1,300	スラスラ−と *surasura-to* “smoothly”, “easily”, “fluently”

Table 3: Syntactic Functions and Examples

code	function	size	example
_P	Proverb, Old-Saying	2,300	百聞−は−一見−に−如か−ず *hyakubun-ha-ikken-ni-sika-zu* (lit. hearing about something a hundred times is not as good as seeing it once) “a picture is worth a thousand words”
_Self	Soliloquy, Monologue	200	困っ−た−なあ *komat-ta-naa* “Oh boy, we’re in trouble!”
_Call	Call, Yell	150	済み−ませ−ん−が *sumi-mase-n-ga* “Excuse me.”
_Grt	Greeting	200	いらっしゃい−ませ *irasshai-mase* “Welcome!”
_Res	Response	350	どう−いたし−まし−て *dou-itasi-masi-te* “You’re welcome.”

Table 4: Communicative Functions and Examples

4.3 Structural Information (Field-S)

4.3.1 Dependency Structure

The dependency structure of an MWE is given in Field-S by a phrase marker bracketing the modifier-head pairs, using POS symbols for conceptual words.[3] For example, an idiom 真っ赤-な - 嘘 *makka-na-uso* (lit. “crimson lie”) “downright lie” is given a marker [[K_{00} *na*] N]. This description represents the structure shown in Figure 4, where K_{00} and N are POS symbols denoting an adjective-verb stem and a noun, respectively.

[3] The intra-sentential dependency relation in Japanese is unilateral, i.e., the left modifier depends on the right head.

The JDMWE contains 49,000 verbal entries, making this the largest functional class in the JDMWE. For these verbal entries, more than 90 patterns are actually used as structural descriptors in Field-S. This fact can indicate the broadness of the structural spectrum of Japanese verbal MWEs. Some examples are shown in Table 5.

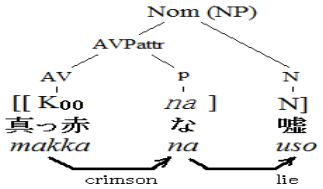


Figure 4: Example of dependency structure given in Field-S

example of structural pattern of verbal MWE	example of MWE
[[N *wo*] V_{30}]	異−を−唱える *i-wo-tonaeru* (lit. chant the difference) “raise an objection”
[[N *ga*] V_{30}]	撚り-が-戻る *yori-ga-modoru* (lit. the twist comes undone) “get reconciled”
[[N *ni*] V_{30}]	手−に−入れる *te-ni-ireru* (lit. put...into hands) “get”, “obtain”
[[[[N *no*] N] *ga*] V_{30}]	化け−の−皮−が−剥げる *bake-no-kawa-ga-hageru* (lit. peel off disguise) “expose the true colors”
[[[[N *no*] N] *ni*] V_{30}]	玉−の−輿−に−乗る *tama-no-kosi-ni-noru* (lit. ride on a palanquin for the nobility) “marry into wealth”
[[N *de*][[N *wo*] V_{30}]]	顎−で−人−を−使う *ago-de-hito-wo-tukau* (lit. use person by a chin) “order a person around”
[[N *ni*][[N *ga*] V_{30}]]	尻−に−火−が−付く *siri-ni-higa-tuku* (lit. buttocks catch fire) “get in great haste”
[[V_{23} *te*] V_{30}]	切っ−て−落とす *ki-te-otosu* (lit. cut and drop) “cut off”
[[V_{23} *ba*] V_{30}]	打て−ば−響く *ute-ba-hibiku* (lit. reverberate if hit) “respond quickly”
[[[[N *ni*] V_{23}] *te*] V_{30}]	束−に−なっ−て−掛かる *taba-ni-nat-te-kakaru* (lit. attack someone by becoming a bunch) “attack all at once”
[Adv [[N *ga*] V_{30}]]	どっと−疲れ−が−出る *dotto-tukare-ga-deru* (lit. fatigue bursts out) “being suddenly overcome with fatigue”

Table 5: Examples of structural types of verbal MWEs (N: noun, V_{23}: verb (adverbial form), V_{30}: verb (end form), Adv: adverb, *wo*, *ga*, *ni*, *no*, *de*, *te*, and *ba*: particle)

4.3.2 Coordinate Structure

Approximately 2,500 MWEs in the JDMWE contain internal coordinate structures. This information is described in Field-S by bracketing with "<" and ">", and the coordinated parts by "(" and ")". The coordinative phrase specification usually requires that the conjuncts must be parallel with respect to the syntactic function of the constituents appearing in the bracketed description. For example, an expression 後-は-野-と-なれ-山-と-なれ *ato-ha-no-to-nare-yama-to-nare* (lit. "the rest might become either a field or a mountain") "what will be, will be", has an internal coordinate structure. Thus, its Field-S is [[N *ha*]<([[N *to*] V_{60}])([[N *to*] V_{60}])>]. This description represents the structure shown in Figure 5, where V_{60} denotes an imperative form of the verb.

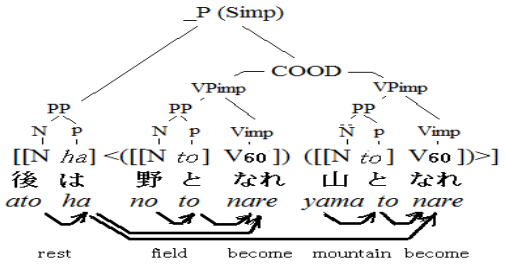


Figure 5: Example of the coordinate structure shown by "<" and ">" in Field-S

4.3.3 Non-phrasal Structure

Approximately 250 MWEs in the JDMWE are syntactically ill-formed in the sense of context-free grammar but still form a syntactic unit on their own. For example, 揺り籠-から-墓場-まで *yurikago-kara-hakaba-made* "from the cradle to the grave" is an adjunct of two postpositional phrases but is often used as a state-describing noun as in 揺り籠-から-墓場-まで-の-保証 *yurikago-kara-hakaba-made-no-hoshou* (lit. security of from cradle to grave) "security from the cradle to the grave". Thus Field-F and Field-S have a functional code N_k and a description [[N *kara*][N *made*] $]], respectively. The symbol "$" denotes a null constituent occupying the position of the governor on which this MWE depends. This structure is shown in Figure 6.

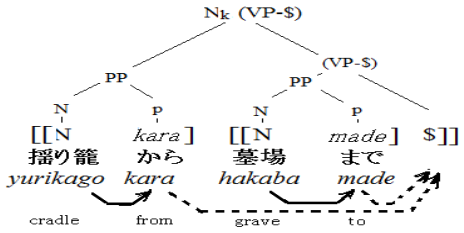


Figure 6: Example of a non-phrasal expression with a null constituent marked with "$" in Field-S

The total number of structural types specified in Field-S is nearly 6,000. This indicates that Japanese MWEs present a wide structural variety.

4.3.4 Internal Modifiability

Some MWEs are not fixed-length word strings, but allow the occurrence of phrasal modifiers internally. In our system, this aspect is captured by prefixing a modifiable element of the structural description stated in the Field-S with an asterisk "*". An adverbial MWE 上-に-述べ-た-様-に *ue-ni-nobe-ta-you-ni* "as I explained above" is one such MWE and thus has a description [[[[[N *ni*] *V_{23}] *ta*] N] *ni*] in Field-S, meaning that the third element V_{23} is a verb that can be modified internally by adverb phrases. Since the asterisk designates such optional phrasal modification, our system allows a derivative expression like 理由 -を-上-に-詳しく-述べ-た-様-に *riyuu-wo-ue-ni-kuwasiku-nobe-ta-you-ni* "as I explained in detail the reason above", which contains two additional, internal modifiers. The structure is shown in Figure 7.[4]

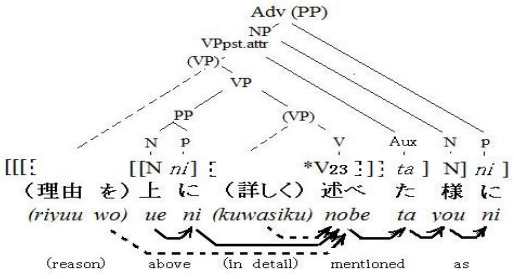


Figure 7: Example of internal modifiability marked by "*" in Field-S

[4] The positions to be taken by an internal modifier can be easily decided by the structural description given in Field-S along with the nest structure requirement.

Roughly speaking, 30,000 MWEs in the JDMWE have no asterisk in their Field-S. Our rigid examination reveals that internal modification is not allowed for them.

5 Contextual Condition (Field-C_f , C_b)

Approximately 6,700 MWEs need to be classified differently because they require particular forward contexts, i.e., they require co-occurrence of a particular syntactic phrase in the context that immediately precedes them. For example, 顔-を-する *kao-wo-suru* (lit. "do face") which is a support-verb construction, cannot occur without an immediately preceding adnominal modifier, e.g., the adjective 悲しい *kanasii* "sad", yielding 悲しい-顔-を-する *kanasii-kao-wo-suru* (lit. "do sad face") "make a sad face". This adnominal modifier co-occurrence requirement is stipulated in Field-C_f by a code <adnom. modifier>. There are about 30 of these forward contextual requirements. Similarly, backward contextual requirements, of which there are about 70, are stated in Field-C_b. Approximately 300 MWEs require particular backward contexts.

6 Statistical Properties

Without a rule system of semantic composition, it is difficult to evaluate the validity of the JDMWE concerning idiomaticity. However, we can confirm that 3,600 Japanese standard idioms that Sato (2007) listed from five Japanese idiom dictionaries published for human readers are included in the JDMWE as a proper subset. In addition, the JDMWE contains the information about their syntactic functions, structures, and flexibilities.

6.1 Comparison with Web N-gram Frequency Data

We examined the statistical properties of the JDMWE using the Japanese Web N-gram, version 1: LDC2009T08, which is a word N-gram ($1 \leq N \leq 7$) frequency dataset generated from 2×10^{10} sentences in a Japanese Web corpus, supplied by Google Inc. (Kudo et al. 2009). We will refer to this (or the Web corpus examined) subsequently as GND. We will refer to trigram $w_1w_2w_3$ as an NpV-trigram only when w_1 and w_3 are restricted to a noun and a verb (end form), respectively, and w_2 is one of the following case-particles: accusative を *wo*, subjective が *ga*, or dative に *ni*.[5] We write the number of occurrences of an expression x, counted in the GND, as C(x).

First, we obtain from the GND sets G, T, D, B, and R_i's defined below, using a Japanese word dictionary IPADIC (Asahara et al. 2003):

G={$w_1w_2w_3$| $w_1w_2w_3 \in$ GND, $w_1w_2w_3$ is an NpV-trigram.}
T={$w_1w_2w_3$| $w_1w_2w_3 \in$ JDMWE, $w_1w_2w_3$ is an NpV-trigram.}
D={w_1w_2|$\exists w_3$, $w_1w_2w_3 \in G$}
B={w_1w_2|$\exists w_3$, $w_1w_2w_3 \in T$}
R_i={$w_1w_2w_3$| $w_1w_2w_3 \in T$, C($w_1w_2w_3$) is the i-th largest among C(w_1w_2v)'s for all $w_1w_2v \in G$}.

We then found the following data:
・$|B|$=10,548
・$|D|$=110,822
・$|R_1|$=4,983, $|R_2|$=1,495, $|R_3|$=786, $|R_4|$=433, …
From these, we realize, for example, that 47.2% =($|R_1|/|B|$)×100 of trigrams in T have verbs that occur most frequently in the GND, succeeding the individual bigrams. An example of such a trigram is アクション-を-起こす *akushon-wo-okosu* (lit. "raise action") "take action". Similarly, 14.0%= ($|R_2|/|B|$)×100 have the second most frequent verbs, 7.5% have the third most frequent verbs, and so on. Figure 8(a) illustrates the results. From this, we can assume that the higher probability $\boldsymbol{p}_f(w_3|w_1w_2)$ a trigram $w_1w_2w_3$ has, the more likely w_3 is chosen for each w_1w_2 in the JDMWE. This is consistent with what we wrote in section 3.2. Figure 8(b) is the accumulative substitute of Figure 8(a). Extrapolating Figure 8(b) suggests that 10% of NpV-trigrams in the JDMWE do not occur in the GND. This implies that the size, i.e., 2×10^{10} sentences of the Web corpus used by the GND is not sufficiently large to allow MWE extraction.[6]

[5] The NpV-trigrams represent the typical forms of shortest Japanese sentences, corresponding roughly to subject-verb, verb-object/direct, and verb-object/indirect constructions in English.
[6] Otherwise, the frequency cut-off point of 20 adopted in GND is too high.

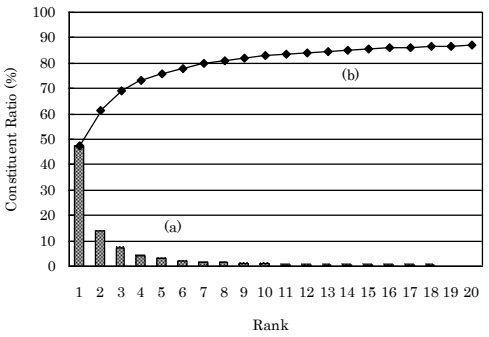


Figure 8 (a): Constituent ratio ($|R_i|/|B|$)×100 for rank i of probability $\boldsymbol{p}_f(w_3|w_1w_2)$; (b): Accumulative variant of (a) for rank i of probability $\boldsymbol{p}_f(w_3|w_1w_2)$

Second, we calculate the (normalized) entropy $H_f(w_3|w_1w_2)$ for each $w_1w_2 \in D$ defined below, where the probability $\boldsymbol{p}_f(w_3|w_1w_2)$ is estimated by $C(w_1w_2w_3)/C(w_1w_2)$. This provides a measure of the flatness of the $\boldsymbol{p}_f(w_3|w_1w_2)$ distribution canceling out the influence of the number N of verb types w_3's.

$$H_f(w_3|w_1w_2) = -\Big(\sum_{w_3} \boldsymbol{p}_f(w_3|w_1w_2) \log \boldsymbol{p}_f(w_3|w_1w_2)\Big) / \log N$$

After arranging 110,822 bigrams in *D* in ascending order of $H_f(w_3|w_1w_2)$, we divided them into 20 intervals $A_1, A_2, \ldots, A_{20}$ each with an equal number of bigrams (5,542). We then examined how many bigrams in *B* were included in each interval. Figures 9(a) and (b) plot the resulting constituent ratio of the bigrams in *B* and the mean value of $H_f(w_3|w_1w_2)$'s in each interval, respectively. We found, for example, that 1,262 out of 5,542 bigrams are in *B* for the first interval, i.e., the constituent ratio is 22.8%=(1,262/5,542) × 100. Similarly, we obtain 22.5%=(1,248/5,542) × 100 for the second interval, 20.5%=(1,136/5,542)×100 for the third, and so on. From this, we realize the macroscopic tendency that the larger the entropy $H_f(w_3|w_1w_2)$, or equivalently the perplexity of the succeeding verb w_3, a bigram w_1w_2 has, the less likely it is adopted as a prefix of a trigram in *T*.

Taking the results in Figure 8 and Figure 9 together, we can presume that not only frequently but also exclusively occurring verbs would be the preferred choice in *T*.

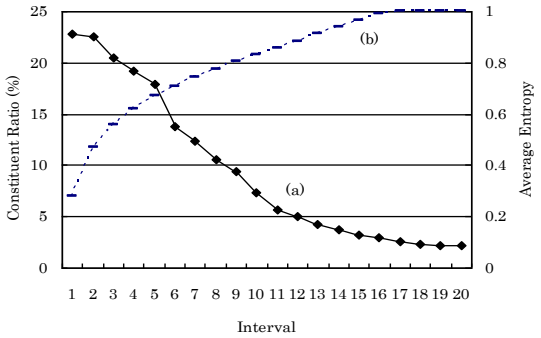


Figure 9 (a): Constituent ratio of the bigrams in *B* among bigrams in *D* in interval k ($1 \le k \le 20$); (b): Mean value of entropies $H_f(w_3|w_1w_2)$'s in the interval k ($1 \le k \le 20$)

This suggests the general feasibility of the JDMWE, for its relative compactness, in effectively disambiguating the syntactic structures of input word strings.

The above investigations were carried out on the forward conditional probabilities for restricted types of MWEs. However, the results imply a general validity of the JDMWE since the same criteria for selection were applied to all kinds of multiword expressions.

6.2 Occurrences in Newspapers

We examined 2,500 randomly selected sentences in Nikkei newspaper articles (published in 2009) to determine how many MWE tokens of the JDMWE occur in them. We found that in 100 sentences an average of 74 tokens of our MWEs were used. This suggests a large lexical coverage of the JDMWE.

7 Concluding Remarks

The JDMWE is a slotted tree bank for idiosyncratic multiword expressions, annotated with detailed notational, syntactic information.

The idea underlying the JDMWE is that the volume and meticulousness of the lexical resource crucially affects the outcome of the rule-oriented, large-scale NLP. In view of this, the JDMWE was designed to encompass the wide range of linguistic objects related to Japanese MWEs, by placing importance on the recall rate in the selection of the

candidate expressions.[7] The statistical properties clarified in this paper imply the general feasibility of the JDMWE at least in the probabilistic respect.

Possible fields of application of the JDMWE include, for example:

- Phrase-based machine translation
- Phrase-based speech recognition
- Phrase-based kana-to-kanji conversion
- Search engine for Japanese corpus
- Paraphrasing system
- Japanese dialoguer
- Japanese language education system

Another aspect of the JDMWE is that it would provide linguists with lexicological data. For example, the usage of Japanese onomatopoeic adverbs, which are mostly bound probabilistically to specific verbs or adjectives, is extensively catalogued in the JDMWE.

The first version of the JDMWE will be released after proofreading.[8] If possible, we would like to add further information to each MWE on morphological variants, passivization, relativization, decomposability, paraphrasing, and semantic disambiguation for future versions.

Acknowledgments

We would like to thank the late Professors Toshihiko Kurihara and Sho Yoshida, who inspired our current research in the 1970s. Similar thanks go to Makoto Nagao. We are also grateful to everyone who assisted in the development of the JDMWE. Further special thanks go to Akira Shimazu, Takano Ogino, and Kenji Yoshimura for their encouragement and useful discussions, to those who worked on the LDC2009T08 and IPADIC, to the three anonymous reviewers for their valuable comments and advice, and to Stephan Howe for advice on matters of English style in the current paper.

References

Asahara, M. and Matsumoto, Y. 2003. IPADIC version 2.7.0 User's Manual (in Japanese). NAIST, Information Science Division.

Baldwin, T. and Bond, F. 2003. Multiword Expressions: Some Problems for Japanese NLP. Proceedings of the 8th Annual Meeting of the Association for Natural Language Processing (Japan): 379–382.

Bannard, C. 2007. A Measure of Syntactic Flexibility for Automatically Identifying Multiword Expressions in Corpora. Proceedings of A Broader Perspective on Multiword Expressions, Workshop at the ACL 2007 Conference: 1–8.

Baptista, J., Correia, A., and Fernandes, G. 2004. Frozen Sentences of Portuguese: Formal Descriptions for NLP. Proceedings of ACL 2004 Workshop on Multiword Expressions: Integrating Processing: 72–79.

Fazly, A. and Stevenson, S. 2006. Automatically Constructing a Lexicon of Verb Phrase Idiomatic Combinations. Proceedings of the 11th Conference of the European Chapter of the ACL: 337–344.

Fellbaum, C. (ed.) 1999. WordNet. An Electronic Lexical Database, Cambridge, MA: MIT Press.

Fellbaum, C., Geyken, A., Herold, A., Koerner, F., and Neumann, G. 2006. Corpus-Based Studies of German Idioms and Light Verbs. International Journal of Lexicography, Vol. 19, No. 4: 349-360.

Gross, M. 1986. Lexicon-Grammar. The Representation of Compound Words. Proceedings of the 11th International Conference on Computational Linguistics, COLING86:1–6.

Hashimoto, C. and Kawahara, D. 2009. Compilation of an Idiom Example Database for Supervised Idiom Identification. Language Resource and Evaluation Vol. 43, No. 4 : 355-384.

Jackendoff, R. 1997. The Architecture of Language Faculty. Cambridge, MA: MIT Press.

Koyama, Y., Yasutake, M., Yoshimura, K., and Shudo, K. 1998. Large Scale Collocation Data and Their Application to Japanese Word Processor Technology. Proceedings of the 17th International Conference on Computational Linguistics, COLING98: 694–698.

Kudo, T. and Kazawa, H. 2009. Japanese Web N-gram Version 1. Linguistic Data Consortium, Philadelphia.

Kuiper, K., McCan, H., Quinn, H., Aitchison, T., and Van der Veer, K. 2003. SAID: A Syntactically Anno tated Idiom Dataset. Linguistic Data Consortium 2003T10.

Laporte, É. and Voyatzi, S. 2008. An Electronic Dictionary of French Multiword Adverbs. Proceedings of the LREC Workshop towards a Shared Task for Multiword Expressions (MWE 2008): 31–34.

[7] The time required to compile this dictionary is estimated at 24,000 working hours.

[8] A portion of the JDMWE is available at http://jefi.info/.

Pantel, P. and Lin, D. 2001. A Statistical Corpus-Based Term Extractor. Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence, Springer-Verlag: 36–46.

Sag, I. A., Baldwin, T., Bond, F., Copestake, A., and Flickinger, D. 2002. Multiword Expressions: A Pain in the Neck for NLP. Proceedings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics, CICLING2002: 1–15.

Sato, S. 2007. Compilation of a Comparative List of Basic Japanese Idioms from Five Sources (in Japanese). IPSJ SIG Notes 178: 1-6.

Shudo, K., Narahara, T., and Yoshida, S. 1980. Morphological Aspect of Japanese Language Processing. Proceedings of the 8th International Conference on Computational Linguistics, COLING80: 1–8.

Shudo, K., Tanabe, T., Takahashi, M., and Yoshimura, K. 2004. MWEs as Non-Propositional Content Indicators. Proceedings of ACL 2004 Workshop on Multiword Expressions: Integrating Processing: 31–39.

Uchiyama, K. and Ishizaki, S. 2003. A Disambiguation of Compound Verbs. Proceedings of ACL 2003. Workshop on Multiword Expressions: Analysis, Acquisition and Treatment: 81–88.

Villavicencio, A. 2004. Lexical Encoding of MWEs. Proceedings of ACL 2004 Workshop on Multiword Expressions: Integrating Processing: 80–87.

Multi-Modal Annotation of Quest Games in Second Life

Sharon Gower Small, Jennifer Stromer-Galley and Tomek Strzalkowski
ILS Institute
State University of New York at Albany
Albany, NY 12222
small@albany.edu, jstromer@albany.edu, tomek@albany.edu

Abstract

We describe an annotation tool developed to assist in the creation of multimodal action-communication corpora from on-line massively multi-player games, or MMGs. MMGs typically involve groups of players (5-30) who control their avatars[1], perform various activities (questing, competing, fighting, etc.) and communicate via chat or speech using assumed screen names. We collected a corpus of 48 group quests in Second Life that jointly involved 206 players who generated over 30,000 messages in quasi-synchronous chat during approximately 140 hours of recorded action. Multiple levels of co-ordinated annotation of this corpus (dialogue, movements, touch, gaze, wear, etc) are required in order to support development of automated predictors of selected real-life social and demographic characteristics of the players. The annotation tool presented in this paper was developed to enable efficient and accurate annotation of all dimensions simultaneously.

1 Introduction

The aim of our project is to predict the real world characteristics of players of massively-multiplayer online games, such as Second Life (SL). We sought to predict actual player attributes like age or education levels, and personality traits including leadership or conformity. Our task was to do so using only the behaviors, communication, and interaction among the players produced during game play. To do so, we logged all players' avatar movements, "touch events" (putting on or taking off clothing items, for example), and their public chat messages (i.e., messages that can be seen by all players in the group). Given the complex nature of interpreting chat in an online game environment, we required a tool that would allow annotators to have a synchronized view of both the event action as well as the chat utterances. This would allow our annotators to correlate the events and the chat by marking them simultaneously. More importantly, being able to view game events enables more accurate chat annotation; and conversely, viewing chat utterances helps to interpret the significance of certain events in the game, e.g., one avatar following another. For example, an exclamation of: "I can't do it!" could be simply a response (rejection) to a request from another player; however, when the game action is viewed and the speaker is seen attempting to enter a building without success, another interpretation may arise (an assertion, a call for help, etc.).

The Real World (RW) characteristics of SL players (and other on-line games) may be inferred to varying degrees from the appearance of their avatars, the behaviors they engage in, as well as from their on-line chat communications. For example, the avatar gender generally matches the gender of the owner; on the other hand, vocabulary choices in chat are rather poor predictors of a player's age, even though such correlation is generally seen in real life conversation.

Second Life[2] was the chosen platform because of the ease of creating objects, controlling the play environment, and collecting players' movement, chat, and other behaviors. We generated a corpus of chat and movement data from 48 quests comprised of 206 participants who generated over 30,000

[1] All avatar names seen in this paper have been changed to protect players' identities.

[2] An online Virtual World developed and launched in 2003, by Linden Lab, San Francisco, CA. http://secondlife.com

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 171–179,
Portland, Oregon, June 19-24, 2011.

messages and approximately 140 hours of recorded action. We required an annotation tool to help us efficiently annotate dialogue acts and communication links in chat utterances as well as avatar movements from such a large corpus. Moreover, we required correlation between these two dimensions of chat and movement since movement and other actions may be both causes and effects of verbal communication. We developed a multimodal event and chat annotation tool (called RAT, the Relational Annotation Tool), which will simultaneously display a 2D rendering of all movement activity recorded during our Second Life studies, synchronized with the chat utterances. In this way both chat and movements can be annotated simultaneously: the avatar movement actions can be reviewed while making dialogue act annotations. This has the added advantage of allowing the annotator to see the relationships between chat, behavior, and location/movement. This paper will describe our annotation process and the RAT tool.

2 Related Work

Annotation tools have been built for a variety of purposes. The CSLU Toolkit (Sutton et al., 1998) is a suite of tools used for annotating spoken language. Similarly, the EMU System (Cassidy and Harrington, 2001) is a speech database management system that supports multi-level annotations. Systems have been created that allow users to readily build their own tools such as AGTK (Bird et al., 2001). The multi-modal tool DAT (Core and Allen, 1997) was developed to assist testing of the DAMSL annotation scheme. With DAT, annotators were able to listen to the actual dialogues as well as view the transcripts. While these tools are all highly effective for their respective tasks, ours is unique in its synchronized view of both event action and chat utterances.

Although researchers studying online communication use either off-the shelf qualitative data analysis programs like Atlas.ti or NVivo, a few studies have annotated chat using custom-built tools. One approach uses computer-mediated discourse analysis approaches and the Dynamic Topic Analysis tool (Herring, 2003; Herring & Nix; 1997; Stromer-Galley & Martison, 2009), which allows annotators to track a specific phenomenon of online interaction in chat: topic shifts during an interaction. The Virtual Math Teams project (Stahl, 2009) created a tool that allowed for the simultaneous playback of messages posted to a quasi-synchronous discussion forum with whiteboard drawings that student math team members used to illustrate their ideas or visualize the math problem they were trying to solve (Çakir, 2009).

A different approach to data capture of complex human interaction is found in the AMI Meeting Corpus (Carletta, 2007). It captures participants' head movement information from individual head-mounted cameras, which allows for annotation of nodding (consent, agreement) or shaking (disagreement), as well as participants' locations within the room; however, no complex events involving series of movements or participant proximity are considered. We are unaware of any other tools that facilitate the simultaneous playback of multi-modes of communication and behavior.

3 Second Life Experiments

To generate player data, we rented an island in Second Life and developed an approximately two hour quest, the Case of the Missing Moonstone. In this quest, small groups of 4 to 5 players, who were previously unacquainted, work their way together through the clues and puzzles to solve a murder mystery. We recruited Second Life players in-game through advertising and setting up a shop that interested players could browse. We also used Facebook ads, which were remarkably effective.

The process of the quest experience for players started after they arrived in a starting area of the island (the quest was open only to players who were made temporary members of our island) where they met other players, browsed quest-appropriate clothing to adorn their avatars, and received information from one of the researchers. Once all players arrived, the main quest began, progressing through five geographic areas in the island. Players were accompanied by a "training sergeant", a researcher using a robot avatar, that followed players through the quest and provided hints when groups became stymied along their investigation but otherwise had little interaction with the group.

The quest was designed for players to encounter obstacles that required coordinated action, such as all players standing on special buttons to activate a door, or the sharing of information between players, such as solutions to a word puzzle, in order to advance to the next area of the quest (Figure 1).

Slimy Roastbeef*: "who's got the square gear?"*

Kenny Superstar: *"I do, but I'm stuck"*

Slimy Roastbeef: *"can you hand it to me?"*

Kenny Superstar*: "i don't know how"*

Slimy Roastbeef: *"open your inventory, click and drag it onto me"*

Figure 1: Excerpt of dialogue during a coordination activity

Quest activities requiring coordination among the players were common and also necessary to ensure a sufficient degree of movement and message traffic to provide enough material to test our predictions, and to allow us to observe particular social characteristics of players. Players answered a survey before and then again after the quest, providing demographic and trait information and evaluating other members of their group on the characteristics of interest.

3.1 Data Collection

We recorded all players' avatar movements as they purposefully moved avatars through the virtual spaces of the game environment, their public chat, and their "touch events", which are the actions that bring objects out of player inventories, pick up objects to put in their inventories, or to put objects, such as hats or clothes, onto the avatars, and the like. We followed Yee and Bailenson's (2008) technical approach for logging player behavior. To get a sense of the volume of data generated, 206 players generated over 30,000 messages into the group's public chat from the 48 sessions. We compiled approximately 140 hours of recorded action. The avatar logger was implemented to record each avatar's location through their (x,y,z) coordinates, recorded at two second intervals. This information was later used to render the avatar's position on our 2D representation of the action (section 4.1).

4 RAT

The Relational Annotation Tool (RAT) was built to assist in annotating the massive collection of data collected during the Second Life experiments. A tool was needed that would allow annotators to see the textual transcripts of the chat while at the same time view a 2D representation of the action. Additionally, we had a textual transcript for a select set of events: touch an object, stand on an object, attach an object, etc., that we needed to make available to the annotator for review.

These tool characteristics were needed for several reasons. First, in order to fully understand the communication and interaction occurring between players in the game environment and accurately annotate those messages, we needed annotators to have as much information about the context as possible. The 2D map coupled with the events information made it easier to understand. For example, in the quest, players in a specific zone, encounter a dead, maimed body. As annotators assigned codes to the chat, they would sometimes encounter exclamations, such as "ew" or "gross". Annotators would use the 2D map and the location of the exclaiming avatar to determine if the exclamation was a result of their location (in the zone with the dead body) or because of something said or done by another player. Location of avatars on the 2D map synchronized with chat was also helpful for annotators when attempting to disambiguate communicative links. For example, in one subzone, mad scribblings are written on a wall. If player A says "You see that scribbling on the wall?" the annotator needs to use the 2D map to see who the player is speaking to. If player A and player C are both standing in that subzone, then the annotator can make a reasonable assumption that player A is directing the question to player C, and not player B who is located in a different subzone. Second, we annotated coordinated avatar movement actions (such as following each other into a building or into a room), and the only way to readily identify such complex events was through the 2D map of avatar movements.

The overall RAT interface, Figure 2, allows the annotator to simultaneously view all modes of representation. There are three distinct panels in this interface. The left hand panel is the 2D representation of the action (section 4.1). The upper right hand panel displays the chat and event transcripts (section 4.2), while the lower right hand portion is reserved for the three annotator sub-panels (section 4.3).

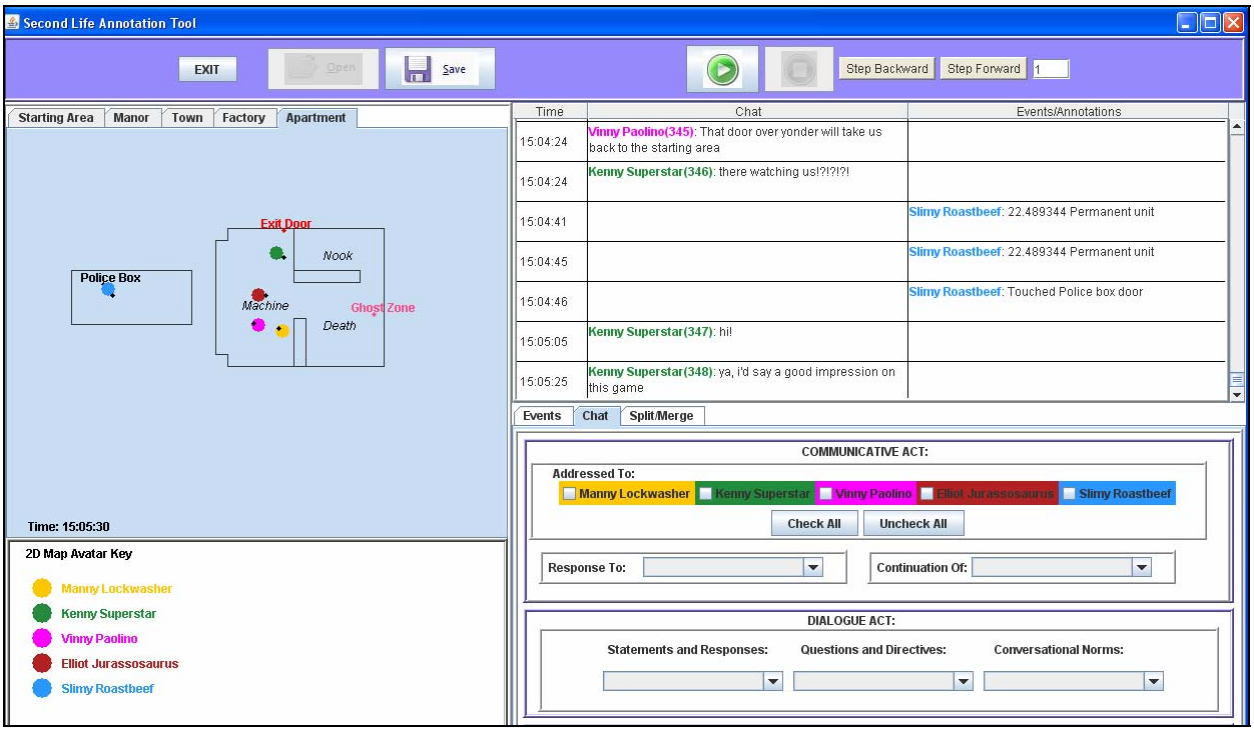


Figure 2: RAT interface

4.1 The 2D Game Representation

The 2D representation was the most challenging of the panels to implement. We needed to find the proper level of abstraction for the action, while maintaining its usefulness for the annotator. Too complex a representation would cause cognitive overload for the annotator, thus potentially deteriorating the speed and quality of the annotations. Conversely, an overly abstract representation would not be of significant value in the annotation process.

There were five distinct geographic areas on our Second Life Island: *Starting Area, Mansion, Town Center, Factory and Apartments*. An overview of the area in Second Life is displayed in Figure 3. We decided to represent each area separately as each group moves between the areas together, and it was therefore never necessary to display more than one area at a time. The 2D representation of the Mansion Area is displayed in Figure 4 below. Figure 5 is an exterior view of the actual Mansion in Second Life. Each area's fixed representation was rendered using Java Graphics, reading in the Second Life (x,y,z) coordinates from an XML data file. We represented the walls of the buildings as connected solid black lines with openings left for doorways. Key item locations were marked and labeled, e.g. *Kitten, maid, the Idol*, etc. Even though annotators visited the island to familiarize themselves with the layout, many mansion rooms were labeled to help the annotator recall the layout of the building, and minimize error of annotation based on flawed recall. Finally, the exact time of the action that is currently being represented is displayed in the lower left hand corner.

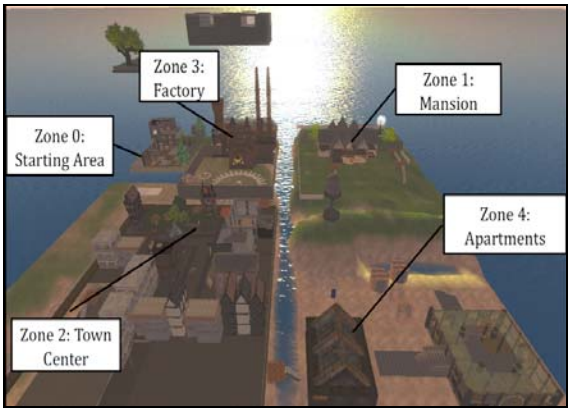


Figure 3: Second Life overview map

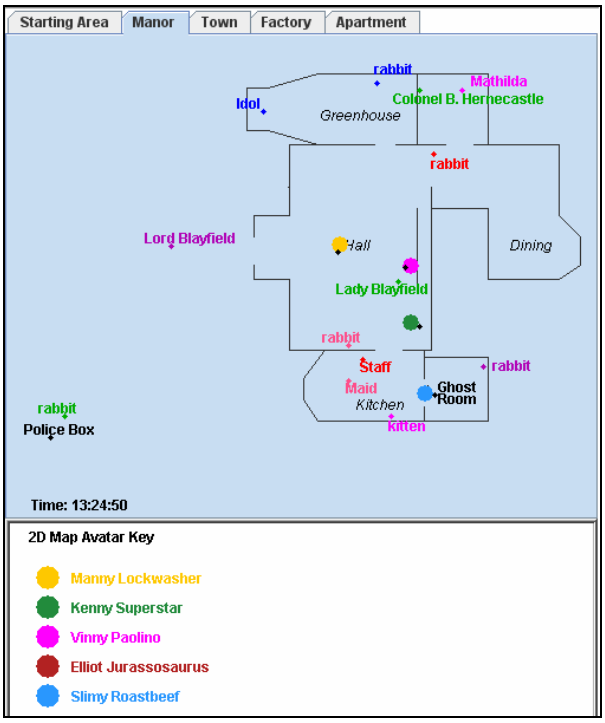


Figure 4: 2D representation of Second Life action inside the Mansion/Manor

Figure 5: Second Life view of Mansion exterior

Avatar location was recorded in our log files as an (x,y,z) coordinate at a two second interval. Avatars were represented in our 2D panel as moving solid color circles, using the x and y coordinates. A color coded avatar key was displayed below the 2D representation. This key related the full name of every avatar to its colored circle representation. The z coordinate was used to determine if the avatar was on the second floor of a building. If the z value indicated an avatar was on a second floor, their icon was modified to include the number "2" for the duration of their time on the second floor. Also logged was the avatar's degree of rotation. Using this we were able to represent which direction the avatar was looking by a small black dot on their colored circle.

As the annotators stepped through the chat and event annotation, the action would move forward, in synchronized step in the 2D map. In this way at any given time the annotator could see the avatar action corresponding to the chat and event transcripts appearing in the right panels. The annotator had the option to step forward or backward through the data at any step interval, where each step corresponded to a two second increment or decrement, to provide maximum flexibility to the annotator in viewing and reviewing the actions and communications to be annotated. Additionally, "Play" and "Stop" buttons were added to the tool so the annotator may simply watch the action play forward rather than manually stepping through.

4.2 The Chat & Event Panel

Avatar utterances along with logged Second Life events were displayed in the Chat and Event Panel (Figure 6). Utterances and events were each displayed in their own column. Time was recorded for every utterance and event, and this was displayed in the first column of the Chat and Event Panel. All avatar names in the utterances and events were color coded, where the colors corresponded to the avatar color used in the 2D panel. This panel was synchronized with the 2D Representation panel and as the annotator stepped through the game action on the 2D display, the associated utterances and events populated the Chat and Event panel.

Time	Chat	Events/Annotations
13:47:40		Kenny Superstar: Touched Square gear
13:47:46		Slimy Roastbeef: Touched Round gear
13:48:01	Kenny Superstar(142): we have to put all the gears in place	
13:48:06	Slimy Roastbeef(143): put the triangle gear on there or something	
13:48:10		Elliot Jurassosaurus: Touched Sharp gear
13:48:25	Manny Lockwasher(144): how	
13:48:26	Slimy Roastbeef(145): probably have to detach from your hand first	

Figure 6: Chat & Event Panel

4.3 The Annotator Panels

The Annotator Panels (Figures 7 and 10) contains all features needed for the annotator to quickly annotate the events and dialogue. Annotators could choose from a number of categories to label each dialogue utterance. Coding categories included communicative links, dialogue acts, and selected multi-avatar actions. In the following we briefly outline each of these. A more detailed description of the chat annotation scheme is available in (Shaikh et al., 2010).

4.3.1 Communicative Links

One of the challenges in multi-party dialogue is to establish which user an utterance is directed towards. Users do not typically add addressing information in their utterances, which leads to ambiguity while creating a communication link between users. With this annotation level, we asked the annotators to determine whether each utterance was addressed to some user, in which case they were asked to mark which specific user it was addressed to; was in response to another prior utterance by a different user, which required marking the specific utterance responded to; or a continuation of the user's own prior utterance.

Communicative link annotation allows for accurate mapping of dialogue dynamics in the multi-party setting, and is a critical component of tracking such social phenomena as disagreements and leadership.

4.3.2 Dialogue Acts

We developed a hierarchy of 19 dialogue acts for annotating the functional aspect of the utterance in the discussion. The tagset we adopted is loosely based on DAMSL (Allen & Core, 1997) and SWBD (Jurafsky et al., 1997), but greatly reduced and also tuned significantly towards dialogue pragmatics and away from more surface characteristics of utterances. In particular, we ask our annotators what is the pragmatic function of each utterance within the dialogue, a decision that often depends upon how earlier utterances were classified. Thus augmented, DA tags become an important source of evidence for detecting language uses and such social phenomena as conformity. Examples of dialogue act tags include Assertion-Opinion, Acknowledge, Information-Request, and Confirmation-Request.

Using the augmented DA tagset also presents a fairly challenging task to our annotators, who need to be trained for many hours before an acceptable rate of inter-annotator agreement is achieved. For this reason, we consider our current DA tagging as a work in progress.

4.3.3 Zone coding

Each of the five main areas had a corresponding set of subzones. A subzone is a building, a room within a building, or any other identifiable area within the playable spaces of the quest, e.g. the *Mansion* has the subzones: *Hall*, *Dining Room*, *Kitchen*, *Outside*, *Ghost Room*, etc. The subzone was determined based on the avatar(s) (x,y,z) coordinates and the known subzone boundaries. This additional piece of data allowed for statistical analysis at different levels: avatar, dialogue unit, and subzone.

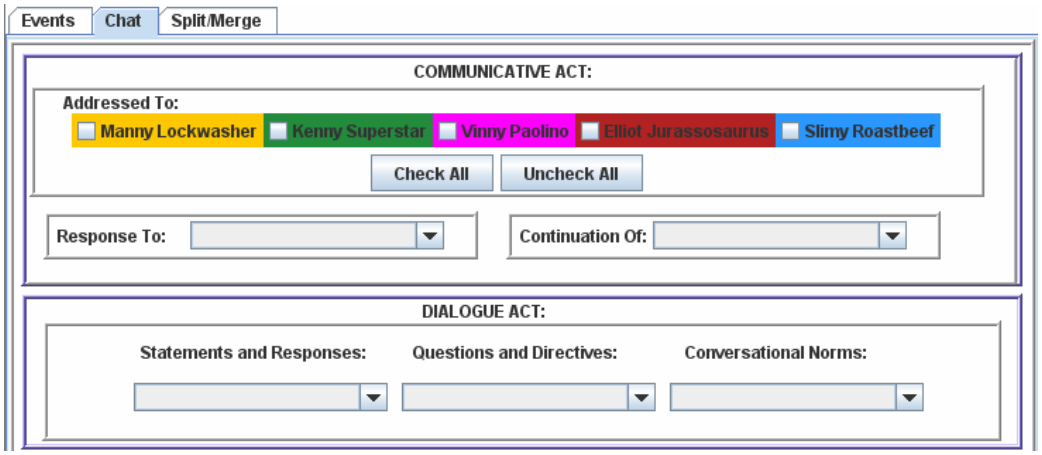


Figure 7: Chat Annotation Sub-Panel

4.3.4 Multi-avatar events

As mentioned, in addition to chat we also were interested in having the annotators record composite events involving multiple avatars over a span of time and space. While the design of the RAT tool will support annotation of any event of interest with only slight modifications, for our purposes, we were interested in annotating two types of events that we considered significant for our research hypotheses. The first type of event was the multi-avatar entry (or exit) into a sub-zone, including the order in which the avatars moved.

Figure 8 shows an example of a "Moves into Subzone" annotation as displayed in the Chat & Event Panel. Figure 9 shows the corresponding series of progressive moments in time portraying entry into the Bank subzone as represented in RAT. In the annotation, each avatar name is recorded in order of its entry into the subzone (here, the Bank). Additionally, we record the subzone name and the time the event is completed[3].

The second type of event we annotated was the "follow X" event, i.e., when one or more avatars appeared to be following one another within a subzone. These two types of events were of particular interest because we hypothesized that players who are leaders are likely to enter first into a subzone and be followed around once inside.

In addition, support for annotation of other types of composite events can be added as needed; for example, group forming and splitting, or certain joint activities involving objects, etc. were fairly common in quests and may be significant for some analyses (although not for our hypotheses).

For each type of event, an annotation subpanel is created to facilitate speedy markup while minimizing opportunities for error (Figure 10). A "Moves Into Subzone" event is annotated by recording the ordinal (1, 2, 3, etc.) for each avatar. Similarly, a "Follows" event is coded as avatar group "A" follows group "B', where each group will contain one or more avatars.

Figure 8: The corresponding annotation for Figure 9 event, as displayed in the Chat & Event Panel

5 The Annotation Process

To annotate the large volume of data generated from the Second Life quests, we developed an annotation guide that defined and described the annotation categories and decision rules annotators were to follow in categorizing the data units (following previous projects (Shaikh et al., 2010). Two students were hired and trained for approximately 60 hours, during which time they learned how to use the annotation tool and the categories and rules for the annotation process. After establishing a satisfactory level of interrater reliability (average Krippendorff's alpha of all measures was <0.8. Krippendorff's alpha accounts for the probability of

[3] We are also able to record the start time of any event but for our purposes we were only concerned with the end time.

chance agreement and is therefore a conservative measure of agreement), the two students then annotated the 48 groups over a four-month period. It took approximately 230 hours to annotate the sessions, and they assigned over 39,000 dialogue act tags. Annotators spent roughly 7 hours marking up the movements and chat messages per 2.5 hour quest session.

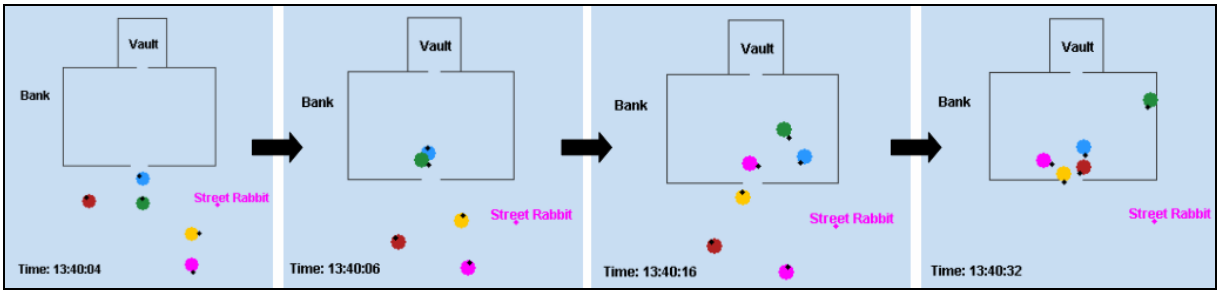


Figure 9: A series of progressive moments in time portraying avatar entry into the Bank subzone

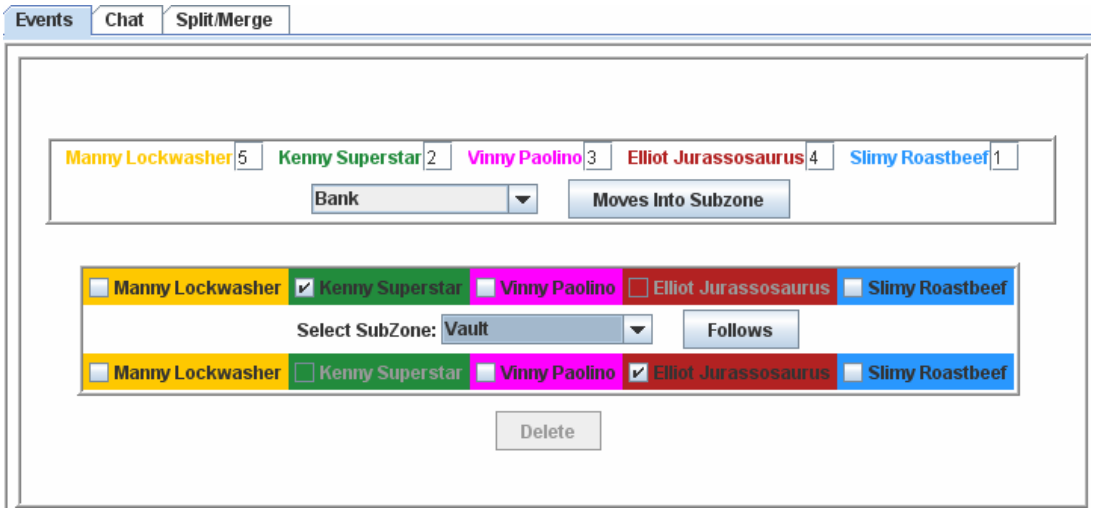


Figure 10: Event Annotation Sub-Panel, currently showing the "Moves Into Subzone" event from figure 9, as well as: "*Kenny follows Elliot in Vault*"

5.1 The Annotated Corpus

The current version of the annotated corpus consists of thousands of tagged messages including: 4,294 action-directives, 17,129 assertion-opinions, 4,116 information requests, 471 confirmation requests, 394 offer-commits, 3,075 responses to information requests, 1,317 agree-accepts, 215 disagree-rejects, and 2,502 acknowledgements, from 30,535 pre-split utterances (31,801 post-split). We also assigned 4,546 following events.

6 Conclusion

In this paper we described the successful implementation and use of our multi-modal annotation tool, RAT. Our tool was used to accurately and simultaneously annotate over 30,000 messages and approximately 140 hours of action. For each hour spent annotating, our annotators were able to tag approximately 170 utterances as well as 36 minutes of action.

The annotators reported finding the tool highly functional and very efficient at helping them easily assign categories to the relevant data units, and that they could assign those categories without producing too many errors, such as accidentally assigning the wrong category or selecting the wrong avatar. The function allowing for the synchronized playback of the chat and movement data coupled with the 2D map increased comprehension of utterances

and behavior of the players during the quest, improving validity and reliability of the results.

Acknowledgements
This research is part of an Air Force Research Laboratory sponsored study conducted by Colorado State University, Ohio University, the University at Albany, SUNY, and Lockheed Martin.

References

Steven Bird, Kazuaki Maeda, Xiaoyi Ma and Haejoong Lee. 2001. annotation tools based on the annotation graph API. In Proceedings of ACL/EACL 2001 Workshop on Sharing Tools and Resources for Research and Education.

M. P. Çakir. 2009. The organization of graphical, narrative and symbolic interactions. In Studying virtual math teams (pp. 99-140). New York, Springer.

J. Carletta. 2007. Unleashing the killer corpus: experiences in creating the multi-everything AMI Meeting Corpus. Language Resources and Evaluation Journal 41(2): 181-190.

Mark G. Core and James F. Allen. 1997. Coding dialogues with the DAMSL annotation scheme. In Proceedings of AAAI Fall 1997 Symposium.

Steve Cassidy and Jonathan Harrington. 2001. Multi-level annotation in the Emu speech database management system. Speech Communication, 33:61-77.

S. C. Herring. 2003. Dynamic topic analysis of synchronous chat. Paper presented at the New Research for New Media: Innovative Research Symposium. Minneapolis, MN.

S. C. Herring and Nix, C. G. 1997. Is "serious chat" an oxymoron? Pedagogical vs. social use of internet relay chat. Paper presented at the American Association of Applied Linguistics, Orlando, FL.

Samira Shaikh, Strzalkowski, T., Broadwell, A., Stromer-Galley, J., Taylor, S., and Webb, N. 2010. MPC: A Multi-party chat corpus for modeling social phenomena in discourse. *Proceedings of the Seventh Conference on International Language Resources and Evaluation.* Valletta, Malta: European Language Resources Association.

G. Stahl. 2009. The VMT vision. In G. Stahl, (Ed.), Studying virtual math teams (pp. 17-29). New York, Springer.

Stephen Sutton, Ronald Cole, Jacques De Villiers, Johan Schalkwyk, Pieter Vermeulen, Mike Macon, Yonghong Yan, Ed Kaiser, Brian Rundle, Khaldoun Shobaki, Paul Hosom, Alex Kain, Johan Wouters, Dominic Massaro, Michael Cohen. 1998. Universal Speech Tools: The CSLU toolkit. Proceedings of the 5th ICSLP, Australia.

Jennifer Stromer-Galley and Martinson, A. 2009. Coherence in political computer-mediated communication: Comparing topics in chat. *Discourse & Communication, 3,* 195-216.

N. Yee and Bailenson, J. N. 2008. A method for longitudinal behavioral data collection in *Second Life*. *Presence, 17,* 594-596.

A New Dataset and Method for Automatically Grading ESOL Texts

Helen Yannakoudakis
Computer Laboratory
University of Cambridge
United Kingdom
Helen.Yannakoudakis@cl.cam.ac.uk

Ted Briscoe
Computer Laboratory
University of Cambridge
United Kingdom
Ted.Briscoe@cl.cam.ac.uk

Ben Medlock
iLexIR Ltd
Cambridge
United Kingdom
ben@ilexir.co.uk

Abstract

We demonstrate how supervised discriminative machine learning techniques can be used to automate the assessment of 'English as a Second or Other Language' (ESOL) examination scripts. In particular, we use rank preference learning to explicitly model the grade relationships between scripts. A number of different features are extracted and ablation tests are used to investigate their contribution to overall performance. A comparison between regression and rank preference models further supports our method. Experimental results on the first publically available dataset show that our system can achieve levels of performance close to the upper bound for the task, as defined by the agreement between human examiners on the same corpus. Finally, using a set of 'outlier' texts, we test the validity of our model and identify cases where the model's scores diverge from that of a human examiner.

1 Introduction

The task of automated assessment of free text focuses on automatically analysing and assessing the quality of writing competence. Automated assessment systems exploit textual features in order to measure the overall quality and assign a score to a text. The earliest systems used superficial features, such as word and sentence length, as proxies for understanding the text. More recent systems have used more sophisticated automated text processing techniques to measure grammaticality, textual coherence, prespecified errors, and so forth.

Deployment of automated assessment systems gives a number of advantages, such as the reduced workload in marking texts, especially when applied to large-scale assessments. Additionally, automated systems guarantee the application of the same marking criteria, thus reducing inconsistency, which may arise when more than one human examiner is employed. Often, implementations include feedback with respect to the writers' writing abilities, thus facilitating self-assessment and self-tutoring.

Implicitly or explicitly, previous work has mostly treated automated assessment as a supervised text classification task, where training texts are labelled with a grade and unlabelled test texts are fitted to the same grade point scale via a regression step applied to the classifier output (see Section 6 for more details). Different techniques have been used, including cosine similarity of vectors representing text in various ways (Attali and Burstein, 2006), often combined with dimensionality reduction techniques such as Latent Semantic Analysis (LSA) (Landauer et al., 2003), generative machine learning models (Rudner and Liang, 2002), domain-specific feature extraction (Attali and Burstein, 2006), and/or modified syntactic parsers (Lonsdale and Strong-Krause, 2003).

A recent review identifies twelve different automated free-text scoring systems (Williamson, 2009). Examples include e-Rater (Attali and Burstein, 2006), Intelligent Essay Assessor (IEA) (Landauer et al., 2003), IntelliMetric (Elliot, 2003; Rudner et al., 2006) and Project Essay Grade (PEG) (Page, 2003). Several of these are now deployed in high-stakes assessment of examination scripts. Although there are many published analyses of the perfor-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 180–189,
Portland, Oregon, June 19-24, 2011.

mance of individual systems, as yet there is no publically available shared dataset for training and testing such systems and comparing their performance. As it is likely that the deployment of such systems will increase, standardised and independent evaluation methods are important. We make such a dataset of ESOL examination scripts available[1] (see Section 2 for more details), describe our novel approach to the task, and provide results for our system on this dataset.

We address automated assessment as a supervised discriminative machine learning problem and particularly as a rank preference problem (Joachims, 2002). Our reasons are twofold:

Discriminative classification techniques often outperform non-discriminative ones in the context of text classification (Joachims, 1998). Additionally, rank preference techniques (Joachims, 2002) allow us to explicitly learn an optimal ranking model of text quality. Learning a ranking directly, rather than fitting a classifier score to a grade point scale after training, is both a more generic approach to the task and one which exploits the labelling information in the training data efficiently and directly.

Techniques such as LSA (Landauer and Foltz, 1998) measure, in addition to writing competence, the semantic relevance of a text written in response to a given prompt. However, although our corpus of manually-marked texts was produced by learners of English in response to prompts eliciting free-text answers, the marking criteria are primarily based on the accurate use of a range of different linguistic constructions. For this reason, we believe that an approach which directly measures linguistic competence will be better suited to ESOL text assessment, and will have the additional advantage that it may not require retraining for new prompts or tasks.

As far as we know, this is the first application of a rank preference model to automated assessment (hereafter AA). In this paper, we report experiments on rank preference Support Vector Machines (SVMs) trained on a relatively small amount of data, on identification of appropriate feature types derived automatically from generic text processing tools, on comparison with a regression SVM model, and on the robustness of the best model to 'outlier' texts. We report a consistent, comparable and replicable set of results based entirely on the new dataset and on public-domain tools and data, whilst also experimentally motivating some novel feature types for the AA task, thus extending the work described in (Briscoe et al., 2010).

In the following sections we describe in more detail the dataset used for training and testing, the system developed, the evaluation methodology, as well as ablation experiments aimed at studying the contribution of different feature types to the AA task. We show experimentally that discriminative models with appropriate feature types can achieve performance close to the upper bound, as defined by the agreement between human examiners on the same test corpus.

2 Cambridge Learner Corpus

The Cambridge Learner Corpus[2] (CLC), developed as a collaborative project between Cambridge University Press and Cambridge Assessment, is a large collection of texts produced by English language learners from around the world, sitting Cambridge Assessment's English as a Second or Other Language (ESOL) examinations[3].

For the purpose of this work, we extracted scripts produced by learners taking the First Certificate in English (FCE) exam, which assesses English at an upper-intermediate level. The scripts, which are anonymised, are annotated using XML and linked to meta-data about the question prompts, the candidate's grades, native language and age. The FCE writing component consists of two tasks asking learners to write either a letter, a report, an article, a composition or a short story, between 200 and 400 words. Answers to each of these tasks are annotated with marks (in the range 1–40), which have been fitted to a RASCH model (Fischer and Molenaar, 1995) to correct for inter-examiner inconsistency and comparability. In addition, an overall mark is assigned to both tasks, which is the one we use in our experiments.

Each script has been also manually tagged with information about the linguistic errors committed,

[1] http://www.ilexir.com/

[2] http://www.cup.cam.ac.uk/gb/elt/catalogue/subject/custom/item3646603/Cambridge-International-Corpus-Cambridge-Learner-Corpus/?site_locale=en_GB

[3] http://www.cambridgeesol.org/

using a taxonomy of approximately 80 error types (Nicholls, 2003). The following is an example error-coded sentence:

> *In the morning, you are* <NS type = "TV"> *waken|woken*</NS> *up by a singing puppy.*

In this sentence, TV denotes an incorrect tense of verb error, where *waken* can be corrected to *woken*.

Our data consists of 1141 scripts from the year 2000 for training written by 1141 distinct learners, and 97 scripts from the year 2001 for testing written by 97 distinct learners. The learners' ages follow a bimodal distribution with peaks at approximately 16–20 and 26–30 years of age.

The prompts eliciting the free text are provided with the dataset. However, in this paper we make no use of prompt information and do not make any attempt to check that the text answer is appropriate to the prompt. Our focus is on developing an accurate AA system for ESOL text that does not require prompt-specific or topic-specific training. There is no overlap between the prompts used in 2000 and in 2001. A typical prompt taken from the 2000 training dataset is shown below:

Your teacher has asked you to write a story for the school's English language magazine. The story must begin with the following words: "Unfortunately, Pat wasn't very good at keeping secrets".

3 Approach

We treat automated assessment of ESOL text (see Section 2) as a rank preference learning problem (see Section 1). In the experiments reported here we use Support Vector Machines (SVMs) (Vapnik, 1995) through the SVM$^{\text{light}}$ package (Joachims, 1999). Using the dataset described in Section 2, a number of linguistic features are automatically extracted and their contribution to overall performance is investigated.

3.1 Rank preference model

SVMs have been extensively used for learning classification, regression and ranking functions. In its basic form, a binary SVM classifier learns a linear threshold function that discriminates data points of two categories. By using a different loss function, the ε-insensitive loss function (Smola, 1996), SVMs can also perform regression. SVMs in regression mode estimate a function that outputs a real number based on the training data. In both cases, the model generalises by computing a hyperplane that has the largest (soft-)margin.

In rank preference SVMs, the goal is to learn a ranking function which outputs a score for each data point, from which a global ordering of the data is constructed. This procedure requires a set R consisting of training samples $\vec{x}_n$ and their target rankings r_n:

$$R = \{(\vec{x}_1, r_1), (\vec{x}_2, r_2), ..., (\vec{x}_n, r_n)\} \quad (1)$$

such that $\vec{x}_i \succ_R \vec{x}_j$ when $r_i < r_j$, where $1 \leq i, j \leq n$ and $i \neq j$.

A rank preference model is not trained directly on this set of data objects and their labels; rather a set of pair-wise difference vectors is created. The goal of a linear ranking model is to compute a weight vector $\vec{w}$ that maximises the number of correctly ranked pairs:

$$\forall(\vec{x}_i \succ_R \vec{x}_j) : \vec{w}(\vec{x_i} - \vec{x_j}) > 0 \quad (2)$$

This is equivalent to solving the following optimisation problem:

Minimise:

$$\frac{1}{2}\|\vec{w}\|^2 + C\sum \xi_{ij} \quad (3)$$

Subject to the constraints:

$$\forall(\vec{x}_i \succ_R \vec{x}_j) : \vec{w}(\vec{x_i} - \vec{x_j}) \geq 1 - \xi_{ij} \quad (4)$$

$$\xi_{ij} \geq 0 \quad (5)$$

The factor C allows a trade-off between the training error and the margin size, while ξ_{ij} are non-negative slack variables that measure the degree of misclassification.

The optimisation problem is equivalent to that for the classification model on pair-wise difference vectors. In this case, generalisation is achieved by maximising the differences between closely-ranked data pairs.

The principal advantage of applying rank preference learning to the AA task is that we explicitly

model the grade relationships between scripts and do not need to apply a further regression step to fit the classifier output to the scoring scheme. The results reported in this paper are obtained by learning a linear classification function.

3.2 Feature set

We parsed the training and test data (see Section 2) using the Robust Accurate Statistical Parsing (RASP) system with the standard tokenisation and sentence boundary detection modules (Briscoe et al., 2006) in order to broaden the space of candidate features suitable for the task. The features used in our experiments are mainly motivated by the fact that lexical and grammatical features should be highly discriminative for the AA task. Our full feature set is as follows:

i. Lexical ngrams
   (a) Word unigrams
   (b) Word bigrams

ii. Part-of-speech (PoS) ngrams
   (a) PoS unigrams
   (b) PoS bigrams
   (c) PoS trigrams

iii. Features representing syntax
   (a) Phrase structure (PS) rules
   (b) Grammatical relation (GR) distance measures

iv. Other features
   (a) Script length
   (b) Error-rate

Word unigrams and bigrams are lower-cased and used in their inflected forms. PoS unigrams, bigrams and trigrams are extracted using the RASP tagger, which uses the CLAWS[4] tagset. The most probable posterior tag per word is used to construct PoS ngram features, but we use the RASP parser's option to analyse words assigned multiple tags when the posterior probability of the highest ranked tag is less than 0.9, and the next n tags have probability greater than $\frac{1}{50}$ of it.

[4]http://ucrel.lancs.ac.uk/claws/

Based on the most likely parse for each identified sentence, we extract the rule names from the phrase structure (PS) tree. RASP's rule names are semi-automatically generated and encode detailed information about the grammatical constructions found (e.g. V1/modal_bse/+-, 'a VP consisting of a modal auxiliary head followed by an (optional) adverbial phrase, followed by a VP headed by a verb with base inflection'). Moreover, rule names explicitly represent information about peripheral or rare constructions (e.g. S/pp-ap_s-r, 'a S with preposed PP with adjectival complement, e.g. *for better or worse, he left*'), as well as about fragmentary and likely extra-grammatical sequences (e.g. T/txt-frag, 'a text unit consisting of 2 or more subanalyses that cannot be combined using any rule in the grammar'). Therefore, we believe that many (longer-distance) grammatical constructions and errors found in texts can be (implicitly) captured by this feature type.

In developing our AA system, a number of different grammatical complexity measures were extracted from parses, and their impact on the accuracy of the system was explored. For the experiments reported here, we use complexity measures representing the sum of the longest distance in word tokens between a head and dependent in a grammatical relation (GR) from the RASP GR output, calculated for each GR graph from the top 10 parses per sentence. In particular, we extract the mean and median values of these distances per sentence and use the maximum values per script. Intuitively, this feature captures information about the grammatical sophistication of the writer. However, it may also be confounded in cases where sentence boundaries are not identified through, for example, poor punctuation.

Although the CLC contains information about the linguistic errors committed (see Section 2), we try to extract an error-rate in a way that doesn't require manually tagged data. However, we also use an error-rate calculated from the CLC error tags to obtain an upper bound for the performance of an automated error estimator (true CLC error-rate).

In order to estimate the error-rate, we build a trigram language model (LM) using ukWaC (ukWaC LM) (Ferraresi et al., 2008), a large corpus of English containing more than 2 billion tokens. Next, we extend our language model with trigrams extracted from a subset of the texts contained in the

Features	Pearson's correlation	Spearman's correlation
word ngrams	0.601	0.598
+PoS ngrams	0.682	0.687
+script length	0.692	0.689
+PS rules	0.707	0.708
+complexity	0.714	0.712
Error-rate features		
+ukWaC LM	0.735	0.758
+CLC LM	0.741	0.773
+true CLC error-rate	0.751	0.789

Table 1: Correlation between the CLC scores and the AA system predicted values.

Ablated feature	Pearson's correlation	Spearman's correlation
none	0.741	0.773
word ngrams	0.713	0.762
PoS ngrams	0.724	0.737
script length	0.734	0.772
PS rules	0.712	0.731
complexity	0.738	0.760
ukWaC+CLC LM	0.714	0.712

Table 2: Ablation tests showing the correlation between the CLC and the AA system.

CLC (CLC LM). As the CLC contains texts produced by second language learners, we only extract frequently occurring trigrams from highly ranked scripts to avoid introducing erroneous ones to our language model. A word trigram in test data is counted as an error if it is not found in the language model. We compute presence/absence efficiently using a Bloom filter encoding of the language models (Bloom, 1970).

Feature instances of types i and ii are weighted using the *tf*idf* scheme and normalised by the L2 norm. Feature type iii is weighted using frequency counts, while iii and iv are scaled so that their final value has approximately the same order of magnitude as i and ii.

The script length is based on the number of words and is mainly added to balance the effect the length of a script has on other features. Finally, features whose overall frequency is lower than four are discarded from the model.

4 Evaluation

In order to evaluate our AA system, we use two correlation measures, Pearson's product-moment correlation coefficient and Spearman's rank correlation coefficient (hereafter Pearson's and Spearman's correlation respectively). Pearson's correlation determines the degree to which two linearly dependent variables are related. As Pearson's correlation is sensitive to the distribution of data and, due to outliers, its value can be misleading, we also report Spearman's correlation. The latter is a non-parametric robust measure of association which is sensitive only to the ordinal arrangement of values. As our data contains some tied values, we calculate Spearman's correlation by using Pearson's correlation on the ranks.

Table 1 presents the Pearson's and Spearman's correlation between the CLC scores and the AA system predicted values, when incrementally adding to the model the feature types described in Section 3.2. Each feature type improves the model's performance. Extending our language model with frequent trigrams extracted from the CLC improves Pearson's and Spearman's correlation by 0.006 and 0.015 respectively. The addition of the error-rate obtained from the manually annotated CLC error tags on top of all the features further improves performance by 0.01 and 0.016. An evaluation of our best error detection method shows a Pearson correlation of 0.611 between the estimated and the true CLC error counts. This suggests that there is room for improvement in the language models we developed to estimate the error-rate. In the experiments reported hereafter, we use the ukWaC+CLC LM to calculate the error-rate.

In order to assess the independent as opposed to the order-dependent additive contribution of each feature type to the overall performance of the system, we run a number of ablation tests. An ablation test consists of removing one feature of the system at a time and re-evaluating the model on the test set. Table 2 presents Pearson's and Spearman's correlation between the CLC and our system, when removing one feature at a time. All features have a positive effect on performance, while the error-rate has a big impact, as its absence is responsible for a 0.061 decrease of Spearman's correlation. In addition, the

Model	Pearson's correlation	Spearman's correlation
Regression	0.697	0.706
Rank preference	0.741	0.773

Table 3: Comparison between regression and rank preference model.

removal of either the word ngrams, the PS rules, or the error-rate estimate contributes to a large decrease in Pearson's correlation.

In order to test the significance of the improved correlations, we ran one-tailed t-tests with $a = 0.05$ for the difference between dependent correlations (Williams, 1959; Steiger, 1980). The results showed that PoS ngrams, PS rules, the complexity measures, and the estimated error-rate contribute significantly to the improvement of Spearman's correlation, while PS rules also contribute significantly to the improvement of Pearson's correlation.

One of the main approaches adopted by previous systems involves the identification of features that measure writing skill, and then the application of linear or stepwise regression to find optimal feature weights so that the correlation with manually assigned scores is maximised. We trained a SVM regression model with our full set of feature types and compared it to the SVM rank preference model. The results are given in Table 3. The rank preference model improves Pearson's and Spearman's correlation by 0.044 and 0.067 respectively, and these differences are significant, suggesting that rank preference is a more appropriate model for the AA task.

Four senior and experienced ESOL examiners remarked the 97 FCE test scripts drawn from 2001 exams, using the marking scheme from that year (see Section 2). In order to obtain a ceiling for the performance of our system, we calculate the average correlation between the CLC and the examiners' scores, and find an upper bound of **0.796** and **0.792** Pearson's and Spearman's correlation respectively.

In order to evaluate the overall performance of our system, we calculate its correlation with the four senior examiners in addition to the RASCH-adjusted CLC scores. Tables 4 and 5 present the results obtained.

The average correlation of the AA system with the CLC and the examiner scores shows that it is close

	CLC	E1	E2	E3	E4	AA
CLC	-	0.820	0.787	0.767	0.810	0.741
E1	0.820	-	0.851	0.845	0.878	0.721
E2	0.787	0.851	-	0.775	0.788	0.730
E3	0.767	0.845	0.775	-	0.779	0.747
E4	0.810	0.878	0.788	0.779	-	0.679
AA	0.741	0.721	0.730	0.747	0.679	-
Avg	0.785	0.823	0.786	0.782	0.786	0.723

Table 4: Pearson's correlation of the AA system predicted values with the CLC and the examiners' scores, where E1 refers to the first examiner, E2 to the second etc.

	CLC	E1	E2	E3	E4	AA
CLC	-	0.801	0.799	0.788	0.782	0.773
E1	0.801	-	0.809	0.806	0.850	0.675
E2	0.799	0.809	-	0.744	0.787	0.724
E3	0.788	0.806	0.744	-	0.794	0.738
E4	0.782	0.850	0.787	0.794	-	0.697
AA	0.773	0.675	0.724	0.738	0.697	-
Avg	0.788	0.788	0.772	0.774	0.782	0.721

Table 5: Spearman's correlation of the AA system predicted values with the CLC and the examiners' scores, where E1 refers to the first examiner, E2 to the second etc.

to the upper bound for the task. Human–machine agreement is comparable to that of human–human agreement, with the exception of Pearson's correlation with examiner E4 and Spearman's correlation with examiners E1 and E4, where the discrepancies are higher. It is likely that a larger training set and/or more consistent grading of the existing training data would help to close this gap. However, our system is not measuring some properties of the scripts, such as discourse cohesion or relevance to the prompt eliciting the text, that examiners will take into account.

5 Validity tests

The practical utility of an AA system will depend strongly on its robustness to subversion by writers who understand something of its workings and attempt to exploit this to maximise their scores (independently of their underlying ability). Surprisingly, there is very little published data on the robustness of existing systems. However, Powers et al. (2002) invited writing experts to trick the scoring

capabilities of an earlier version of e-Rater (Burstein et al., 1998). e-Rater (see Section 6 for more details) assigns a score to a text based on linguistic feature types extracted using relatively domain-specific techniques. Participants were given a description of these techniques as well as of the cue words that the system uses. The results showed that it was easier to fool the system into assigning higher than lower scores.

Our goal here is to determine the extent to which knowledge of the feature types deployed poses a threat to the validity of our system, where certain text generation strategies may give rise to large positive discrepancies. As mentioned in Section 2, the marking criteria for FCE scripts are primarily based on the accurate use of a range of different grammatical constructions relevant to specific communicative goals, but our system assesses this indirectly.

We extracted 6 high-scoring FCE scripts from the CLC that do not overlap with our training and test data. Based on the features used by our system and without bias towards any modification, we modified each script in one of the following ways:

i. Randomly order:
   (a) word unigrams within a sentence
   (b) word bigrams within a sentence
   (c) word trigrams within a sentence
   (d) sentences within a script

ii. Swap words that have the same PoS within a sentence

Although the above modifications do not exhaust the potential challenges a deployed AA system might face, they represent a threat to the validity of our system since we are using a highly related feature set. In total, we create 30 such 'outlier' texts, which were given to an ESOL examiner for marking. Using the 'outlier' scripts as well as their original/unmodified versions, we ran our system on each modification separately and calculated the correlation between the predicted values and the examiner's scores. Table 6 presents the results.

The predicted values of the system have a high correlation with the examiner's scores when tested on 'outlier' texts of modification types i(a), i(b) and i(c). However, as i(c) has a lower correlation compared to i(a) and i(b), it is likely that a random ordering of ngrams with $N > 3$ will further decrease performance. A modification of type ii, where words with the same PoS within a sentence are swapped, results in a Pearson and Spearman correlation of 0.634 and 0.761 respectively.

Modification	**Pearson's correlation**	**Spearman's correlation**
i(a)	0.960	0.912
i(b)	0.938	0.914
i(c)	0.801	0.867
i(d)	0.08	0.163
ii	0.634	0.761

Table 6: Correlation between the predicted values and the examiner's scores on 'outlier' texts.

Analysis of the results showed that our system predicted higher scores than the ones assigned by the examiner. This can be explained by the fact that texts produced using modification type ii contain a small portion of correct sentences. However, the marking criteria are based on the overall writing quality. The final case, where correct sentences are randomly ordered, receives the lowest correlation. As our system is not measuring discourse cohesion, discrepancies are much higher; the system's predicted scores are high whilst the ones assigned by the examiner are very low. However, for a writer to be able to generate text of this type already requires significant linguistic competence, whilst a number of generic methods for assessing text and/or discourse cohesion have been developed and could be deployed in an extended version of our system.

It is also likely that highly creative 'outlier' essays may give rise to large negative discrepancies. Recent comments in the British media have focussed on this issue, reporting that, for example, one deployed essay marking system assigned Winston Churchill's speech 'We Shall Fight on the Beaches' a low score because of excessive repetition[5]. Our model predicted a high passing mark for this text, but not the highest one possible, that some journalists clearly feel it deserves.

[5] http://news.bbc.co.uk/1/hi/education/8356572.stm

6 Previous work

In this section we briefly discuss a number of the more influential and/or better described approaches. Pérez-Marín et al. (2009), Williamson (2009), Dikli (2006) and Valenti et al. (2003) provide a more detailed overview of existing AA systems.

Project Essay Grade (PEG) (Page, 2003), one of the earliest systems, uses a number of manually-identified mostly shallow textual features, which are considered to be proxies for intrinsic qualities of writing competence. Linear regression is used to assign optimal feature weights that maximise the correlation with the examiner's scores. The main issue with this system is that features such as word length and script length are easy to manipulate independently of genuine writing ability, potentially undermining the validity of the system.

In e-Rater (Attali and Burstein, 2006), texts are represented using vectors of weighted features. Each feature corresponds to a different property of texts, such as an aspect of grammar, style, discourse and topic similarity. Additional features, representing stereotypical grammatical errors for example, are extracted using manually-coded task-specific detectors based, in part, on typical marking criteria. An unmarked text is scored based on the cosine similarity between its weighted vector and the ones in the training set. Feature weights and/or scores can be fitted to a marking scheme by stepwise or linear regression. Unlike our approach, e-Rater models discourse structure, semantic coherence and relevance to the prompt. However, the system contains manually developed task-specific components and requires retraining or tuning for each new prompt and assessment task.

Intelligent Essay Assessor (IEA) (Landauer et al., 2003) uses Latent Semantic Analysis (LSA) (Landauer and Foltz, 1998) to compute the semantic similarity between texts, at a specific grade point, and a test text. In LSA, text is represented by a matrix, where rows correspond to words and columns to context (texts). Singular Value Decomposition (SVD) is used to obtain a reduced dimension matrix clustering words and contexts. The system is trained on topic and/or prompt specific texts while test texts are assigned a score based on the ones in the training set that are most similar. The overall score, which is calculated using regression techniques, is based on the content score as well as on other properties of texts, such as style, grammar, and so forth, though the methods used to assess these are not described in any detail in published work. Again, the system requires retraining or tuning for new prompts and assessment tasks.

Lonsdale and Strong-Krause (2003) use a modified syntactic parser to analyse and score texts. Their method is based on a modified version of the Link Grammar parser (Sleator and Templerley, 1995) where the overall score of a text is calculated as the average of the scores assigned to each sentence. Sentences are scored on a five-point scale based on the parser's cost vector, which roughly measures the complexity and deviation of a sentence from the parser's grammatical model. This approach bears some similarities to our use of grammatical complexity and extragrammaticality features, but grammatical features represent only one component of our overall system, and of the task.

The Bayesian Essay Test Scoring sYstem (BETSY) (Rudner and Liang, 2002) uses multinomial or Bernoulli Naive Bayes models to classify texts into different classes (e.g. pass/fail, grades A–F) based on content and style features such as word unigrams and bigrams, sentence length, number of verbs, noun–verb pairs etc. Classification is based on the conditional probability of a class given a set of features, which is calculated using the assumption that each feature is independent of the other. This system shows that treating AA as a text classification problem is viable, but the feature types are all fairly shallow, and the approach doesn't make efficient use of the training data as a separate classifier is trained for each grade point.

Recently, Chen et al. (2010) has proposed an unsupervised approach to AA of texts addressing the same topic, based on a voting algorithm. Texts are clustered according to their grade and given an initial Z-score. A model is trained where the initial score of a text changes iteratively based on its similarity with the rest of the texts as well as their Z-scores. The approach might be better described as weakly supervised as the distribution of text grades in the training data is used to fit the final Z-scores to grades. The system uses a bag-of-words representation of text, so would be easy to subvert. Never-

theless, exploration of the trade-offs between degree of supervision required in training and grading accuracy is an important area for future research.

7 Conclusions and future work

Though many of the systems described in Section 6 have been shown to correlate well with examiners' marks on test data in many experimental contexts, no cross-system comparisons are available because of the lack of a shared training and test dataset. Furthermore, none of the published work of which we are aware has systematically compared the contribution of different feature types to the AA task, and only one (Powers et al., 2002) assesses the ease with which the system can be subverted given some knowledge of the features deployed.

We have shown experimentally how rank preference models can be effectively deployed for automated assessment of ESOL free-text answers. Based on a range of feature types automatically extracted using generic text processing techniques, our system achieves performance close to the upper bound for the task. Ablation tests highlight the contribution of each feature type to the overall performance, while significance of the resulting improvements in correlation with human scores has been calculated. A comparison between regression and rank preference models further supports our approach. Preliminary experiments based on a set of 'outlier' texts have shown the types of texts for which the system's scoring capability can be undermined.

We plan to experiment with better error detection techniques, since the overall error-rate of a script is one of the most discriminant features. Briscoe et al. (2010) describe an approach to automatic off-prompt detection which does not require retraining for each new question prompt and which we plan to integrate with our system. It is clear from the 'outlier' experiments reported here that our system would benefit from features assessing discourse coherence, and to a lesser extent from features assessing semantic (selectional) coherence over longer bounds than those captured by ngrams. The addition of an incoherence metric to the feature set of an AA system has been shown to improve performance significantly (Miltsakaki and Kukich, 2000; Miltsakaki and Kukich, 2004).

Finally, we hope that the release of the training and test dataset described here will facilitate further research on the AA task for ESOL free text and, in particular, precise comparison of different systems, feature types, and grade fitting methods.

Acknowledgements

We would like to thank Cambridge ESOL, a division of Cambridge Assessment, for permission to use and distribute the examination scripts. We are also grateful to Cambridge Assessment for arranging for the test scripts to be remarked by four of their senior examiners. Finally, we would like to thank Marek Rei, Øistein Andersen and the anonymous reviewers for their useful comments.

References

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with e-rater v.2. *Journal of Technology, Learning, and Assessment*, 4(3):1–30.

Burton H. Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July.

E.J. Briscoe, J. Carroll, and R Watson. 2006. The second release of the RASP system. In *ACL-Coling'06 Interactive Presentation Session*, pages 77–80, Sydney, Australia.

E.J. Briscoe, B. Medlock, and Ø. Andersen. 2010. *Automated Assessment of ESOL Free Text Examinations*. Cambridge University, Computer Laboratory, TR-790.

Jill Burstein, Karen Kukich, Susanne Wolff, Chi Lu, Martin Chodorow, Lisa Braden-Harder, and Mary Dee Harris. 1998. Automated scoring using a hybrid feature identification technique. *Proceedings of the 36th annual meeting on Association for Computational Linguistics*, pages 206–210.

YY Chen, CL Liu, TH Chang, and CH Lee. 2010. An Unsupervised Automated Essay Scoring System. *IEEE Intelligent Systems*, pages 61–67.

Semire Dikli. 2006. An overview of automated scoring of essays. *Journal of Technology, Learning, and Assessment*, 5(1).

S. Elliot. 2003. IntelliMetric: From here to validity. In M.D. Shermis and J.C. Burstein, editors, *Automated essay scoring: A cross-disciplinary perspective*, pages 71–86.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukWaC, a very large web-derived corpus of English.

In S. Evert, A. Kilgarriff, and S. Sharoff, editors, *Proceedings of the 4th Web as Corpus Workshop (WAC-4)*.

G.H. Fischer and I.W. Molenaar. 1995. *Rasch models: Foundations, recent developments, and applications*. Springer.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, pages 137–142. Springer.

Thorsten Joachims. 1999. Making large scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142. ACM.

T.K. Landauer and P.W. Foltz. 1998. An introduction to latent semantic analysis. *Discourse processes*, pages 259–284.

T.K. Landauer, D. Laham, and P.W. Foltz. 2003. Automated scoring and annotation of essays with the Intelligent Essay Assessor. In M.D. Shermis and J.C. Burstein, editors, *Automated essay scoring: A cross-disciplinary perspective*, pages 87–112.

Deryle Lonsdale and D. Strong-Krause. 2003. Automated rating of ESL essays. In *Proceedings of the HLT-NAACL 2003 Workshop: Building Educational Applications Using Natural Language Processing*.

Eleni Miltsakaki and Karen Kukich. 2000. Automated evaluation of coherence in student essays. In *Proceedings of LREC 2000*.

Eleni Miltsakaki and Karen Kukich. 2004. Evaluation of text coherence for electronic essay scoring systems. *Natural Language Engineering*, 10(01):25–55, March.

D. Nicholls. 2003. The Cambridge Learner Corpus: Error coding and analysis for lexicography and ELT. In *Proceedings of the Corpus Linguistics 2003 conference*, pages 572–581.

E.B. Page. 2003. Project essay grade: PEG. In M.D. Shermis and J.C. Burstein, editors, *Automated essay scoring: A cross-disciplinary perspective*, pages 43–54.

D. Pérez-Marín, Ismael Pascual-Nieto, and P. Rodríguez. 2009. Computer-assisted assessment of free-text answers. *The Knowledge Engineering Review*, 24(04):353–374, December.

D.E. Powers, J.C. Burstein, M. Chodorow, M.E. Fowles, and K. Kukich. 2002. Stumping e-rater: challenging the validity of automated essay scoring. *Computers in Human Behavior*, 18(2):103–134.

L.M. Rudner and Tahung Liang. 2002. Automated essay scoring using Bayes' theorem. *The Journal of Technology, Learning and Assessment*, 1(2):3–21.

L.M. Rudner, Veronica Garcia, and Catherine Welch. 2006. An Evaluation of the IntelliMetric Essay Scoring System. *Journal of Technology, Learning, and Assessment*, 4(4):1–21.

D.D.K. Sleator and D. Templerley. 1995. Parsing English with a link grammar. *Proceedings of the 3rd International Workshop on Parsing Technologies, ACL.*

AJ Smola. 1996. Regression estimation with support vector learning machines. *Master's thesis, Technische Universität Munchen*.

J.H. Steiger. 1980. Tests for comparing elements of a correlation matrix. *Psychological Bulletin*, 87(2):245–251.

Salvatore Valenti, Francesca Neri, and Alessandro Cucchiarelli. 2003. An overview of current research on automated essay grading. *Journal of Information Technology Education*, 2:3–118.

Vladimir N. Vapnik. 1995. *The nature of statistical learning theory*. Springer.

E. J. Williams. 1959. The Comparison of Regression Variables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 21(2):396–399.

DM Williamson. 2009. A Framework for Implementing Automated Scoring. In *Annual Meeting of the American Educational Research Association and the National Council on Measurement in Education*, San Diego, CA.

Collecting Highly Parallel Data for Paraphrase Evaluation

David L. Chen
Department of Computer Science
The University of Texas at Austin
Austin, TX 78712, USA
dlcc@cs.utexas.edu

William B. Dolan
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
billdol@microsoft.com

Abstract

A lack of standard datasets and evaluation metrics has prevented the field of *paraphrasing* from making the kind of rapid progress enjoyed by the machine translation community over the last 15 years. We address both problems by presenting a novel data collection framework that produces highly parallel text data relatively inexpensively and on a large scale. The highly parallel nature of this data allows us to use simple n-gram comparisons to measure both the semantic adequacy and lexical dissimilarity of paraphrase candidates. In addition to being simple and efficient to compute, experiments show that these metrics correlate highly with human judgments.

1 Introduction

Machine paraphrasing has many applications for natural language processing tasks, including machine translation (MT), MT evaluation, summary evaluation, question answering, and natural language generation. However, a lack of standard datasets and automatic evaluation metrics has impeded progress in the field. Without these resources, researchers have resorted to developing their own small, ad hoc datasets (Barzilay and McKeown, 2001; Shinyama et al., 2002; Barzilay and Lee, 2003; Quirk et al., 2004; Dolan et al., 2004), and have often relied on human judgments to evaluate their results (Barzilay and McKeown, 2001; Ibrahim et al., 2003; Bannard and Callison-Burch, 2005). Consequently, it is difficult to compare different systems and assess the progress of the field as a whole.

Despite the similarities between paraphrasing and translation, several major differences have prevented researchers from simply following standards that have been established for machine translation. Professional translators produce large volumes of bilingual data according to a more or less consistent specification, indirectly fueling work on machine translation algorithms. In contrast, there are no "professional paraphrasers", with the result that there are no readily available large corpora and no consistent standards for what constitutes a high-quality paraphrase. In addition to the lack of standard datasets for training and testing, there are also no standard metrics like BLEU (Papineni et al., 2002) for evaluating paraphrase systems. Paraphrase evaluation is inherently difficult because the range of potential paraphrases for a given input is both large and unpredictable; in addition to being meaning-preserving, an ideal paraphrase must also diverge as sharply as possible in form from the original while still sounding natural and fluent.

Our work introduces two novel contributions which combine to address the challenges posed by paraphrase evaluation. First, we describe a framework for easily and inexpensively crowdsourcing arbitrarily large training and test sets of independent, redundant linguistic descriptions of the same semantic content. Second, we define a new evaluation metric, PINC (Paraphrase In N-gram Changes), that relies on simple BLEU-like n-gram comparisons to measure the degree of novelty of automatically generated paraphrases. We believe that this metric, along with the sentence-level paraphrases provided by our data collection approach, will make it possi-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 190–200,
Portland, Oregon, June 19-24, 2011.

ble for researchers working on paraphrasing to compare system performance and exploit the kind of automated, rapid training-test cycle that has driven work on Statistical Machine Translation.

In addition to describing a mechanism for collecting large-scale sentence-level paraphrases, we are also making available to the research community 85K parallel English sentences as part of the Microsoft Research Video Description Corpus [1].

The rest of the paper is organized as follows. We first review relevant work in Section 2. Section 3 then describes our data collection framework and the resulting data. Section 4 discusses automatic evaluations of paraphrases and introduces the novel metric PINC. Section 5 presents experimental results establishing a correlation between our automatic metric and human judgments. Sections 6 and 7 discuss possible directions for future research and conclude.

2 Related Work

Since paraphrase data are not readily available, various methods have been used to extract parallel text from other sources. One popular approach exploits multiple translations of the same data (Barzilay and McKeown, 2001; Pang et al., 2003). Examples of this kind of data include the Multiple-Translation Chinese (MTC) Corpus [2] which consists of Chinese news stories translated into English by 11 translation agencies, and literary works with multiple translations into English (e.g. Flaubert's *Madame Bovary*.) Another method for collecting monolingual paraphrase data involves aligning semantically parallel sentences from different news articles describing the same event (Shinyama et al., 2002; Barzilay and Lee, 2003; Dolan et al., 2004).

While utilizing multiple translations of literary work or multiple news stories of the same event can yield significant numbers of parallel sentences, this data tend to be noisy, and reliably identifying good paraphrases among all possible sentence pairs remains an open problem. On the other hand, multiple translations on the sentence level such as the MTC Corpus provide good, natural paraphrases, but relatively little data of this type exists. Finally, some approaches avoid the need for monolingual paraphrase data altogether by using a second language as the pivot language (Bannard and Callison-Burch, 2005; Callison-Burch, 2008; Kok and Brockett, 2010). Phrases that are aligned to the same phrase in the pivot language are treated as potential paraphrases. One limitation of this approach is that only words and phrases are identified, not whole sentences.

While most work on evaluating paraphrase systems has relied on human judges (Barzilay and McKeown, 2001; Ibrahim et al., 2003; Bannard and Callison-Burch, 2005) or indirect, task-based methods (Lin and Pantel, 2001; Callison-Burch et al., 2006), there have also been a few attempts at creating automatic metrics that can be more easily replicated and used to compare different systems. ParaMetric (Callison-Burch et al., 2008) compares the paraphrases discovered by an automatic system with ones annotated by humans, measuring precision and recall. This approach requires additional human annotations to identify the paraphrases within parallel texts (Cohn et al., 2008) and does not evaluate the systems at the sentence level. The more recently proposed metric PEM (Paraphrase Evaluation Metric) (Liu et al., 2010) produces a single score that captures the semantic adequacy, fluency, and lexical dissimilarity of candidate paraphrases, relying on bilingual data to learn semantic equivalences without using n-gram similarity between candidate and reference sentences. In addition, the metric was shown to correlate well with human judgments. However, a significant drawback of this approach is that PEM requires substantial in-domain bilingual data to train the semantic adequacy evaluator, as well as sample human judgments to train the overall metric.

We designed our data collection framework for use on crowdsourcing platforms such as Amazon's Mechanical Turk. Crowdsourcing can allow inexpensive and rapid data collection for various NLP tasks (Ambati and Vogel, 2010; Bloodgood and Callison-Burch, 2010a; Bloodgood and Callison-Burch, 2010b; Irvine and Klementiev, 2010), including human evaluations of NLP systems (Callison-Burch, 2009; Denkowski and Lavie, 2010; Zaidan and Callison-Burch, 2009). Of particular relevance are the paraphrasing work by Buzek et al. (2010)

[1] Available for download at `http://research.microsoft.com/en-us/downloads/38cf15fd-b8df-477e-a4e4-a4680caa75af/`

[2] Linguistic Data Consortium (LDC) Catalog Number LDC2002T01, ISBN 1-58563-217-1.

and Denkowski et al. (2010). Buzek et al. automatically identified problem regions in a translation task and had workers attempt to paraphrase them, while Denkowski et al. asked workers to assess the validity of automatically extracted paraphrases. Our work is distinct from these earlier efforts both in terms of the task – attempting to collect linguistic descriptions using a visual stimulus – and the dramatically larger scale of the data collected.

3 Data Collection

Since our goal was to collect large numbers of paraphrases quickly and inexpensively using a crowd, our framework was designed to make the tasks short, simple, easy, accessible and somewhat fun. For each task, we asked the annotators to watch a very short video clip (usually less than 10 seconds long) and describe in one sentence the main action or event that occurred in the video clip

We deployed the task on Amazon's Mechanical Turk, with video segments selected from YouTube. A screenshot of our annotation task is shown in Figure 1. On average, annotators completed each task within 80 seconds, including the time required to watch the video. Experienced annotators were even faster, completing the task in only 20 to 25 seconds.

One interesting aspect of this framework is that each annotator approaches the task from a linguistically independent perspective, unbiased by the lexical or word order choices in a pre-existing description. The data thus has some similarities to parallel news descriptions of the same event, while avoiding much of the noise inherent in news. It is also similar in spirit to the 'Pear Stories' film used by Chafe (1997). Crucially, our approach allows us to gather arbitrarily many of these independent descriptions for each video, capturing nearly-exhaustive coverage of how native speakers are likely to summarize a small action. It might be possible to achieve similar effects using images or panels of images as the stimulus (von Ahn and Dabbish, 2004; Fei-Fei et al., 2007; Rashtchian et al., 2010), but we believed that videos would be more engaging and less ambiguous in their focus. In addition, videos have been shown to be more effective in prompting descriptions of motion and contact verbs, as well as verbs that are generally not imageable (Ma and Cook, 2009).

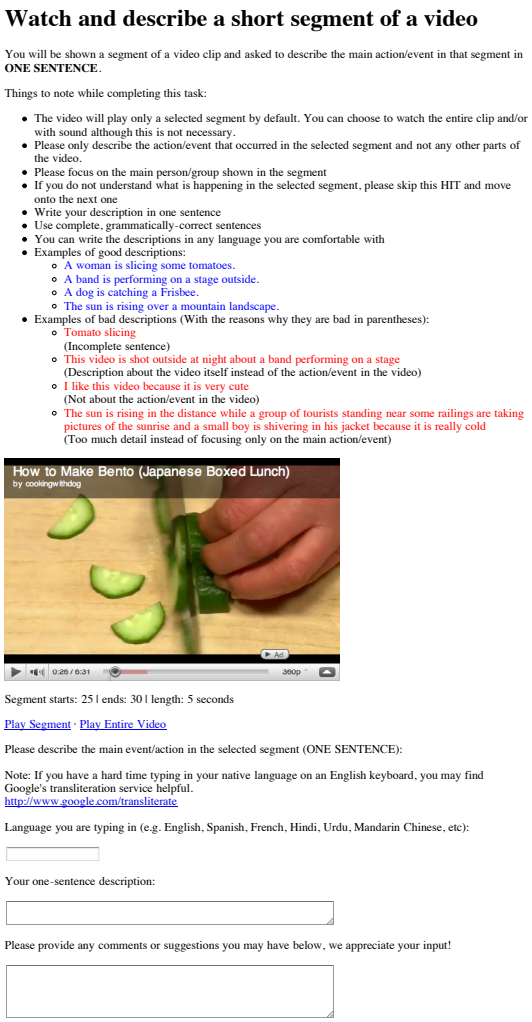


Figure 1: A screenshot of our annotation task as it was deployed on Mechanical Turk.

3.1 Quality Control

One of the main problems with collecting data using a crowd is quality control. While the cost is very low compared to traditional annotation methods, workers recruited over the Internet are often unqualified for the tasks or are incentivized to cheat in order to maximize their rewards.

To encourage native and fluent contributions, we asked annotators to write the descriptions in the language of their choice. The result was a significant amount of translation data, unique in its multilingual parallelism. While included in our data release, we leave aside a full discussion of this multilingual data for future work.

To ensure the quality of the annotations being produced, we used a two-tiered payment system. The idea was to reward workers who had shown the ability to write quality descriptions and the willingness to work on our tasks consistently. While everyone had access to the Tier-1 tasks, only workers who had been manually qualified could work on the Tier-2 tasks. The tasks were identical in the two tiers but each Tier-1 task only paid 1 cent while each Tier-2 task paid 5 cents, giving the workers a strong incentive to earn the qualification.

The qualification process was done manually by the authors. We periodically evaluated the workers who had submitted the most Tier-1 tasks (usually on the order of few hundred submissions) and granted them access to the Tier-2 tasks if they had performed well. We assessed their work mainly on the grammaticality and spelling accuracy of the submitted descriptions. Since we had hundreds of submissions to base our decisions on, it was fairly easy to identify the cheaters and people with poor English skills [3]. Workers who were rejected during this process were still allowed to work on the Tier-1 tasks.

While this approach requires significantly more manual effort initially than other approaches such as using a qualification test or automatic post-annotation filtering, it creates a much higher quality workforce. Moreover, the initial effort is amortized over time as these quality workers are retained over the entire duration of the data collection. Many of them annotated all the available videos we had.

3.2 Video Collection

To find suitable videos to annotate, we deployed a separate task. Workers were asked to submit short (generally 4-10 seconds) video segments depicting single, unambiguous events by specifying links to YouTube videos, along with the start and end times. We again used a tiered payment system to reward and retain workers who performed well.

Since the scope of this data collection effort extended beyond gathering English data alone, we tried to collect videos that could be understood regardless of the annotator's linguistic or cultural background. In order to avoid biasing lexical choices in the descriptions, we muted the audio and excluded videos that contained either subtitles or overlaid text. Finally, we manually filtered the submitted videos to ensure that each met our criteria and was free of inappropriate content.

- Someone is coating a pork chop in a glass bowl of flour.
- A person breads a pork chop.
- Someone is breading a piece of meat with a white powdery substance.
- A chef seasons a slice of meat.
- Someone is putting flour on a piece of meat.
- A woman is adding flour to meat.
- A woman is coating a piece of pork with breadcrumbs.
- A man dredges meat in bread crumbs.
- A person breads a piece of meat.
- A woman is breading some meat.
- Someone is breading meat.
- A woman coats a meat cutlet in a dish.
- A woman is coating a pork loin in bread crumbs.
- The laldy coated the meat in bread crumbs.
- The woman is breading pork chop.
- A woman adds a mixture to some meat.
- The lady put the batter on the meat.

Figure 2: Examples of English descriptions collected for a particular video segment.

3.3 Data

We deployed our data collection framework on Mechanical Turk over a two-month period from July to September in 2010, collecting 2,089 video segments and 85,550 English descriptions. The rate of data collection accelerated as we built up our workforce, topping 10K descriptions a day when we ended our data collection. Of the descriptions, 33,855 were from Tier-2 tasks, meaning they were provided by workers who had been manually identified as good performers. Examples of some of the descriptions collected are shown in Figure 2.

Overall, 688 workers submitted at least one English description. Of these workers, 113 submitted at least 100 descriptions and 51 submitted at least 500. The largest number of descriptions submitted by a single worker was 3496 [4]. Out of the 688 workers, 50 were granted access to the Tier-2 tasks. The

[3] Everyone who submitted descriptions in a foreign language was granted access to the Tier-2 tasks. This was done to encourage more submissions in different languages and also because we could not verify the quality of those descriptions other than using online translation services (and some of the languages were not available to be translated).

[4] This number exceeds the total number of videos because the worker completed both Tier-1 and Tier-2 tasks for the same videos

	Tier 1	Tier 2
pay	$0.01	$0.05
# workers (English)	683	50
# workers (total)	835	94
# submitted (English)	51510	33829
# submitted (total)	68578	55682
# accepted (English)	51052	33825
# accepted (total)	67968	55658

Table 1: Statistics for the two video description tasks

success of our data collection effort was in part due to our ability to retain these good workers, building a reliable and efficient workforce. Table 1 shows some statistics for the Tier-1 and Tier-2 tasks [5]. Overall, we spent under $5,000 including Amazon's service fees, some pilot experiments and surveys.

On average, 41 descriptions were produced for each video, with at least 27 for over 95% of the videos. Even limiting the set to descriptions produced from the Tier-2 tasks, there are still 16 descriptions on average for each video, with at least 12 descriptions for over 95% of the videos. For most clusters, then, we have a dozen or more high-quality parallel descriptions that can be paired with one another to create monolingual parallel training data.

4 Paraphrase Evaluation Metrics

One of the limitations to the development of machine paraphrasing is the lack of standard metrics like BLEU, which has played a crucial role in driving progress in MT. Part of the issue is that a good paraphrase has the additional constraint that it should be lexically dissimilar to the source sentence while preserving the meaning. These can become competing goals when using n-gram overlaps to establish semantic equivalence. Thus, researchers have been unable to rely on BLEU or some derivative: the optimal paraphrasing engine under these terms would be one that simply returns the input.

To combat such problems, Liu et al. (2010) have proposed PEM, which uses a second language as pivot to establish semantic equivalence. Thus, no n-gram overlaps are required to determine the semantic adequacy of the paraphrase candidates. PEM also separately measures lexical dissimilarity and fluency. Finally, all three scores are combined using a support vector machine (SVM) trained on human ratings of paraphrase pairs. While PEM was shown to correlate well with human judgments, it has some limitations. It only models paraphrasing at the phrase level and not at the sentence level. Further, while it does not need reference sentences for the evaluation dataset, PEM does require suitable bilingual data to train the metric. The result is that training a successful PEM becomes almost as challenging as the original paraphrasing problem, since paraphrases need to be learned from bilingual data.

The highly parallel nature of our data suggests a simpler solution to this problem. To measure semantic equivalence, we simply use BLEU with multiple references. The large number of reference paraphrases capture a wide space of sentences with equivalent meanings. While the set of reference sentences can of course never be exhaustive, our data collection method provides a natural distribution of common phrases that might be used to describe an action or event. A tight cluster with many similar parallel descriptions suggests there are only few common ways to express that concept.

In addition to measuring semantic adequacy and fluency using BLEU, we also need to measure lexical dissimilarity with the source sentence. We introduce a new scoring metric PINC that measures how many n-grams differ between the two sentences. In essence, it is the inverse of BLEU since we want to minimize the number of n-gram overlaps between the two sentences. Specifically, for source sentence s and candidate sentence c:

$$PINC(s,c) = \frac{1}{N}\sum_{n=1}^{N} 1 - \frac{|\text{ n-gram}_s \cap \text{n-gram}_c|}{|\text{ n-gram}_c|}$$

where N is the maximum n-gram considered and n-gram$_s$ and n-gram$_c$ are the lists of n-grams in the source and candidate sentences, respectively. We use $N = 4$ in our evaluations.

The PINC score computes the percentage of n-grams that appear in the candidate sentence but not in the source sentence. This score is similar to the Jaccard distance, except that it excludes n-grams that only appear in the source sentence and not in the candidate sentence. In other words, it rewards candi-

[5] The numbers for the English data are slightly underestimated since the workers sometimes incorrectly filled out the form when reporting what language they were using.

dates for introducing new n-grams but not for omitting n-grams from the original sentence. The results for each n are averaged arithmetically. PINC evaluates single sentences instead of entire documents because we can reliably measure lexical dissimilarity at the sentence level. Also notice that we do not put additional constraints on sentence length: while extremely short and extremely long sentences are likely to score high on PINC, they still must maintain semantic adequacy as measured by BLEU.

We use BLEU and PINC together as a 2-dimensional scoring metric. A good paraphrase, according to our evaluation metric, has few n-gram overlaps with the source sentence but many n-gram overlaps with the reference sentences. This is consistent with our requirement that a good paraphrase should be lexically dissimilar from the source sentence while preserving its semantics.

Unlike Liu et al. (2010), we treat these two criteria separately, since different applications might have different preferences for each. For example, a paraphrase suggestion tool for a word processing software might be more concerned with semantic adequacy, since presenting a paraphrase that does not preserve the meaning would likely result in a negative user experience. On the other hand, a query expansion algorithm might be less concerned with preserving the precise meaning so long as additional relevant terms are added to improve search recall.

5 Experiments

To verify the usefulness of our paraphrase corpus and the BLEU/PINC metric, we built and evaluated several paraphrase systems and compared the automatic scores to human ratings of the generated paraphrases. We also investigated the pros and cons of collecting paraphrases using video annotation rather than directly eliciting them.

5.1 Building paraphrase models

We built 4 paraphrase systems by training English to English translation models using Moses (Koehn et al., 2007) with the default settings. Using our paraphrase corpus to train and to test, we divided the sentence clusters associated with each video into 90% for training and 10% for testing. We restricted our attention to sentences produced from the Tier-2 tasks

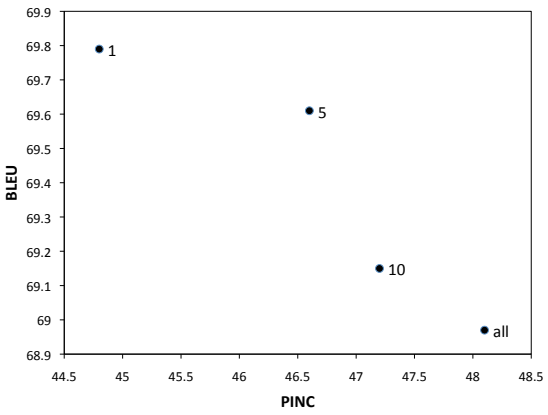


Figure 3: Evaluation of paraphrase systems trained on different numbers of parallel sentences. As more training pairs are used, the model produces more varied sentences (PINC) but preserves the meaning less well (BLEU)

in order to avoid excessive noise in the datasets, resulting in 28,785 training sentences and 3,367 test sentences. To construct the training examples, we randomly paired each sentence with 1, 5, 10, or all parallel descriptions of the same video segment. This corresponds to 28K, 143K, 287K, and 449K training pairs respectively. For the test set, we used each sentence once as the source sentence with all parallel descriptions as references (there were 16 references on average, with a minimum of 10 and a maximum of 31.) We also included the source sentence as a reference for itself.

Overall, all the trained models produce reasonable paraphrase systems, even the model trained on just 28K single parallel sentences. Examples of the outputs produced by the models trained on single parallel sentences and on all parallel sentences are shown in Table 2. Some of the changes are simple word substitutions, e.g. *rabbit* for *bunny* or *gun* for *revolver*, while others are phrasal, e.g. *frying meat* for *browning pork* or *made a basket* for *scores in a basketball game*. One interesting result of using videos as the stimulus to collect training data is that sometimes the learned paraphrases are not based on linguistic closeness, but rather on visual similarity, e.g. substituting *cricket* for *baseball*.

To evaluate the results quantitatively, we used the BLEU/PINC metric. The performance of all the trained models is shown in Figure 3. Unsurprisingly, there is a tradeoff between preserving the meaning

Original sentence	Trained on 1 parallel sentence	Trained on all parallel sentences
a bunny is cleaning its paw	a rabbit is licking its paw	a rabbit is cleaning itself
a man fires a revolver	a man is shooting targets	a man is shooting a gun
a big turtle is walking	a huge turtle is walking	a large tortoise is walking
a guy is doing a flip over a park bench	a man does a flip over a bench	a man is doing stunts on a bench
milk is being poured into a mixer	a man is pouring milk into a mixer	a man is pouring milk into a bowl
children are practicing baseball	children are doing a cricket	children are playing cricket
a boy is doing karate	a man is doing karate	a boy is doing martial arts
a woman is browning pork in a pan	a woman is browning pork in a pan	a woman is frying meat in a pan
a player scores in a basketball game	a player made a basketball game	a player made a basket

Table 2: Examples of paraphrases generated by the trained models.

and producing more varied paraphrases. Systems trained on fewer parallel sentences are more conservative and make fewer mistakes. On the other hand, systems trained on more parallel sentences often produce very good paraphrases but are also more likely to diverge from the original meaning. As a comparison, evaluating each human description as a paraphrase for the other descriptions in the same cluster resulted in a BLEU score of 52.9 and a PINC score of 77.2. Thus, all the systems performed very well in terms of retaining semantic content, although not as well in producing novel sentences.

To validate the results suggested by the automatic metrics, we asked two fluent English speakers to rate the generated paraphrases on the following categories: semantic, dissimilarity, and overall. Semantic measures how well the paraphrase preserves the original meaning while dissimilarity measures how much the paraphrase differs from the source sentence. Each category is rated from 1 to 4, with 4 being the best. A paraphrase identical to the source sentence would receive a score of 4 for meaning and 1 for dissimilarity and overall. We randomly selected 200 source sentences and generated 2 paraphrases for each, representing the two extremes: one paraphrase produced by the model trained with single parallel sentences, and the other by the model trained with all parallel sentences. The average scores of the two human judges are shown in Table 3. The results confirm our finding that the system trained with single parallel sentences preserves the meaning better but is also more conservative.

	Semantic	Dissimilarity	Overall
1	3.09	2.65	2.51
All	2.91	2.89	2.43

Table 3: Average human ratings of the systems trained on single parallel sentences and on all parallel sentences.

5.2 Correlation with human judgments

Having established rough correspondences between BLEU/PINC scores and human judgments of semantic equivalence and lexical dissimilarity, we quantified the correlation between these automatic metrics and human ratings using Pearson's correlation coefficient, a measure of linear dependence between two random variables. We computed the inter-annotator agreement as well as the correlation between BLEU, PINC, PEM (Liu et al., 2010) and the average human ratings on the sentence level. Results are shown in Table 4.

In order to measure correlation, we need to score each paraphrase individually. Thus, we recomputed BLEU on the sentence level and left the PINC scores unchanged. While BLEU is typically not reliable at the single sentence level, our large number of reference sentences makes BLEU more stable even at this granularity. Empirically, BLEU correlates fairly well with human judgments of semantic equivalence, although still not as well as the inter-annotator agreement. On the other hand, PINC correlates as well as humans agree with each other in assessing lexical dissimilarity. We also computed each metric's correlation with the overall ratings, although neither should be used alone to assess the overall quality of paraphrases.

PEM had the worst correlation with human judgments of all the metrics. Since PEM was trained on newswire data, its poor adaptation to this domain is expected. However, given the large amount of training data needed (PEM was trained on 250K Chinese-

	Semantic	Dissimilarity	Overall
Judge A vs. B	0.7135	0.6319	0.4920
BLEU vs. Human	0.5095	N/A	0.2127
PINC vs. Human	N/A	0.6672	0.0775
PEM vs. Human	N/A	N/A	0.0654
PINC vs. Human (BLEU $>$ threshold)			
threshold $= 0$	N/A	0.6541	0.1817
threshold $= 30$	N/A	0.6493	0.1984
threshold $= 60$	N/A	0.6815	0.3986
threshold $= 90$	N/A	0.7922	0.4350
Combined BLEU and PINC vs. Human			
Arithmetic Mean	N/A	N/A	0.3173
Geometric Mean	N/A	N/A	0.3003
Harmonic Mean	N/A	N/A	0.3036
PINC $\times$ Sigmoid(BLEU)	N/A	N/A	0.3532

Table 4: Correlation between the human judges as well as between the automatic metrics and the human judges.

English sentence pairs and 2400 human ratings of paraphrase pairs), it is difficult to use PEM as a general metric. Adapting PEM to a new domain would require sufficient in-domain bilingual data to support paraphrase extraction. In contrast, our approach only requires monolingual data, and evaluation can be performed using arbitrarily small, highly-parallel datasets. Moreover, PEM requires sample human ratings in training, thereby lessening the advantage of having automatic metrics.

Since lexical dissimilarity is only desirable when the semantics of the original sentence is unchanged, we also computed correlation between PINC and the human ratings when BLEU is above certain thresholds. As we restrict our attention to the set of paraphrases with higher BLEU scores, we see an increase in correlation between PINC and the human assessments. This confirms our intuition that PINC is a more useful measure when semantic content has been preserved.

Finally, while we do not believe any single score could adequately describe the quality of a paraphrase outside of a specific application, we experimented with different ways of combining BLEU and PINC into a single score. Almost any simple combination, such as taking the average of the two, yielded decent correlation with the human ratings. The best correlation was achieved by taking the product of PINC and a sigmoid function of BLEU. This follows the intuition that semantic preservation is closer to a

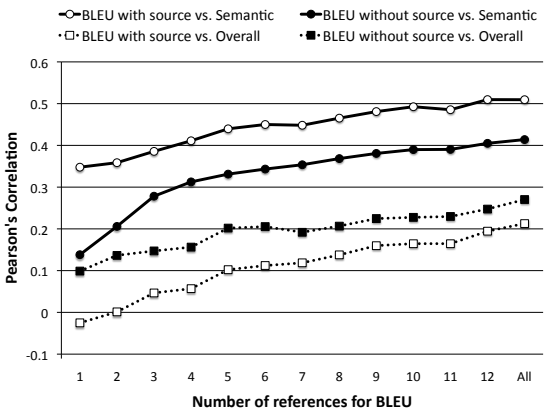


Figure 4: Correlation between BLEU and human judgments as we vary the number of reference sentences.

binary decision (i.e. a paraphrase either preserves the meaning or it does not, in which case PINC does not matter at all) than a linear function. We used an oracle to pick the best logistic function in our experiment. In practice, some sample human ratings would be required to tune this function. Other more complicated methods for combining BLEU and PINC are also possible with sample human ratings, such as using a SVM as was done in PEM.

We quantified the utility of our highly parallel data by computing the correlation between BLEU and human ratings when different numbers of references were available. The results are shown in Figure 4. As the number of references increases, the correlation with human ratings also increases. The graph also shows the effect of adding the source sentence as a reference. If our goal is to assess semantic equivalence only, then it is better to include the source sentence. If we are trying to assess the overall quality of the paraphrase, it is better to exclude the source sentence, since otherwise the metric will tend to favor paraphrases that introduce fewer changes.

5.3 Direct paraphrasing versus video annotation

In addition to collecting paraphrases through video annotations, we also experimented with the more traditional task of presenting a sentence to an annotator and explicitly asking for a paraphrase. We randomly selected a thousand sentences from our data and collected two paraphrases of each using Mechanical Turk. We conducted a post-annotation sur-

vey of workers who had completed both the video description and the direct paraphrasing tasks, and found that paraphrasing was considered more difficult and less enjoyable than describing videos. Of those surveyed, 92% found video annotations more enjoyable, and 75% found them easier. Based on the comments, the only drawback of the video annotation task is the time required to load and watch the videos. Overall, half of the workers preferred the video annotation task while only 16% of the workers preferred the paraphrasing task.

The data produced by the direct paraphrasing task also diverged less, since the annotators were inevitably biased by lexical choices and word order in the original sentences. On average, a direct paraphrase had a PINC score of 70.08, while a parallel description of the same video had a score of 78.75.

6 Discussions and Future Work

While our data collection framework yields useful parallel data, it also has some limitations. Finding appropriate videos is time-consuming and remains a bottleneck in the process. Also, more abstract actions such as *reducing the deficit* or *fighting for justice* cannot be easily captured by our method. One possible solution is to use longer video snippets or other visual stimuli such as graphs, schemas, or illustrated storybooks to convey more complicated information. However, the increased complexity is also likely to reduce the semantic closeness of the parallel descriptions.

Another limitation is that sentences produced by our framework tend to be short and follow similar syntactic structures. Asking annotators to write multiple descriptions or longer descriptions would result in more varied data but at the cost of more noise in the alignments. Other than descriptions, we could also ask the annotators for more complicated responses such as "fill in the blanks" in a dialogue (e.g. "If you were this person in the video, what would you say at this point?"), their opinion of the event shown, or the moral of the story. However, as with the difficulty of aligning news stories, finding paraphrases within these more complex responses could require additional annotation efforts.

In our experiments, we only used a subset of our corpus to avoid dealing with excessive noise. However, a significant portion of the remaining data is useful. Thus, an automatic method for filtering those sentences could allow us to utilize even more of the data. For example, sentences from the Tier-2 tasks could be used as positive examples to train a string classifier to determine whether a noisy sentence belongs in the same cluster or not.

We have so far used BLEU to measure semantic adequacy since it is the most common MT metric. However, other more advanced MT metrics that have shown higher correlation with human judgments could also be used.

In addition to paraphrasing, our data collection framework could also be used to produces useful data for machine translation and computer vision. By pairing up descriptions of the same video in different languages, we obtain parallel data without requiring any bilingual skills. Another application for our data is to apply it to computer vision tasks such as video retrieval. The dataset can be readily used to train and evaluate systems that can automatically generate full descriptions of unseen videos. As far as we know, there are currently no datasets that contain whole-sentence descriptions of open-domain video segments.

7 Conclusion

We introduced a data collection framework that produces highly parallel data by asking different annotators to describe the same video segments. Deploying the framework on Mechanical Turk over a two-month period yielded 85K English descriptions for 2K videos, one of the largest paraphrase data resources publicly available. In addition, the highly parallel nature of the data allows us to use standard MT metrics such as BLEU to evaluate semantic adequacy reliably. Finally, we also introduced a new metric, PINC, to measure the lexical dissimilarity between the source sentence and the paraphrase.

Acknowledgments

We are grateful to everyone in the NLP group at Microsoft Research and Natural Language Learning group at UT Austin for helpful discussions and feedback. We thank Chris Brockett, Raymond Mooney, Katrin Erk, Jason Baldridge and the anonymous reviewers for helpful comments on a previous draft.

References

Vamshi Ambati and Stephan Vogel. 2010. Can crowds build parallel corpora for machine translation systems? In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of Human Language Technology Conference / North American Association for Computational Linguistics Annual Meeting (HLT-NAACL-2003)*.

Regina Barzilay and Kathleen McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-2001)*.

Michael Bloodgood and Chris Callison-Burch. 2010a. Bucking the trend: Large-scale cost-focused active learning for statistical machine translation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-2010)*.

Michael Bloodgood and Chris Callison-Burch. 2010b. Using Mechanical Turk to build machine translation evaluation sets. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Olivia Buzek, Philip Resnik, and Benjamin B. Bederson. 2010. Error driven paraphrase annotation using Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proceedings of Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL-06)*.

Chris Callison-Burch, Trevor Cohn, and Mirella Lapata. 2008. Parametric: An automatic evaluation metric for paraphrasing. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-2008)*.

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-2008)*.

Chris Callison-Burch. 2009. Fast, cheap, and creative: Evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-2009)*.

Wallace L. Chafe. 1997. *The Pear Stories: Cognitive, Cultural and Linguistic Aspects of Narrative Production*. Ablex, Norwood, NJ.

Trevor Cohn, Chris Callison-Burch, and Mirella Lapata. 2008. Constructing corpora for the development and evaluation of paraphrase systems. *Computational Linguistics*, 34:597–614, December.

Michael Denkowski and Alon Lavie. 2010. Exploring normalization techniques for human judgments of machine translation adequacy collected using Amazon Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Michael Denkowski, Hassan Al-Haj, and Alon Lavie. 2010. Turker-assisted paraphrasing for English-Arabic machine translation. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-2004)*.

Li Fei-Fei, Asha Iyer, Christof Koch, and Pietro Perona. 2007. What do we perceive in a glance of a real-world scene? *Journal of Vision*, 7(1):1–29.

Ali Ibrahim, Boris Katz, and Jimmy Lin. 2003. Extracting structural paraphrases from aligned monolingual corpora. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*.

Ann Irvine and Alexandre Klementiev. 2010. Using Mechanical Turk to annotate lexicons for less commonly used languages. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*.

Stanley Kok and Chris Brockett. 2010. Hitting the right paraphrases in good time. In *Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT-2010)*.

Dekang Lin and Patrick Pantel. 2001. DIRT-discovery of inference rules from text. In *Proceedings of the*

Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2001).

Chang Liu, Daniel Dahlmeier, and Hwee Tou Ng. 2010. PEM: A paraphrase evaluation metric exploiting parallel texts. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-2010)*.

Xiaojuan Ma and Perry R. Cook. 2009. How well do visual verbs work in daily communication for young and old adults. In *Proceedings of ACM CHI 2009 Conference on Human Factors in Computing Systems*.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of Human Language Technology Conference / North American Association for Computational Linguistics Annual Meeting (HLT-NAACL-2003)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pages 311–318, Philadelphia, PA, July.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-2004)*.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. 2010. Collecting image annotations using Amazon's Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Yusuke Shinyama, Satoshi Sekine, Kiyoshi Sudo, and Ralph Grishman. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of Human Language Technology Conference*.

Luis von Ahn and Laura Dabbish. 2004. Labeling images with a computer game. In *Proceedings of ACM CHI 2004 Conference on Human Factors in Computing Systems*.

Omar F. Zaidan and Chris Callison-Burch. 2009. Feasibility of human-in-the-loop minimum error rate training. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-2009)*.

A Large Scale Distributed Syntactic, Semantic and Lexical Language Model for Machine Translation

Ming Tan Wenli Zhou Lei Zheng Shaojun Wang
Kno.e.sis Center
Department of Computer Science and Engineering
Wright State University
Dayton, OH 45435, USA
{tan.6,zhou.23,lei.zheng,shaojun.wang}@wright.edu

Abstract

This paper presents an attempt at building a large scale distributed composite language model that simultaneously accounts for local word lexical information, mid-range sentence syntactic structure, and long-span document semantic content under a directed Markov random field paradigm. The composite language model has been trained by performing a convergent N-best list approximate EM algorithm that has linear time complexity and a follow-up EM algorithm to improve word prediction power on corpora with up to a billion tokens and stored on a supercomputer. The large scale distributed composite language model gives drastic perplexity reduction over n-grams and achieves significantly better translation quality measured by the BLEU score and "readability" when applied to the task of re-ranking the N-best list from a state-of-the-art parsing-based machine translation system.

1 Introduction

The Markov chain (n-gram) source models, which predict each word on the basis of previous n-1 words, have been the workhorses of state-of-the-art speech recognizers and machine translators that help to resolve acoustic or foreign language ambiguities by placing higher probability on more likely original underlying word strings. Research groups (Brants et al., 2007; Zhang, 2008) have shown that using an immense distributed computing paradigm, up to 6-grams can be trained on up to billions and trillions of words, yielding consistent system improvements, but Zhang (2008) did not observe much improvement beyond 6-grams. Although the Markov chains are efficient at encoding local word interactions, the n-gram model clearly ignores the rich syntactic and semantic structures that constrain natural languages. As the machine translation (MT) working groups stated on page 3 of their final report (Lavie et al., 2006), "These approaches have resulted in small improvements in MT quality, but have not fundamentally solved the problem. There is a dire need for developing novel approaches to language modeling."

Wang et al. (2006) integrated n-gram, structured language model (SLM) (Chelba and Jelinek, 2000) and probabilistic latent semantic analysis (PLSA) (Hofmann, 2001) under the directed MRF framework (Wang et al., 2005) and studied the stochastic properties for the composite language model. They derived a *generalized inside-outside* algorithm to train the composite language model from a general EM (Dempster et al., 1977) by following Jelinek's ingenious definition of the inside and outside probabilities for SLM (Jelinek, 2004) with 6th order of sentence length time complexity. Unfortunately, there are no experimental results reported.

In this paper, we study the same composite language model. Instead of using the 6th order generalized inside-outside algorithm proposed in (Wang et al., 2006), we train this composite model by a convergent N-best list approximate EM algorithm that has linear time complexity and a follow-up EM algorithm to improve word prediction power. We conduct comprehensive experiments on corpora with 44 million tokens, 230 million tokens, and 1.3 billion tokens and compare perplexity results with n-grams (n=3,4,5 respectively) on these three corpora, we obtain drastic perplexity reductions. Finally, we ap-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 201–210,
Portland, Oregon, June 19-24, 2011.

ply our language models to the task of re-ranking the N-best list from Hiero (Chiang, 2005; Chiang, 2007), a state-of-the-art parsing-based MT system, we achieve significantly better translation quality measured by the BLEU score and "readability".

2 Composite language model

The n-gram language model is essentially a word predictor that given its entire document history it predicts next word w_{k+1} based on the last n-1 words with probability $p(w_{k+1}|w_{k-n+2}^{k})$ where $w_{k-n+2}^{k} = w_{k-n+2}, \cdots, w_k$.

The SLM (Chelba and Jelinek, 1998; Chelba and Jelinek, 2000) uses syntactic information beyond the regular n-gram models to capture sentence level long range dependencies. The SLM is based on statistical parsing techniques that allow syntactic analysis of sentences; it assigns a probability $p(W, T)$ to every sentence W and every possible binary parse T. The terminals of T are the words of W with POS tags, and the nodes of T are annotated with phrase headwords and non-terminal labels. Let W be a sentence of length n words to which we have prepended the sentence beginning marker $<s>$ and appended the sentence end marker $</s>$ so that $w_0 =<s>$ and $w_{n+1} =</s>$. Let $W_k = w_0, \cdots, w_k$ be the word k-prefix of the sentence – the words from the beginning of the sentence up to the current position k and $W_k T_k$ the word-parse k-prefix. A word-parse k-prefix has a set of exposed heads $h_{-m}, \cdots, h_{-1}$, with each head being a pair (headword, non-terminal label), or in the case of a root-only tree (word, POS tag). An *m-th order SLM* (m-SLM) has three operators to generate a sentence: WORD-PREDICTOR predicts the next word w_{k+1} based on the m left-most exposed headwords $h_{-m}^{-1} = h_{-m}, \cdots, h_{-1}$ in the word-parse k-prefix with probability $p(w_{k+1}|h_{-m}^{-1})$, and then passes control to the TAGGER; the TAGGER predicts the POS tag t_{k+1} to the next word w_{k+1} based on the next word w_{k+1} and the POS tags of the m left-most exposed headwords h_{-m}^{-1} in the word-parse k-prefix with probability $p(t_{k+1}|w_{k+1}, h_{-m}.\text{tag}, \cdots, h_{-1}.\text{tag})$; the CONSTRUCTOR builds the partial parse T_k from T_{k-1}, w_k, and t_k in a series of moves ending with NULL, where a parse move a is made with probability $p(a|h_{-m}^{-1})$; $a \in \mathcal{A}$={(unary, NTlabel), (adjoin-left, NTlabel), (adjoin-right, NTlabel), null}. Once the CONSTRUCTOR hits NULL, it passes control to the WORD-PREDICTOR. See detailed description in (Chelba and Jelinek, 2000).

A PLSA model (Hofmann, 2001) is a generative probabilistic model of word-document co-occurrences using the bag-of-words assumption described as follows: (i) choose a document d with probability $p(d)$; (ii) SEMANTIZER: select a semantic class g with probability $p(g|d)$; and (iii) WORD-PREDICTOR: pick a word w with probability $p(w|g)$. Since only one pair of (d, w) is being observed, as a result, the joint probability model is a mixture of log-linear model with the expression $p(d, w) = p(d) \sum_g p(w|g)p(g|d)$. Typically, the number of documents and vocabulary size are much larger than the size of latent semantic class variables. Thus, latent semantic class variables function as bottleneck variables to constrain word occurrences in documents.

When combining n-gram, m order SLM and PLSA models together to build a composite generative language model under the directed MRF paradigm (Wang et al., 2005; Wang et al., 2006), the TAGGER and CONSTRUCTOR in SLM and SEMANTIZER in PLSA remain unchanged; however the WORD-PREDICTORs in n-gram, m-SLM and PLSA are combined to form a stronger WORD-PREDICTOR that generates the next word, w_{k+1}, not only depending on the m left-most exposed headwords h_{-m}^{-1} in the word-parse k-prefix but also its n-gram history w_{k-n+2}^{k} and its semantic content g_{k+1}. The parameter for WORD-PREDICTOR in the composite n-gram/m-SLM/PLSA language model becomes $p(w_{k+1}|w_{k-n+2}^{k} h_{-m}^{-1} g_{k+1})$. The resulting composite language model has an even more complex dependency structure but with more expressive power than the original SLM. Figure 1 illustrates the structure of a composite n-gram/m-SLM/PLSA language model.

The composite n-gram/m-SLM/PLSA language model can be formulated as a directed MRF model (Wang et al., 2006) with local normalization constraints for the parameters of each model component, WORD-PREDICTOR, TAGGER, CONSTRUCTOR, SEMANTIZER, i.e., $\sum_{w \in \mathcal{V}} p(w|w_{-n+1}^{-1} h_{-m}^{-1} g) = 1, \sum_{t \in \mathcal{O}} p(t|w h_{-m}^{-1}.\text{tag}) = 1, \sum_{a \in \mathcal{A}} p(a|h_{-m}^{-1}) = 1, \sum_{g \in \mathcal{G}} p(g|d) = 1$.

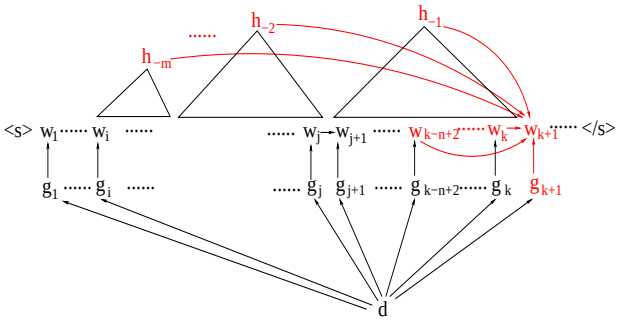


Figure 1: *A composite n-gram/m-SLM/PLSA language model where the hidden information is the parse tree T and semantic content g. The WORD-PREDICTOR generates the next word w_{k+1} with probability $p(w_{k+1}|w_{k-n+2}^{k}h_{-m}^{-1}g_{k+1})$ instead of $p(w_{k+1}|w_{k-n+2}^{k})$, $p(w_{k+1}|h_{-m}^{-1})$ and $p(w_{k+1}|g_{k+1})$ respectively.*

3 Training algorithm

Under the composite n-gram/m-SLM/PLSA language model, the likelihood of a training corpus $\mathcal{D}$, a collection of documents, can be written as

$$\mathcal{L}(\mathcal{D},p)=\prod_{d\in\mathcal{D}}\left(\prod_{l}\left(\sum_{G^l}\left(\sum_{T^l}P_p(W^l,T^l,G^l|d)\right)\right)\right) \quad (1)$$

where (W^l,T^l,G^l,d) denote the joint sequence of the lth sentence W^l with its parse tree structure T^l and semantic annotation string G^l in document d. This sequence is produced by a unique sequence of model actions: WORD-PREDICTOR, TAGGER, CONSTRUCTOR, SEMANTIZER moves, its probability is obtained by chaining the probabilities of these moves

$$\begin{aligned}
&P_p(W^l,T^l,G^l|d)\\
=&\prod_{g\in\mathcal{G}}\left(p(g|d)^{\#(g,W^l,G^l,d)}\prod_{h_{-1},\cdots,h_{-m}\in\mathcal{H}}\right.\\
&\prod_{w,w_{-1},\cdots,w_{-n+1}\in\mathcal{V}}p(w|w_{-n+1}^{-1}h_{-m}^{-1}g)^{\#(w_{-n+1}^{-1}wh_{-m}^{-1}g,W^l,T^l,G^l,d)}\\
&\prod_{t\in\mathcal{O}}p(t|wh_{-m}^{-1}.\text{tag})^{\#(t,wh_{-m}^{-1}.\text{tag},W^l,T^l,d)}\\
&\left.\prod_{a\in\mathcal{A}}p(a|h_{-m}^{-1})^{\#(a,h_{-m}^{-1},W^l,T^l,d)}\right)
\end{aligned}$$

where $\#(g,W^l,G^l,d)$ is the count of semantic content g in semantic annotation string G^l of the lth sentence W^l in document d, $\#(w_{-n+1}^{-1}wh_{-m}^{-1}g,W^l,T^l,G^l,d)$ is the count of n-grams, its m most recent exposed headwords and semantic content g in parse T^l and semantic annotation string G^l of the lth sentence W^l in document d, $\#(twh_{-m}^{-1}.\text{tag},W^l,T^l,d)$ is the count of tag t predicted by word w and the tags of m most recent exposed headwords in parse tree T^l of the lth sentence W^l in document d, and finally $\#(ah_{-m}^{-1},W^l,T^l,d)$ is the count of constructor move a conditioning on m exposed headwords h_{-m}^{-1} in parse tree T^l of the lth sentence W^l in document d.

The objective of maximum likelihood estimation is to maximize the likelihood $\mathcal{L}(\mathcal{D},p)$ respect to model parameters. For a given sentence, its parse tree and semantic content are hidden and the number of parse trees grows faster than exponential with sentence length, Wang et al. (2006) have derived a generalized inside-outside algorithm by applying the standard EM algorithm. However, the complexity of this algorithm is 6th order of sentence length, thus it is computationally too expensive to be practical for a large corpus even with the use of pruning on charts (Jelinek and Chelba, 1999; Jelinek, 2004).

3.1 N-best list approximate EM

Similar to SLM (Chelba and Jelinek, 2000), we adopt an N-best list approximate EM re-estimation with modular modifications to seamlessly incorporate the effect of n-gram and PLSA components. Instead of maximizing the likelihood $\mathcal{L}(\mathcal{D},p)$, we maximize the N-best list likelihood,

$$\begin{aligned}
\max_{\mathcal{T'}_N}\mathcal{L}(\mathcal{D},p,\mathcal{T'}_N)=\prod_{d\in\mathcal{D}}\left(\prod_{l}\left(\max_{\mathcal{T'}^l_N\in\mathcal{T'}_N}\sum_{G^l}\right.\right.\\
\left.\left.\left(\sum_{T^l\in\mathcal{T'}^l_N,||\mathcal{T'}^l_N||=N}P_p(W^l,T^l,G^l|d)\right)\right)\right)
\end{aligned}$$

where $\mathcal{T'}^l_N$ is a set of N parse trees for sentence W^l in document d and $||\cdot||$ denotes the cardinality and $\mathcal{T'}_N$ is a collection of $\mathcal{T'}^l_N$ for sentences over entire corpus $\mathcal{D}$.

The N-best list approximate EM involves two steps:

1. N-best list search: For each sentence W in document d, find N-best parse trees,

$$\mathcal{T}^l_N=\arg\max_{\mathcal{T'}^l_N}\left\{\sum_{G^l}\sum_{T^l\in\mathcal{T'}^l_N}P_p(W^l,T^l,G^l|d),||\mathcal{T'}^l_N||=N\right\}$$

   and denote $\mathcal{T}_N$ as the collection of N-best list parse trees for sentences over entire corpus $\mathcal{D}$ under model parameter p.

2. EM update: Perform one iteration (or several iterations) of EM algorithm to estimate model

parameters that maximizes N-best-list likelihood of the training corpus $\mathcal{D}$,

$$\tilde{\mathcal{L}}(\mathcal{D}, p, \mathcal{T}_N) = \prod_{d \in \mathcal{D}} (\prod_{l} (\sum_{G^l} (\sum_{T^l \in \mathcal{T}_N^l, \in \mathcal{T}_N} P_p(W^l, T^l, G^l | d))))$$

That is,

(a) E-step: Compute the auxiliary function of the N-best-list likelihood

$$\tilde{Q}(p', p, \mathcal{T}_N) = \sum_{d \in \mathcal{D}} \sum_{l} \sum_{G^l} \sum_{T^l \in \mathcal{T}_N^l, \in \mathcal{T}_N} P_p(T^l, G^l | W^l, d) \log P_{p'}(W^l, T^l, G^l | d)$$

(b) M-step: Maximize $\tilde{Q}(p', p, \mathcal{T}_N)$ with respect to p' to get new update for p.

Iterate steps (1) and (2) until the convergence of the N-best-list likelihood. Due to space constraints, we omit the proof of the convergence of the N-best list approximate EM algorithm which uses Zangwill's global convergence theorem (Zangwill, 1969).

N-best list search strategy: To extract the N-best parse trees, we adopt a synchronous, multi-stack search strategy that is similar to the one in (Chelba and Jelinek, 2000), which involves a set of stacks storing partial parses of the most likely ones for a given prefix W_k and the less probable parses are purged. Each stack contains hypotheses (partial parses) that have been constructed by the same number of WORD-PREDICTOR and the same number of CONSTRUCTOR operations. The hypotheses in each stack are ranked according to the $\log(\sum_{G_k} P_p(W_k, T_k, G_k|d))$ score with the highest on top, where $P_p(W_k, T_k, G_k|d)$ is the joint probability of prefix $W_k = w_0, \cdots, w_k$ with its parse structure T_k and semantic annotation string $G_k = g_1, \cdots, g_k$ in a document d. A stack vector consists of the ordered set of stacks containing partial parses with the same number of WORD-PREDICTOR operations but different number of CONSTRUCTOR operations. In WORD-PREDICTOR and TAGGER operations, some hypotheses are discarded due to the maximum number of hypotheses the stack can contain at any given time. In CONSTRUCTOR operation, the resulting hypotheses are discarded due to either finite stack size or the log-probability threshold: the maximum tolerable difference between the log-probability score of the top-most hypothesis and the bottom-most hypothesis at any given state of the stack.

EM update: Once we have the N-best parse trees for each sentence in document d and N-best topics for document d, we derive the EM algorithm to estimate model parameters.

In E-step, we compute the expected count of each model parameter over sentence W^l in document d in the training corpus $\mathcal{D}$. For the WORD-PREDICTOR and the SEMANTIZER, the number of possible semantic annotation sequences is exponential, we use forward-backward recursive formulas that are similar to those in hidden Markov models to compute the expected counts. We define the forward vector $\alpha^l(g|d)$ to be

$$\alpha^l_{k+1}(g|d) = \sum_{G^l_k} P_p(W^l_k, T^l_k, w^k_{k-n+2} w_{k+1} h^{-1}_{-m} g, G^l_k | d)$$

that can be recursively computed in a forward manner, where W^l_k is the word k-prefix for sentence W^l, T^l_k is the parse for k-prefix. We define backward vector $\beta^l(g|d)$ to be

$$\beta^l_{k+1}(g|d) = \sum_{G^l_{k+1,\cdot}} P_p(W^l_{k+1,\cdot}, T^l_{k+1,\cdot}, G^l_{k+1,\cdot} | w^k_{k-n+2} w_{k+1} h^{-1}_{-m} g, d)$$

that can be computed in a backward manner, here $W^l_{k+1,\cdot}$ is the subsequence after k+1th word in sentence W^l, $T^l_{k+1,\cdot}$ is the incremental parse structure after the parse structure T^l_{k+1} of word k+1-prefix W^l_{k+1} that generates parse tree T^l, $G^l_{k+1,\cdot}$ is the semantic subsequence in G^l relevant to $W^l_{k+1,\cdot}$. Then, the expected count of $w^{-1}_{-n+1} w h^{-1}_{-m} g$ for the WORD-PREDICTOR on sentence W^l in document d is

$$\sum_{G^l} P_p(T^l, G^l | W^l, d) \#(w^{-1}_{-n+1} w h^{-1}_{-m} g, W^l, T^l, G^l, d)$$
$$= \sum_{l} \sum_{k} \alpha^l_{k+1}(g|d) \beta^l_{k+1}(g|d) p(g|d)$$
$$\delta(w^k_{k-n+2} w_{k+1} h^{-1}_{-m} g_{k+1} = w^{-1}_{-n+1} w h^{-1}_{-m} g) / P_p(W^l | d)$$

where $\delta(\cdot)$ is an indicator function and the expected count of g for the SEMANTIZER on sentence W^l in document d is

$$\sum_{G^l} P_p(T^l, G^l | W^l, d) \#(g, W^l, G^l, d)$$
$$= \sum_{k=0}^{j-1} \alpha^l_{k+1}(g|d) \beta^l_{k+1}(g|d) p(g|d) / P_p(W^l | d)$$

For the TAGGER and the CONSTRUCTOR, the expected count of each event of $t w h^{-1}_{-m}.\mathrm{tag}$ and $a h^{-1}_{-m}$ over parse T^l of sentence W^l in

document d is the real count appeared in parse tree T^l of sentence W^l in document d times the conditional distribution $P_p(T^l|W^l,d) = P_p(T^l,W^l|d)/\sum_{T^l\in\mathcal{T}^l} P_p(T^l,W^l|d)$ respectively.

In M-step, the recursive linear interpolation scheme (Jelinek and Mercer, 1981) is used to obtain a smooth probability estimate for each model component, WORD-PREDICTOR, TAGGER, and CONSTRUCTOR. The TAGGER and CONSTRUCTOR are conditional probabilistic models of the type $p(u|z_1,\cdots,z_n)$ where $u, z_1,\cdots,z_n$ belong to a mixed set of words, POS tags, NTtags, CONSTRUCTOR actions (u only), and $z_1,\cdots,z_n$ form a linear Markov chain. The recursive mixing scheme is the standard one among relative frequency estimates of different orders $k = 0,\cdots,n$ as explained in (Chelba and Jelinek, 2000). The WORD-PREDICTOR is, however, a conditional probabilistic model $p(w|w_{-n+1}^{-1}h_{-m}^{-1}g)$ where there are three kinds of context w_{-n+1}^{-1}, h_{-m}^{-1} and g, each forms a linear Markov chain. The model has a combinatorial number of relative frequency estimates of different orders among three linear Markov chains. We generalize Jelinek and Mercer's original recursive mixing scheme (Jelinek and Mercer, 1981) and form a lattice to handle the situation where the context is a mixture of Markov chains.

3.2 Follow-up EM

As explained in (Chelba and Jelinek, 2000), for the SLM component, a large fraction of the partial parse trees that can be used for assigning probability to the next word do not survive in the synchronous, multi-stack search strategy, thus they are not used in the N-best approximate EM algorithm for the estimation of WORD-PREDICTOR to improve its predictive power. To remedy this weakness, we estimate WORD-PREDICTOR using the algorithm below.

The *language model* probability assignment for the word at position k+1 in the input sentence of document d can be computed as

$$P_p(w_{k+1}|W_k,d) = \sum_{h_{-m}^{-1}\in T_k; T_k\in Z_k, g_{k+1}\in\mathcal{G}_d} p(w_{k+1}|w_{k-n+2}^{k}h_{-m}^{-1}g_{k+1}) P_p(T_k|W_k,d)p(g_{k+1}|d) \quad (2)$$

where $P_p(T_k|W_k,d) = \frac{\sum_{G_k} P_p(W_k,T_k,G_k|d)}{\sum_{T_k\in Z_k}\sum_{G_k} P_p(W_k,T_k,G_k|d)}$ and Z_k is the set of all parses present in the stacks at the current stage k during the synchronous multi-stack pruning strategy and it is a function of the word k-prefix W_k.

The likelihood of a training corpus $\mathcal{D}$ under this language model probability assignment that uses partial parse trees generated during the process of the synchronous, multi-stack search strategy can be written as

$$\tilde{\mathcal{L}}(\mathcal{D},p) = \prod_{d\in\mathcal{D}}\prod_{l}\Big(\sum_{k} P_p(w_{k+1}^{(l)}|W_k^l,d)\Big) \quad (3)$$

We employ a second stage of parameter re-estimation for $p(w_{k+1}|w_{k-n+2}^{k}h_{-m}^{-1}g_{k+1})$ and $p(g_{k+1}|d)$ by using EM again to maximize Equation (3) to improve the predictive power of WORD-PREDICTOR.

3.3 Distributed architecture

When using very large corpora to train our composite language model, both the data and the parameters can't be stored in a single machine, so we have to resort to distributed computing. The topic of large scale distributed language models is relatively new, and existing works are restricted to n-grams only (Brants et al., 2007; Emami et al., 2007; Zhang et al., 2006). Even though all use distributed architectures that follow the client-server paradigm, the real implementations are in fact different. Zhang et al. (2006) and Emami et al. (2007) store training corpora in suffix arrays such that one sub-corpus per server serves raw counts and test sentences are loaded in a client. This implies that when computing the language model probability of a sentence in a client, all servers need to be contacted for each n-gram request. The approach by Brants et al. (2007) follows a standard MapReduce paradigm (Dean and Ghemawat, 2004): the corpus is first divided and loaded into a number of clients, and n-gram counts are collected at each client, then the n-gram counts mapped and stored in a number of servers, resulting in exactly one server being contacted per n-gram when computing the language model probability of a sentence. We adopt a similar approach to Brants et al. and make it suitable to perform iterations of N-best list approximate EM algorithm, see Figure 2. The corpus is divided and loaded into a number of clients. We use a public available parser to parse the sentences in each client to get the initial counts for $w_{-n+1}^{-1}wh_{-m}^{-1}g$ etc., finish the Map part, and then the counts for a particular $w_{-n+1}^{-1}wh_{-m}^{-1}g$ at different clients are summed up and stored in one

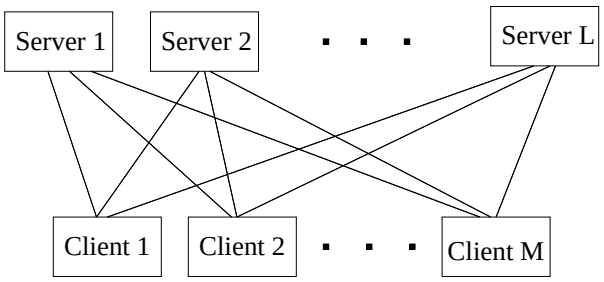


Figure 2: *Distributed architecture is essentially a MapReduce paradigm: clients store partitioned data and perform E-step: compute expected counts, this is Map; servers store parameters (counts) for M-step where counts of $w_{-n+1}^{-1}wh_{-m}^{-1}g$ are hashed by word w_{-1} (or h_{-1}) and its topic g to evenly distribute these model parameters into servers as much as possible, this is Reduce.*

of the servers by hashing through the word w_{-1} (or h_{-1}) and its topic g, finish the Reduce part. This is the initialization of the N-best list approximate EM step. Each client then calls the servers for parameters to perform synchronous multi-stack search for each sentence to get the N-best list parse trees. Again, the expected count for a particular parameter of $w_{-n+1}^{-1}wh_{-m}^{-1}g$ at the clients are computed, thus we finish a Map part, then summed up and stored in one of the servers by hashing through the word w_{-1} (or h_{-1}) and its topic g, thus we finish the Reduce part. We repeat this procedure until convergence.

Similarly, we use a distributed architecture as in Figure 2 to perform the follow-up EM algorithm to re-estimate WORD-PREDICTOR.

4 Experimental results

We have trained our language models using three different training sets: one has 44 million tokens, another has 230 million tokens, and the other has 1.3 billion tokens. An independent test set which has 354 k tokens is chosen. The independent check data set used to determine the linear interpolation coefficients has 1.7 million tokens for the 44 million tokens training corpus, 13.7 million tokens for both 230 million and 1.3 billion tokens training corpora. All these data sets are taken from the LDC English Gigaword corpus with non-verbalized punctuation and we remove all punctuation. Table 1 gives the detailed information on how these data sets are chosen from the LDC English Gigaword corpus.

The vocabulary sizes in all three cases are:

- word (also WORD-PREDICTOR operation) vocabulary: 60 k, open - all words outside the vocabulary are mapped to the <unk> token, these 60 k words are chosen from the most frequently occurred words in 44 millions tokens corpus;
- POS tag (also TAGGER operation) vocabulary: 69, closed;
- non-terminal tag vocabulary: 54, closed;
- CONSTRUCTOR operation vocabulary: 157, closed.

	1.3 BILLION TOKENS TRAINING CORPUS
AFP	19940512.0003 ∼ 19961015.0568
AFW	19941111.0001 ∼ 19960414.0652
NYT	19940701.0001 ∼ 19950131.0483
NYT	19950401.0001 ∼ 20040909.0063
XIN	19970901.0001 ∼ 20041125.0119
	230 MILLION TOKENS TRAINING CORPUS
AFP	19940622.0336 ∼ 19961031.0797
APW	19941111.0001 ∼ 19960419.0765
NYT	19940701.0001 ∼ 19941130.0405
	44 MILLION TOKENS TRAINING CORPUS
AFP	19940601.0001 ∼ 19950721.0137
	13.7 MILLION TOKENS CHECK CORPUS
NYT	19950201.0001 ∼ 19950331.0494
	1.7 MILLION TOKENS CHECK CORPUS
AFP	19940512.0003 ∼ 19940531.0197
	354 K TOKENS TEST CORPUS
CNA	20041101.0006 ∼ 20041217.0009

Table 1: The corpora used in our experiments are selected from the LDC English Gigaword corpus and specified in this table, AFP, AFW, NYT, XIN and CNA denote the sections of the LDC English Gigaword corpus.

Similar to SLM (Chelba and Jelinek, 2000), after the parses undergo headword percolation and binarization, each model component of WORD-PREDICTOR, TAGGER, and CONSTRUCTOR is initialized from a set of parsed sentences. We use the "openNLP" software (Northedge, 2005) to parse a large amount of sentences in the LDC English Gigaword corpus to generate an automatic treebank, which has a slightly different word-tokenization than that of the manual treebank such as the Upenn Treebank used in (Chelba and Jelinek, 2000). For the 44 and 230 million tokens corpora, all sentences are automatically parsed and used to initialize model parameters, while for 1.3 billion tokens corpus, we parse the sentences from a portion of the corpus that

contain 230 million tokens, then use them to initialize model parameters. The parser at "openNLP" is trained by Upenn treebank with 1 million tokens and there is a mismatch between Upenn treebank and LDC English Gigaword corpus. Nevertheless, experimental results show that this approach is effective to provide initial values of model parameters.

As we have explained, the proposed EM algorithms can be naturally cast into a MapReduce framework, see more discussion in (Lin and Dyer, 2010). If we have access to a large cluster of machines with Hadoop installed that are powerful enough to process a billion tokens level corpus, we just need to specify a map function and a reduce function etc., Hadoop will automatically parallelize and execute programs written in this functional style. Unfortunately, we don't have this kind of resources available. Instead, we have access to a supercomputer at a supercomputer center with MPI installed that has more than 1000 core processors usable. Thus we implement our algorithms using C++ under MPI on the supercomputer, where we have to write C++ codes for Map part and Reduce part, and the MPI is used to take care of massage passing, scheduling, synchronization, etc. between clients and servers. This involves a fair amount of programming work, even though our implementation under MPI is not as reliable as under Hadoop but it is more efficient. We use up to 1000 core processors to train the composite language models for 1.3 billion tokens corpus where 900 core processors are used to store the parameters alone. We decide to use linearly smoothed trigram as the baseline model for 44 million token corpus, linearly smoothed 4-gram as the baseline model for 230 million token corpus, and linearly smoothed 5-gram as the baseline model for 1.3 billion token corpus. Model size is a big issue, we have to keep only a small set of topics due to the consideration in both computational time and resource demand. Table 2 shows the perplexity results and computation time of composite n-gram/PLSA language models that are trained on three corpora when the pre-defined number of total topics is 200 but different numbers of most likely topics are kept for each document in PLSA, the rest are pruned. For composite 5-gram/PLSA model trained on 1.3 billion tokens corpus, 400 cores have to be used to keep top 5 most likely topics. For composite trigram/PLSA model trained on 44M tokens corpus, the computation time increases drastically with less than 5% percent perplexity improvement. So in the following experiments, we keep top 5 topics for each document from total 200 topics and all other 195 topics are pruned.

All composite language models are first trained by performing N-best list approximate EM algorithm until convergence, then EM algorithm for a second stage of parameter re-estimation for WORD-PREDICTOR and SEMANTIZER until convergence. We fix the size of topics in PLSA to be 200 and then prune to 5 in the experiments, where the unpruned 5 topics in general account for 70% probability in $p(g|d)$. Table 3 shows comprehensive perplexity results for a variety of different models such as composite n-gram/m-SLM, n-gram/PLSA, m-SLM/PLSA, their linear combinations, etc., where we use online EM with fixed learning rate to re-estimate the parameters of the SEMANTIZER of test document. The m-SLM performs competitively with its counterpart n-gram (n=m+1) on large scale corpus. In Table 3, for composite n-gram/m-SLM model ($n = 3, m = 2$ and $n = 4, m = 3$) trained on 44 million tokens and 230 million tokens, we cut off its fractional expected counts that are less than a threshold 0.005, this significantly reduces the number of predictor's types by 85%. When we train the composite language on 1.3 billion tokens corpus, we have to both aggressively prune the parameters of WORD-PREDICTOR and shrink the order of n-gram and m-SLM in order to store them in a supercomputer having 1000 cores. In particular, for composite 5-gram/4-SLM model, its size is too big to store, thus we use its approximation, a linear combination of 5-gram/2-SLM and 2-gram/4-SLM, and for 5-gram/2-SLM or 2-gram/4-SLM, again we cut off its fractional expected counts that are less than a threshold 0.005, this significantly reduces the number of predictor's types by 85%. For composite 4-SLM/PLSA model, we cut off its fractional expected counts that are less than a threshold 0.002, again this significantly reduces the number of predictor's types by 85%. For composite 4-SLM/PLSA model or its linear combination with models, we ignore all the tags and use only the words in the 4 head words. In this table, we have three items missing (marked by —), since the size of corresponding model is

CORPUS	n	# OF TOPICS	PPL	TIME (HOURS)	# OF SERVERS	# OF CLIENTS	# OF TYPES OF $ww_{-n+1}^{-1}g$
44M	3	5	196	0.5	40	100	120.1M
	3	10	194	1.0	40	100	218.6M
	3	20	190	2.7	80	100	537.8M
	3	50	189	6.3	80	100	1.123B
	3	100	189	11.2	80	100	1.616B
	3	200	188	19.3	80	100	2.280B
230M	4	5	146	25.6	280	100	0.681B
1.3B	5	2	111	26.5	400	100	1.790B
	5	5	102	75.0	400	100	4.391B

Table 2: Perplexity (ppl) results and time consumed of composite n-gram/PLSA language model trained on three corpora when different numbers of most likely topics are kept for each document in PLSA.

LANGUAGE MODEL	44M n=3,m=2	REDUCTION	230M n=4,m=3	REDUCTION	1.3B n=5,m=4	REDUCTION
BASELINE n-GRAM (LINEAR)	262		200		138	
n-GRAM (KNESER-NEY)	244	6.9%	183	8.5%	—	—
m-SLM	279	-6.5%	190	5.0%	137	0.0%
PLSA	825	-214.9%	812	-306.0%	773	-460.0%
n-GRAM+m-SLM	247	5.7%	184	8.0%	129	6.5%
n-GRAM+PLSA	235	10.3%	179	10.5%	128	7.2%
n-GRAM+m-SLM+PLSA	222	15.3%	175	12.5%	123	10.9%
n-GRAM/m-SLM	243	7.3%	171	14.5%	(125)	9.4%
n-GRAM/PLSA	196	25.2%	146	27.0%	102	26.1%
m-SLM/PLSA	198	24.4%	140	30.0%	(103)	25.4%
n-GRAM/PLSA+m-SLM/PLSA	183	30.2%	140	30.0%	(93)	32.6%
n-GRAM/m-SLM+m-SLM/PLSA	183	30.2%	139	30.5%	(94)	31.9%
n-GRAM/m-SLM+n-GRAM/PLSA	184	29.8%	137	31.5%	(91)	34.1%
n-GRAM/m-SLM+n-GRAM/PLSA +m-SLM/PLSA	180	31.3%	130	35.0%	—	—
n-GRAM/m-SLM/PLSA	176	32.8%	—	—	—	—

Table 3: Perplexity results for various language models on test corpus, where + denotes linear combination, / denotes composite model; n denotes the order of n-gram and m denotes the order of SLM; the topic nodes are pruned from 200 to 5.

too big to store in the supercomputer. The composite n-gram/m-SLM/PLSA model gives significant perplexity reductions over baseline n-grams, $n = 3, 4, 5$ and m-SLMs, $m = 2, 3, 4$. The majority of gains comes from PLSA component, but when adding SLM component into n-gram/PLSA, there is a further 10% relative perplexity reduction.

We have applied our composite 5-gram/2-SLM+2-gram/4-SLM+5-gram/PLSA language model that is trained by 1.3 billion word corpus for the task of re-ranking the N-best list in statistical machine translation. We used the same 1000-best list that is used by Zhang et al. (2006). This list was generated on 919 sentences from the MT03 Chinese-English evaluation set by Hiero (Chiang, 2005; Chiang, 2007), a state-of-the-art parsing-based translation model. Its decoder uses a trigram language model trained with modified Kneser-Ney smoothing (Kneser and Ney, 1995) on a 200 million tokens corpus. Each translation has 11 features and language model is one of them. We substitute our language model and use MERT (Och, 2003) to optimize the BLEU score (Papineni et al., 2002). We partition the data into ten pieces, 9 pieces are used as training data to optimize the BLEU score (Papineni et al., 2002) by MERT (Och,

2003), a remaining single piece is used to re-rank the 1000-best list and obtain the BLEU score. The cross-validation process is then repeated 10 times (the folds), with each of the 10 pieces used exactly once as the validation data. The 10 results from the folds then can be averaged (or otherwise combined) to produce a single estimation for BLEU score. Table 4 shows the BLEU scores through 10-fold cross-validation. The composite 5-gram/2-SLM+2-gram/4-SLM+5-gram/PLSA language model gives 1.57% BLEU score improvement over the baseline and 0.79% BLEU score improvement over the 5-gram. This is because there is not much diversity on the 1000-best list, and essentially only $20 \sim 30$ distinct sentences are there in the 1000-best list. Chiang (2007) studied the performance of machine translation on Hiero, the BLEU score is 33.31% when n-gram is used to re-rank the N-best list, however, the BLEU score becomes significantly higher 37.09% when the n-gram is embedded directly into Hiero's one pass decoder, this is because there is not much diversity in the N-best list. It is expected that putting the our composite language into a one pass decoder of both phrase-based (Koehn et al., 2003) and parsing-based (Chiang, 2005; Chiang, 2007) MT systems should result in much improved BLEU scores.

SYSTEM MODEL	MEAN (%)
BASELINE	31.75
5-GRAM	32.53
5-GRAM/2-SLM+2-GRAM/4-SLM	32.87
5-GRAM/PLSA	33.01
5-GRAM/2-SLM+2-GRAM/4-SLM +5-GRAM/PLSA	33.32

Table 4: 10-fold cross-validation BLEU score results for the task of re-ranking the N-best list.

Besides reporting the BLEU scores, we look at the "readability" of translations similar to the study conducted by Charniak et al. (2003). The translations are sorted into four groups: good/bad syntax crossed with good/bad meaning by human judges, see Table 5. We find that many more sentences are perfect, many more are grammatically correct, and many more are semantically correct. The syntactic language model (Charniak, 2001; Charniak, 2003) only improves translations to have good grammar, but does not improve translations to preserve meaning. The composite 5-gram/2-SLM+2-gram/4-SLM+5-gram/PLSA language model improves both significantly. Bear in mind that Charniak et al. (2003) integrated Charniak's language model with the syntax-based translation model Yamada and Knight proposed (2001) to rescore a tree-to-string translation forest, whereas we use only our language model for N-best list re-ranking. Also, in the same study in (Charniak, 2003), they found that the outputs produced using the n-grams received higher scores from BLEU; ours did not. The difference between human judgments and BLEU scores indicate that closer agreement may be possible by incorporating syntactic structure and semantic information into the BLEU score evaluation. For example, semantically similar words like "insure" and "ensure" in the example of BLEU paper (Papineni et al., 2002) should be substituted in the formula, and there is a weight to measure the goodness of syntactic structure. This modification will lead to a better metric and such information can be provided by our composite language models.

SYSTEM MODEL	P	S	G	W
BASELINE	95	398	20	406
5-GRAM	122	406	24	367
5-GRAM/2-SLM +2-GRAM/4-SLM +5-GRAM/PLSA	151	425	33	310

Table 5: Results of "readability" evaluation on 919 translated sentences, P: perfect, S: only semantically correct, G: only grammatically correct, W: wrong.

5 Conclusion

As far as we know, this is the first work of building a complex large scale distributed language model with a principled approach that is more powerful than n-grams when both trained on a very large corpus with up to a billion tokens. We believe our results still hold on web scale corpora that have trillion tokens, since the composite language model effectively encodes long range dependencies of natural language that n-gram is not viable to consider. Of course, this implies that we have to take a huge amount of resources to perform the computation, nevertheless this becomes feasible, affordable, and cheap in the era of cloud computing.

References

L. Bahl and J. Baker,F. Jelinek and R. Mercer. 1977. Perplexityła measure of difficulty of speech recognition tasks. *94th Meeting of the Acoustical Society of America*, 62:S63, Supplement 1.

T. Brants et al.. 2007. Large language models in machine translation. *The 2007 Conference on Empirical Methods in Natural Language Processing* (EMNLP), 858-867.

E. Charniak. 2001. Immediate-head parsing for language models. *The 39th Annual Conference on Association of Computational Linguistics* (ACL), 124-131.

E. Charniak, K. Knight and K. Yamada. 2003. Syntax-based language models for statistical machine translation. MT Summit IX., Intl. Assoc. for Machine Translation.

C. Chelba and F. Jelinek. 1998. Exploiting syntactic structure for language modeling. *The 36th Annual Conference on Association of Computational Linguistics* (ACL), 225-231.

C. Chelba and F. Jelinek. 2000. Structured language modeling. *Computer Speech and Language*, 14(4):283-332.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. *The 43th Annual Conference on Association of Computational Linguistics* (ACL), 263-270.

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

J. Dean and S. Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. *Operating Systems Design and Implementation* (OSDI), 137-150.

A. Dempster, N. Laird and D. Rubin. 1977. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of Royal Statistical Society*, 39:1-38.

A. Emami, K. Papineni and J. Sorensen. 2007. Large-scale distributed language modeling. *The 32nd IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP), IV:37-40.

T. Hofmann. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1):177-196.

F. Jelinek and R. Mercer. 1981. Interpolated estimation of Markov source parameters from sparse data. *Pattern Recognition in Practice*, 381-397.

F. Jelinek and C. Chelba. 1999. Putting language into language modeling. *Sixth European Conference on Speech Communication and Technology* (EUROSPEECH), Keynote Paper 1.

F. Jelinek. 2004. Stochastic analysis of structured language modeling. *Mathematical Foundations of Speech and Language Processing*, 37-72, Springer-Verlag.

D. Jurafsky and J. Martin. 2008. *Speech and Language Processing*, 2nd Edition, Prentice Hall.

R. Kneser and H. Ney. 1995. Improved backing-off for m-gram language modeling. *The 20th IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP), 181-184.

P. Koehn, F. Och and D. Marcu. 2003. Statistical phrase-based translation. *The Human Language Technology Conference* (HLT), 48-54.

S. Khudanpur and J. Wu. 2000. Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling. *Computer Speech and Language*, 14(4):355-372.

A. Lavie et al. 2006. MINDS Workshops Machine Translation Working Group Final Report. http://www-nlpir.nist.gov/MINDS/FINAL/MT.web.pdf

J. Lin and C. Dyer. 2010. *Data-Intensive Text Processing with MapReduce*. Morgan and Claypool Publishers.

R. Northedge. 2005. OpenNLP software http://www.codeproject.com/KB/recipes/englishparsing.aspx

F. Och. 2003. Minimum error rate training in statistical machine translation. *The 41th Annual meeting of the Association for Computational Linguistics* (ACL), 311-318.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. *The 40th Annual meeting of the Association for Computational Linguistics* (ACL), 311-318.

B. Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249-276.

S. Wang et al. 2005. Exploiting syntactic, semantic and lexical regularities in language modeling via directed Markov random fields. *The 22nd International Conference on Machine Learning* (ICML), 953-960.

S. Wang et al. 2006. Stochastic analysis of lexical and semantic enhanced structural language model. *The 8th International Colloquium on Grammatical Inference* (ICGI), 97-111.

K. Yamada and K. Knight. 2001. A syntax-based statistical translation model. *The 39th Annual Conference on Association of Computational Linguistics* (ACL), 1067-1074.

W. Zangwill. 1969. *Nonlinear Programming: A Unified Approach*. Prentice-Hall.

Y. Zhang, A. Hildebrand and S. Vogel. 2006. Distributed language modeling for N-best list re-ranking. *The 2006 Conference on Empirical Methods in Natural Language Processing* (EMNLP), 216-223.

Y. Zhang, 2008. Structured language models for statistical machine translation. *Ph.D. dissertation*, CMU.

Goodness: A Method for Measuring Machine Translation Confidence

Nguyen Bach*
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
nbach@cs.cmu.edu

Fei Huang and Yaser Al-Onaizan
IBM T.J. Watson Research Center
1101 Kitchawan Rd
Yorktown Heights, NY 10567, USA
{huangfe, onaizan}@us.ibm.com

Abstract

State-of-the-art statistical machine translation (MT) systems have made significant progress towards producing user-acceptable translation output. However, there is still no efficient way for MT systems to inform users which words are likely translated correctly and how confident it is about the whole sentence. We propose a novel framework to predict word-level and sentence-level MT errors with a large number of novel features. Experimental results show that the MT error prediction accuracy is increased from 69.1 to 72.2 in F-score. The Pearson correlation between the proposed confidence measure and the human-targeted translation edit rate (HTER) is 0.6. Improvements between 0.4 and 0.9 TER reduction are obtained with the n-best list reranking task using the proposed confidence measure. Also, we present a visualization prototype of MT errors at the word and sentence levels with the objective to improve post-editor productivity.

1 Introduction

State-of-the-art Machine Translation (MT) systems are making progress to generate more usable translation outputs. In particular, statistical machine translation systems (Koehn et al., 2007; Bach et al., 2007; Shen et al., 2008) have advanced to a state that the translation quality for certain language pairs (e.g. Spanish-English, French-English, Iraqi-English) in certain domains (e.g. broadcasting news, force-protection, travel) is acceptable to users.

However, a remaining open question is how to predict confidence scores for machine translated words and sentences. An MT system typically returns the best translation candidate from its search space, but still has no reliable way to inform users which word is likely to be correctly translated and how confident it is about the whole sentence. Such information is vital to realize the utility of machine translation in many areas. For example, a post-editor would like to quickly identify which sentences might be incorrectly translated and in need of correction. Other areas, such as cross-lingual question-answering, information extraction and retrieval, can also benefit from the confidence scores of MT output. Finally, even MT systems can leverage such information to do n-best list reranking, discriminative phrase table and rule filtering, and constraint decoding (Hildebrand and Vogel, 2008).

Numerous attempts have been made to tackle the confidence estimation problem. The work of Blatz et al. (2004) is perhaps the best known study of sentence and word level features and their impact on translation error prediction. Along this line of research, improvements can be obtained by incorporating more features as shown in (Quirk, 2004; Sanchis et al., 2007; Raybaud et al., 2009; Specia et al., 2009). Soricut and Echihabi (2010) developed regression models which are used to predict the expected BLEU score of a given translation hypothesis. Improvement also can be obtained by using target part-of-speech and null dependency link in a MaxEnt classifier (Xiong et al., 2010). Ueffing and Ney (2007) introduced word posterior probabilities (WPP) features and applied them in the n-best list reranking. From the usability point of view, back-translation is a tool to help users to assess the accuracy level of MT output (Bach et al., 2007). Literally, it translates backward the MT output into the source language to see whether the output of backward translation matches the original source sentence.

However, previous studies had a few shortcomings. First, source-side features were not extensively investigated. Blatz et al.(2004) only investigated source n-gram frequency statistics and source language model features, while other work mainly focused on target side features. Second, previous work attempted to incorporate more features but faced scalability issues, i.e., to train many features we need many training examples and to train discriminatively we need to search through all possible translations of each training example. Another issue of previous work was that they are all trained with BLEU/TER score computing against

* Work done during an internship at IBM T.J. Watson Research Center

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 211–219,
Portland, Oregon, June 19-24, 2011.

the translation references which is different from predicting the human-targeted translation edit rate (HTER) which is crucial in post-editing applications (Snover et al., 2006; Papineni et al., 2002). Finally, the back-translation approach faces a serious issue when forward and backward translation models are symmetric. In this case, back-translation will not be very informative to indicate forward translation quality.

In this paper, we predict error types of each word in the MT output with a confidence score, extend it to the sentence level, then apply it to n-best list reranking task to improve MT quality, and finally design a visualization prototype. We try to answer the following questions:

- Can we use a rich feature set such as source-side information, alignment context, and dependency structures to improve error prediction performance?
- Can we predict more translation error types i.e substitution, insertion, deletion and shift?
- How good do our prediction methods correlate with human correction?
- Do confidence measures help the MT system to select a better translation?
- How confidence score can be presented to improve end-user perception?

In Section 2, we describe the models and training method for the classifier. We describe novel features including source-side, alignment context, and dependency structures in Section 3. Experimental results and analysis are reported in Section 4. Section 5 and 6 present applications of confidence scores.

2 Confidence Measure Model

2.1 Problem setting

Confidence estimation can be viewed as a sequential labelling task in which the word sequence is MT output and word labels can be $Bad/Good$ or $Insertion/Substitution/Shift/Good$. We first estimate each individual word confidence and extend it to the whole sentence. Arabic text is fed into an Arabic-English SMT system and the English translation outputs are corrected by humans in two phases. In phase one, a bilingual speaker corrects the MT system translation output. In phase two, another bilingual speaker does quality checking for the correction done in phase one. If bad corrections were spotted, they correct them again. In this paper we use the final correction data from phase two as the reference thus HTER can be used as an evaluation metric. We have 75 thousand sentences with 2.4 million words in total from the human correction process described above.

We obtain training labels for each word by performing TER alignment between MT output and the phase-two human correction. From TER alignments we observed that out of total errors are 48% substitution, 28% deletion, 13% shift, and 11% insertion errors. Based on the alignment, each word produced by the MT system has a label: good, insertion, substitution and shift. Since a deletion error occurs when it only appears in the reference translation, not in the MT output, our model will not predict deletion errors in the MT output.

2.2 Word-level model

In our problem, a training instance is a word from MT output, and its label when the MT sentence is aligned with the human correction. Given a training instance x, y is the true label of x; f stands for its feature vector $f(x, y)$; and w is feature weight vector. We define a feature-rich classifier $score(x, y)$ as follow

$$score(x, y) = w.f(x, y) \tag{1}$$

To obtain the label, we choose the class with the highest score as the predicted label for that data instance. To learn optimized weights, we use the Margin Infused Relaxed Algorithm or MIRA (Crammer and Singer, 2003; McDonald et al., 2005) which is an online learner closely related to both the support vector machine and perceptron learning framework. MIRA has been shown to provide state-of-the-art performance for sequential labelling task (Rozenfeld et al., 2006), and is also able to provide an efficient mechanism to train and optimize MT systems with lots of features (Watanabe et al., 2007; Chiang et al., 2009). In general, weights are updated at each step time t according to the following rule:

$$\begin{aligned} & w_{t+1} = \arg\min_{w_{t+1}} ||w_{t+1} - w_t|| \\ \text{s.t.} \quad & score(x, y) \geq score(x, y') + L(y, y') \end{aligned} \tag{2}$$

where $L(y, y')$ is a measure of the loss of using y' instead of the true label y. In this problem $L(y, y')$ is 0-1 loss function. More specifically, for each instance x_i in the training data at a time t we find the label with the highest score:

$$y' = \arg\max_{y} score(x_i, y) \tag{3}$$

the weight vector is updated as follow

$$w_{t+1} = w_t + \tau(f(x_i, y) - f(x_i, y')) \tag{4}$$

τ can be interpreted as a step size; when τ is a large number we want to update our weights aggressively, otherwise weights are updated conservatively.

$$\begin{aligned} \tau &= \max(0, \alpha) \\ \alpha &= \min\left\{C, \frac{L(y,y') - (score(x_i,y) - score(x_i,y'))}{||f(x_i,y) - f(x_i,y')||_2^2}\right\} \end{aligned} \tag{5}$$

where C is a positive constant used to cap the maximum possible value of τ. In practice, a cut-off threshold n is the parameter which decides the number of features kept (whose occurrence is at least n) during

training. Note that MIRA is sensitive to constant C, the cut-off feature threshold n, and the number of iterations. The final weight is typically normalized by the number of training iterations and the number of training instances. These parameters are tuned on a development set.

2.3 Sentence-level model

Given the feature sets and optimized weights, we use the Viterbi algorithm to find the best label sequence. To estimate the confidence of a sentence S we rely on the information from the forward-backward inference. One approach is to directly use the conditional probabilities of the whole sequence. However, this quantity is the confidence measure for the label sequence predicted by the classifier and it does not represent the goodness of the whole MT output. Another more appropriated method is to use the marginal probability of $Good$ label which can be defined as follow:

$$p(y_i = Good|S) = \frac{\alpha(y_i|S)\beta(y_i|S)}{\sum_j \alpha(y_j|S)\beta(y_j|S)} \quad (6)$$

$p(y_i = Good|S)$ is the marginal probability of label $Good$ at position i given the MT output sentence S. $\alpha(y_i|S)$ and $\beta(y_i|S)$ are forward and backward values. Our confidence estimation for a sentence S of k words is defined as follow

$$goodness(S) = \frac{\sum_{i=1}^{k} p(y_i = Good|S)}{k} \quad (7)$$

$goodness(S)$ is ranging between 0 and 1, where 0 is equivalent to an absolutely wrong translation and 1 is a perfect translation. Essentially, $goodness(S)$ is the arithmetic mean which represents the goodness of translation per word in the whole sentence.

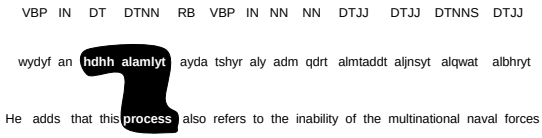
3 Confidence Measure Features

Features are generated from feature types: abstract templates from which specific features are instantiated. Features sets are often parameterized in various ways. In this section, we describe three new feature sets introduced on top of our baseline classifier which has WPP and target POS features (Ueffing and Ney, 2007; Xiong et al., 2010).

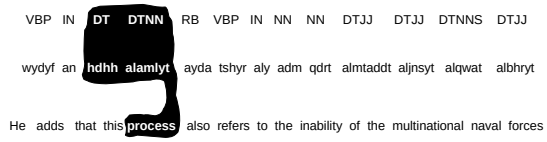
3.1 Source-side features

From MT decoder log, we can track which source phrases generate target phrases. Furthermore, one can infer the alignment between source and target words within the phrase pair using simple aligners such as IBM Model-1 alignment.

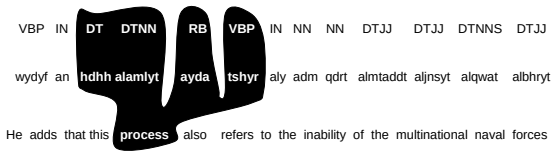
Source phrase features: These features are designed to capture the likelihood that source phrase and target word co-occur with a given error label. The intuition behind them is that if a large percentage of the source phrase and target have often been seen together with the same label, then the produced target word should have this label in the future. Figure 1a illustrates this feature template where the first line is source POS tags, the second line is the Buckwalter romanized source Arabic sequence, and the third line is MT output. The source phrase feature is defined as follow



(a) Source phrase



(b) Source POS



(c) Source POS and phrase in right context

Figure 1: Source-side features.

$$f_{102}(process) = \begin{cases} 1 & \text{if source-phrase="}\textit{hdhh alamlyt}\text{"} \\ 0 & \text{otherwise} \end{cases}$$

Source POS: Source phrase features might be susceptible to sparseness issues. We can generalize source phrases based on their POS tags to reduce the number of parameters. For example, the example in Figure 1a is generalized as in Figure 1b and we have the following feature:

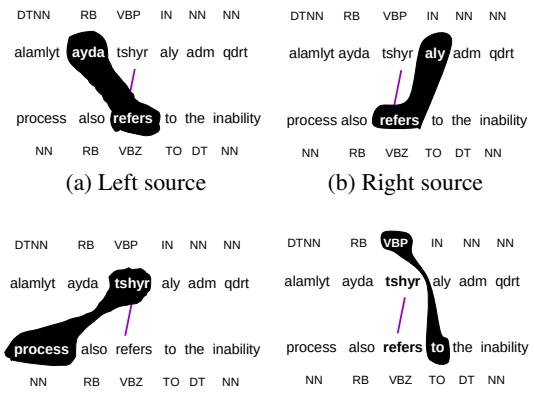
$$f_{103}(process) = \begin{cases} 1 & \text{if source-POS="}\textit{ DT DTNN }\text{"} \\ 0 & \text{otherwise} \end{cases}$$

Source POS and phrase context features: This feature set allows us to look at the surrounding context of the source phrase. For example, in Figure 1c we have "*hdhh alamlyt*" generates "*process*". We also have other information such as on the right hand side the next two phrases are "*ayda*" and "*tshyr*" or the sequence of source target POS on the right hand side is "*RB VBP*". An example of this type of feature is

$$f_{104}(process) = \begin{cases} 1 & \text{if source-POS-context="}\textit{ RB VBP }\text{"} \\ 0 & \text{otherwise} \end{cases}$$

3.2 Alignment context features

The IBM Model-1 feature performed relatively well in comparison with the WPP feature as shown by Blatz et al. (2004). In our work, we incorporate not only the



(a) Left source (b) Right source

(c) Left target (d) Source POS & right target

Figure 2: Alignment context features.

IBM Model-1 feature but also the surrounding alignment context. The key intuition is that collocation is a reliable indicator for judging if a target word is generated by a particular source word (Huang, 2009). Moreover, the IBM Model-1 feature was already used in several steps of a translation system such as word alignment, phrase extraction and scoring. Also the impact of this feature alone might fade away when the MT system is scaled up.

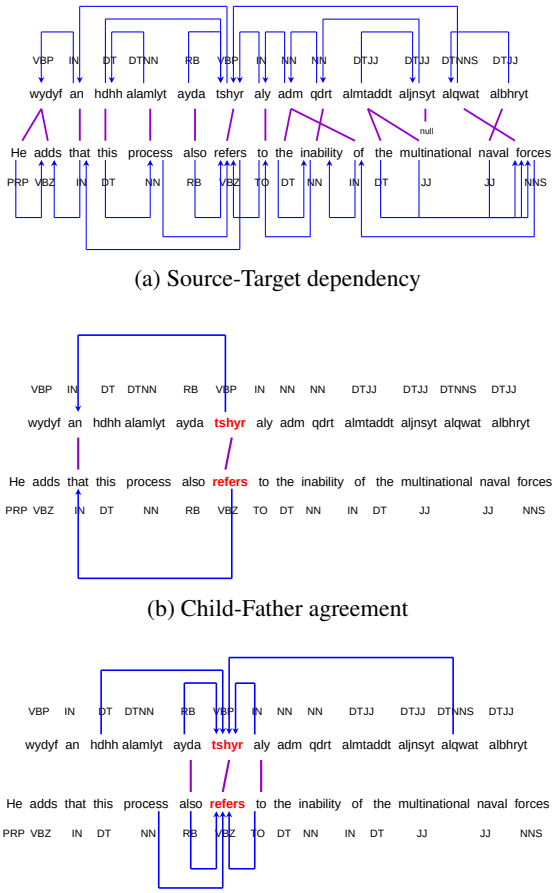
We obtain word-to-word alignments by applying IBM Model-1 to bilingual phrase pairs that generated the MT output. The IBM Model-1 assumes one target word can only be aligned to one source word. Therefore, given a target word we can always identify which source word it is aligned to.

Source alignment context feature: We anchor the target word and derive context features surrounding its source word. For example, in Figure 2a and 2b we have an alignment between "*tshyr*" and "*refers*" The source contexts "*tshyr*" with a window of one word are "*ayda*" to the left and "*aly*" to the right.

Target alignment context feature: Similar to source alignment context features, we anchor the source word and derive context features surrounding the aligned target word. Figure 2c shows a left target context feature of word "*refers*". Our features are derived from a window of four words.

Combining alignment context with POS tags: Instead of using lexical context we have features to look at source and target POS alignment context. For instance, the feature in Figure 2d is

$$f_{141}(refers) = \begin{cases} 1 & \text{if source-POS = "}VBP\text{"} \\ & \text{and target-context = "}to\text{"} \\ 0 & \text{otherwise} \end{cases}$$



(a) Source-Target dependency

(b) Child-Father agreement

(c) Children agreement

Figure 3: Dependency structures features.

3.3 Source and target dependency structure features

The contextual and source information in the previous sections only take into account surface structures of source and target sentences. Meanwhile, dependency structures have been extensively used in various translation systems (Shen et al., 2008; Ma et al., 2008; Bach et al., 2009). The adoption of dependency structures might enable the classifier to utilize deep structures to predict translation errors. Source and target structures are unlikely to be isomorphic as shown in Figure 3a. However, we expect some high-level linguistic structures are likely to transfer across certain language pairs. For example, prepositional phrases (PP) in Arabic and English are similar in a sense that PPs generally appear at the end of the sentence (after all the verbal arguments) and to a lesser extent at its beginning (Habash and Hu, 2009). We use the Stanford parser to obtain dependency trees and POS tags (Marneffe et al., 2006).

Child-Father agreement: The motivation is to take advantage of the long distance dependency relations between source and target words. Given an alignment between a source word s_i and a target word t_j. A child-

father agreement exists when s_k is aligned to t_l, where s_k and t_l are father of s_i and t_j in source and target dependency trees, respectively. Figure 3b illustrates that "*tshyr*" and "*refers*" have a child-father agreement. To verify our intuition, we analysed 243K words of manual aligned Arabic-English bitext. We observed 29.2% words having child-father agreements. In term of structure types, we found 27.2% of copula verb and 30.2% prepositional structures, including object of a preposition, prepositional modifier, and prepositional complement, are having child-father agreements.

Children agreement: In the child-father agreement feature we look up in the dependency tree, however, we also can look down to the dependency tree with a similar motivation. Essentially, given an alignment between a source word s_i and a target word t_j, how many children of s_i and t_j are aligned together? For example, "*tshyr*" and "*refers*" have 2 aligned children which are "*ayda-also*" and "*aly-to*" as shown in Figure 3c.

4 Experiments

4.1 Arabic-English translation system

The SMT engine is a phrase-based system similar to the description in (Tillmann, 2006), where various features are combined within a log-linear framework. These features include source-to-target phrase translation score, source-to-target and target-to-source word-to-word translation scores, language model score, distortion model scores and word count. The training data for these features are 7M Arabic-English sentence pairs, mostly newswire and UN corpora released by LDC. The parallel sentences have word alignment automatically generated with HMM and MaxEnt word aligner (Ge, 2004; Ittycheriah and Roukos, 2005). Bilingual phrase translations are extracted from these word-aligned parallel corpora. The language model is a 5-gram model trained on roughly 3.5 billion English words.

Our training data contains 72k sentences Arabic-English machine translation with human corrections which include of 2.2M words in newswire and weblog domains. We have a development set of 2,707 sentences, 80K words (dev); an unseen test set of 2,707 sentences, 79K words (test). Feature selection and parameter tuning has been done on the development set in which we experimented values of C, n and iterations in range of [0.5:10], [1:5], and [50:200] respectively. The final MIRA classifier was trained by using pocket crf toolkit[1] with 100 iterations, hyper-parameter C was 5 and cut-off feature threshold n was 1.

We use precision (P), recall (R) and F-score (F) to evaluate the classifier performance and they are computed as follow:

$$\begin{aligned} P &= \frac{\text{the number of correctly tagged labels}}{\text{the number of tagged labels}} \\ R &= \frac{\text{the number of correctly tagged labels}}{\text{the number of reference labels}} \\ F &= \frac{2*P*R}{P+R} \end{aligned} \quad (8)$$

4.2 Contribution of feature sets

We designed our experiments to show the impact of each feature separately as well as their cumulative impact. We trained two types of classifiers to predict the error type of each word in MT output, namely Good/Bad with a binary classifier and Good/Insertion/Substitution/Shift with a 4-class classifier. Each classifier is trained with different feature sets as follow:

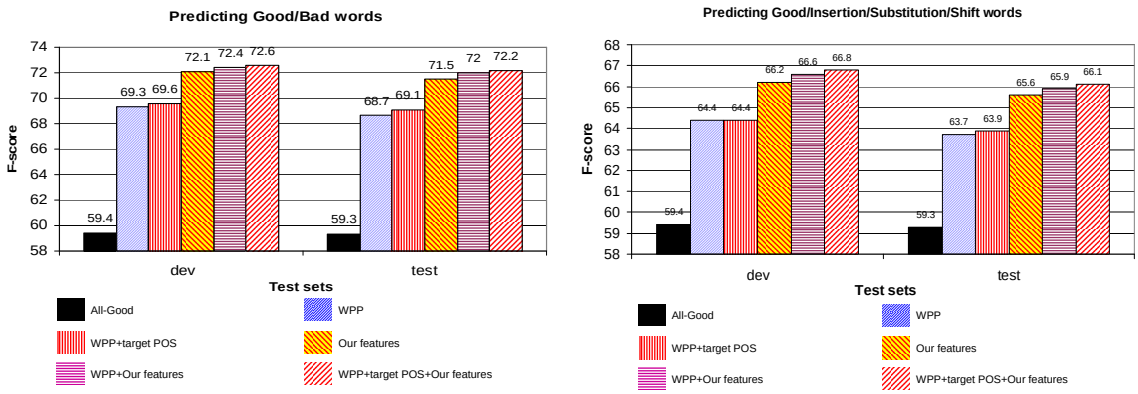
- WPP: we reimplemented WPP calculation based on n-best lists as described in (Ueffing and Ney, 2007).
- WPP + target POS: only WPP and target POS features are used. This is a similar feature set used by Xiong et al. (2010).
- Our features: the classifier has source side, alignment context, and dependency structure features; WPP and target POS features are excluded.
- WPP + our features: adding our features on top of WPP.
- WPP + target POS + our features: using all features.

	binary		4-class	
	dev	test	dev	test
WPP	69.3	68.7	64.4	63.7
+ source side	72.1	**71.6**	66.2	**65.7**
+ alignment context	71.4	70.9	65.7	65.3
+ dependency structures	69.9	69.5	64.9	64.3
WPP+ target POS	69.6	69.1	64.4	63.9
+ source side	72.3	**71.8**	66.3	**65.8**
+ alignment context	71.9	71.2	66	65.6
+ dependency structures	70.4	70	65.1	64.4

Table 1: Contribution of different feature sets measure in F-score.

To evaluate the effectiveness of each feature set, we apply them on two different baseline systems: using WPP and WPP+target POS, respectively. We augment each baseline with our feature sets separately. Table 1 shows the contribution in F-score of our proposed feature sets. Improvements are consistently obtained when combining the proposed features with baseline features. Experimental results also indicate that source-side information, alignment context and dependency

[1] http://pocket-crf-1.sourceforge.net/



(a) Binary (b) 4-class

Figure 4: Performance of binary and 4-class classifiers trained with different feature sets on the development and unseen test sets.

structures have unique and effective levers to improve the classifier performance. Among the three proposed feature sets, we observe the source side information contributes the most gain, which is followed by the alignment context and dependency structure features.

4.3 Performance of classifiers

We trained several classifiers with our proposed feature sets as well as baseline features. We compare their performances, including a naive baseline All-Good classifier, in which all words in the MT output are labelled as good translations. Figure 4 shows the performance of different classifiers trained with different feature sets on development and unseen test sets. On the unseen test set our proposed features outperform WPP and target POS features by 2.8 and 2.4 absolute F-score respectively. Improvements of our features are consistent in development and unseen sets as well as in binary and 4-class classifiers. We reach the best performance by combining our proposed features with WPP and target POS features. Experiments indicate that the gaps in F-score between our best system with the naive All-Good system is 12.9 and 6.8 in binary and 4-class cases, respectively. Table 2 presents precision, recall, and F-score of individual class of the best binary and 4-class classifiers. It shows that *Good* label is better predicted than other labels, meanwhile, $Substitution$ is generally easier to predict than $Insertion$ and $Shift$.

4.4 Correlation between Goodness and HTER

We estimate sentence level confidence score based on Equation 7. Figure 5 illustrates the correlation between our proposed $goodness$ sentence level confidence score and the human-targeted translation edit rate (HTER). The Pearson correlation between $goodness$ and HTER is 0.6, while the correlation of WPP and HTER is 0.52. This experiment shows that $goodness$ has a large correlation with HTER. The black bar is the linear regression line. Blue and red bars are thresholds used to visualize good and bad sentences respectively. We also experimented $goodness$ computation in Equation 7 using geometric mean and harmonic mean; their Pearson correlation values are 0.5 and 0.35 respectively.

	Label	P	R	F
Binary	Good	74.7	80.6	77.5
	Bad	68	60.1	63.8
4-class	Good	70.8	87	78.1
	Insertion	37.5	16.9	23.3
	Substitution	57.8	44.9	50.5
	Shift	35.2	14.1	20.1

Table 2: Detailed performance in precision, recall and F-score of binary and 4-class classifiers with WPP+target POS+Our features on the unseen test set.

5 Improving MT quality with N-best list reranking

Experiments reporting in Section 4 indicate that the proposed confidence measure has a high correlation with HTER. However, it is not very clear if the core MT system can benefit from confidence measure by providing better translations. To investigate this question we present experimental results for the n-best list reranking task.

The MT system generates top n hypotheses and for each hypothesis we compute sentence-level confidence scores. The best candidate is the hypothesis with highest confidence score. Table 3 shows the performance of reranking systems using $goodness$ scores from our best classifier in various n-best sizes. We obtained 0.7 TER reduction and 0.4 BLEU point improvement on the development set with a 5-best list. On the unseen test, we obtained 0.6 TER reduction and 0.2 BLEU point improvement. Although, the improvement of BLEU score

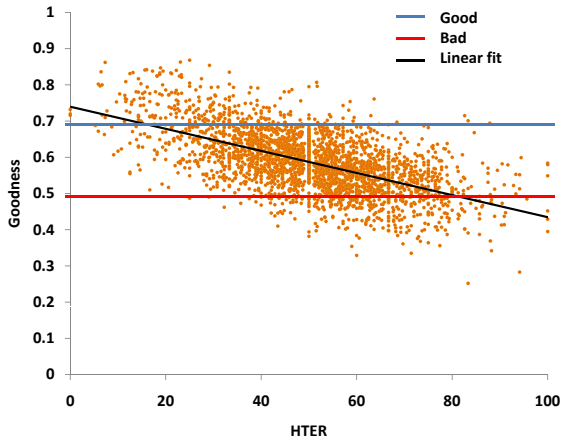


Figure 5: Correlation between Goodness and HTER.

	Dev		Test	
	TER	BLEU	TER	BLEU
Baseline	49.9	31.0	50.2	30.6
2-best	49.5	31.4	49.9	30.8
5-best	**49.2**	**31.4**	**49.6**	**30.8**
10-best	49.2	31.2	49.5	30.8
20-best	49.1	31.0	49.3	30.7
30-best	49.0	31.0	49.3	30.6
40-best	49.0	31.0	49.4	30.5
50-best	49.1	30.9	49.4	30.5
100-best	49.0	30.9	49.3	30.5

Table 3: Reranking performance with *goodness* score.

is not obvious, TER reductions are consistent in both development and unseen sets. Figure 6 shows the improvement of reranking with *goodness* score. Besides, the figure illustrates the upper and lower bound performances with TER metric in which the lower bound is our baseline system and the upper bound is the best hypothesis in a given n-best list. Oracle scores of each n-best list are computed by choosing the translation candidate with lowest TER score.

6 Visualizing translation errors

Besides the application of confidence score in the n-best list reranking task, we propose a method to visualize translation error using confidence scores. Our purpose is to visualize word and sentence-level confidence scores with the following objectives 1) easy for spotting translations errors; 2) simple and intuitive; and 3) helpful for post-editing productivity. We define three categories of translation quality (good/bad/decent) on both word and sentence level. On word level, the marginal probability of good label is used to visualize translation errors as follow:

$$L_i = \begin{cases} good & \text{if } p(y_i = Good|S) \geq 0.8 \\ bad & \text{if } p(y_i = Good|S) \leq 0.45 \\ decent & \text{otherwise} \end{cases}$$

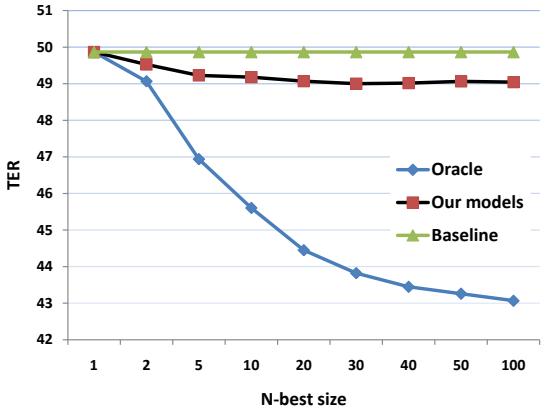


Figure 6: A comparison between reranking and oracle scores with different n-best size in TER metric on the development set.

On sentence level, the goodness score is used as follow:

$$L_S = \begin{cases} good & \text{if } goodness(S) \geq 0.7 \\ bad & \text{if } goodness(S) \leq 0.5 \\ decent & \text{otherwise} \end{cases}$$

	Choices	Intention
Font size	big	bad
	small	good
	medium	decent
Colors	red	bad
	black	good
	orange	decent

Table 4: Choices of layout

Different font sizes and colors are used to catch the attention of post-editors whenever translation errors are likely to appear as shown in Table 4. Colors are applied on word level, while font size is applied on both word and sentence level. The idea of using font size and colour to visualize translation confidence is similar to the idea of using tag/word cloud to describe the content of websites[2]. The reason we are using big font size and red color is to attract post-editors' attention and help them find translation errors quickly. Figure 7 shows an example of visualizing confidence scores by font size and colours. It shows that "*not to deprive yourself*", displayed in big font and red color, is likely to be bad translations. Meanwhile, other words, such as "*you*", "*different*", "*from*", and "*assimilation*", displayed in small font and black color, are likely to be good translation. Medium font and orange color words are decent translations.

[2]http://en.wikipedia.org/wiki/Tag_cloud

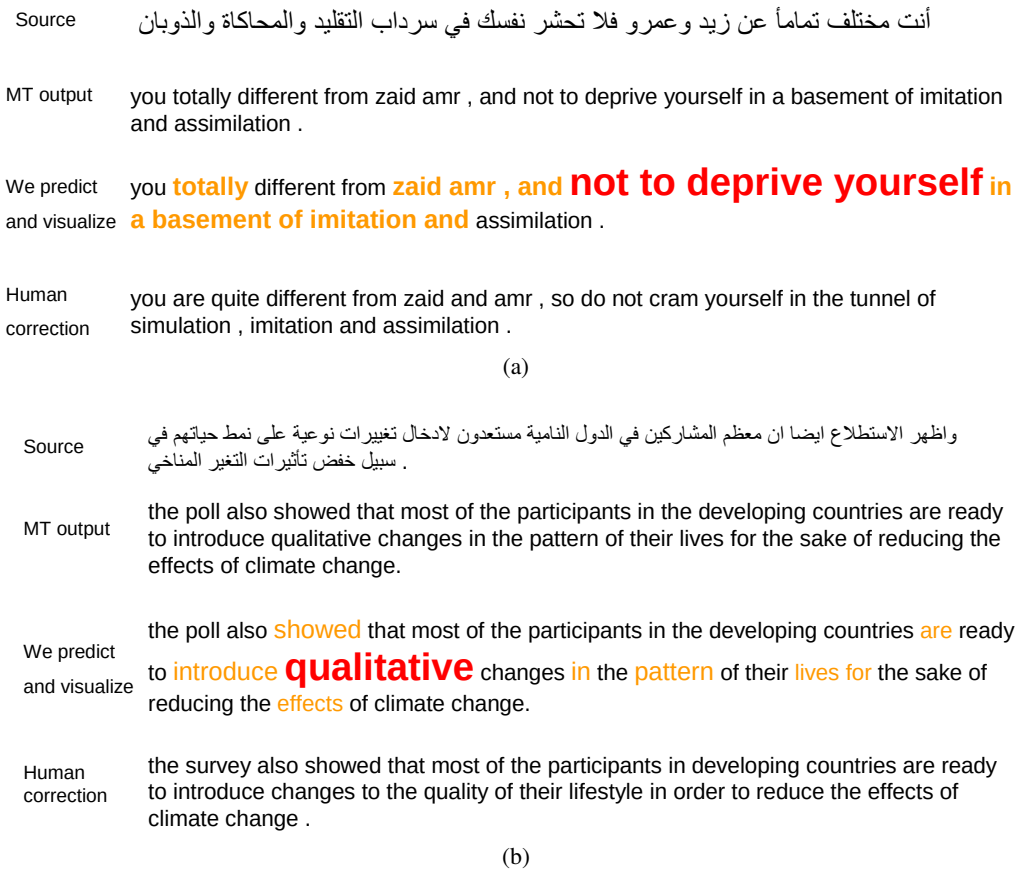


Figure 7: MT errors visualization based on confidence scores.

7 Conclusions

In this paper we proposed a method to predict confidence scores for machine translated words and sentences based on a feature-rich classifier using linguistic and context features. Our major contributions are three novel feature sets including source side information, alignment context, and dependency structures. Experimental results show that by combining the source side information, alignment context, and dependency structure features with word posterior probability and target POS context (Ueffing & Ney 2007; Xiong et al., 2010), the MT error prediction accuracy is increased from 69.1 to 72.2 in F-score. Our framework is able to predict error types namely insertion, substitution and shift. The Pearson correlation with human judgement increases from 0.52 to 0.6. Furthermore, we show that the proposed confidence scores can help the MT system to select better translations and as a result improvements between 0.4 and 0.9 TER reduction are obtained. Finally, we demonstrate a prototype to visualize translation errors.

This work can be expanded in several directions. First, we plan to apply confidence estimation to perform a second-pass constraint decoding. After the first pass decoding, our confidence estimation model can label which word is likely to be correctly translated. The second-pass decoding utilizes the confidence information to constrain the search space and hopefully can find a better hypothesis than in the first pass. This idea is very similar to the multi-pass decoding strategy employed by speech recognition engines. Moreover, we also intend to perform a user study on our visualization prototype to see if it increases the productivity of post-editors.

Acknowledgements

We would like to thank Christoph Tillmann and the IBM machine translation team for their supports. Also, we would like to thank anonymous reviewers, Qin Gao, Joy Zhang, and Stephan Vogel for their helpful comments.

References

Nguyen Bach, Matthias Eck, Paisarn Charoenpornsawat, Thilo Khler, Sebastian Stker, ThuyLinh Nguyen, Roger Hsiao, Alex Waibel, Stephan Vogel, Tanja Schultz, and Alan Black. 2007. The CMU TransTac 2007 Eyes-free and Hands-free Two-way Speech-to-Speech Translation System. In *Proceedings of the IWSLT'07*, Trento, Italy.

Nguyen Bach, Qin Gao, and Stephan Vogel. 2009. Source-side dependency tree reordering models with subtree movements and constraints. In *Proceedings of the MTSummit-XII*, Ottawa, Canada, August. International Association for Machine Translation.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In *The JHU Workshop Final Report*, Baltimore, Maryland, USA, April.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of HLT-ACL*, pages 218–226, Boulder, Colorado, June. Association for Computational Linguistics.

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

Niyu Ge. 2004. Max-posterior HMM alignment for machine translation. In *Presentation given at DARPA/TIDES NIST MT Evaluation workshop*.

Nizar Habash and Jun Hu. 2009. Improving arabic-chinese statistical machine translation using english as pivot language. In *Proceedings of the 4th Workshop on Statistical Machine Translation*, pages 173–181, Morristown, NJ, USA. Association for Computational Linguistics.

Almut Silja Hildebrand and Stephan Vogel. 2008. Combination of machine translation systems via hypothesis selection from combined n-best lists. In *Proceedings of the 8th Conference of the AMTA*, pages 254–261, Waikiki, Hawaii, October.

Fei Huang. 2009. Confidence measure for word alignment. In *Proceedings of the ACL-IJCNLP '09*, pages 932–940, Morristown, NJ, USA. Association for Computational Linguistics.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *Proceedings of the HTL-EMNLP'05*, pages 89–96, Morristown, NJ, USA. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL'07*, pages 177–180, Prague, Czech Republic, June.

Yanjun Ma, Sylwia Ozdowska, Yanli Sun, and Andy Way. 2008. Improving word alignment using syntactic dependencies. In *Proceedings of the ACL-08: HLT SSST-2*, pages 69–77, Columbus, OH.

Marie-Catherine Marneffe, Bill MacCartney, and Christopher Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC'06*, Genoa, Italy.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Flexible text segmentation with structured multilabel classification. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 987–994, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL'02*, pages 311–318, Philadelphia, PA, July.

Chris Quirk. 2004. Training a sentence-level machine translation confidence measure. In *Proceedings of the 4th LREC*.

Sylvain Raybaud, Caroline Lavecchia, David Langlois, and Kamel Smaili. 2009. Error detection for statistical machine translation using linguistic features. In *Proceedings of the 13th EAMT*, Barcelona, Spain, May.

Binyamin Rozenfeld, Ronen Feldman, and Moshe Fresko. 2006. A systematic cross-comparison of sequence classifiers. In *Proceedings of the SDM*, pages 563–567, Bethesda, MD, USA, April.

Alberto Sanchis, Alfons Juan, and Enrique Vidal. 2007. Estimation of confidence measures for machine translation. In *Proceedings of the MT Summit XI*, Copenhagen, Denmark.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585, Columbus, Ohio, June. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of AMTA'06*, pages 223–231, August.

Radu Soricut and Abdessamad Echihabi. 2010. Trustrank: Inducing trust in automatic translations via ranking. In *Proceedings of the 48th ACL*, pages 612–621, Uppsala, Sweden, July. Association for Computational Linguistics.

Lucia Specia, Zhuoran Wang, Marco Turchi, John Shawe-Taylor, and Craig Saunders. 2009. Improving the confidence of machine translation quality estimates. In *Proceedings of the MT Summit XII*, Ottawa, Canada.

Christoph Tillmann. 2006. Efficient dynamic programming search algorithms for phrase-based SMT. In *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, pages 9–16, Morristown, NJ, USA. Association for Computational Linguistics.

Nicola Ueffing and Hermann Ney. 2007. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33(1):9–40.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of the EMNLP-CoNLL*, pages 764–773, Prague, Czech Republic, June. Association for Computational Linguistics.

Deyi Xiong, Min Zhang, and Haizhou Li. 2010. Error detection for statistical machine translation using linguistic features. In *Proceedings of the 48th ACL*, pages 604–611, Uppsala, Sweden, July. Association for Computational Linguistics.

MEANT: An inexpensive, high-accuracy, semi-automatic metric for evaluating translation utility via semantic frames

Chi-kiu Lo and **Dekai Wu**
HKUST
Human Language Technology Center
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
{jackielo,dekai}@cs.ust.hk

Abstract

We introduce a novel semi-automated metric, MEANT, that assesses translation utility by matching semantic role fillers, producing scores that correlate with human judgment as well as HTER but at much lower labor cost. As machine translation systems improve in lexical choice and fluency, the shortcomings of widespread n-gram based, fluency-oriented MT evaluation metrics such as BLEU, which fail to properly evaluate adequacy, become more apparent. But more accurate, non-automatic adequacy-oriented MT evaluation metrics like HTER are highly labor-intensive, which bottlenecks the evaluation cycle. We first show that when using untrained monolingual readers to annotate semantic roles in MT output, the non-automatic version of the metric HMEANT achieves a 0.43 correlation coefficient with human adequacy judgments at the sentence level, far superior to BLEU at only 0.20, and equal to the far more expensive HTER. We then replace the human semantic role annotators with automatic shallow semantic parsing to further automate the evaluation metric, and show that even the semi-automated evaluation metric achieves a 0.34 correlation coefficient with human adequacy judgment, which is still about 80% as closely correlated as HTER despite an even lower labor cost for the evaluation procedure. The results show that our proposed metric is significantly better correlated with human judgment on adequacy than current widespread automatic evaluation metrics, while being much more cost effective than HTER.

1 Introduction

In this paper we show that evaluating machine translation by assessing the translation accuracy of each argument in the semantic role framework correlates with human judgment on translation *adequacy* as well as HTER, at a significantly lower labor cost. The correlation of this new metric, MEANT, with human judgment is far superior to BLEU and other automatic n-gram based evaluation metrics.

We argue that BLEU (Papineni *et al.*, 2002) and other automatic n-gram based MT evaluation metrics do not adequately capture the similarity in meaning between the machine translation and the reference translation—which, ultimately, is essential for MT output to be useful. N-gram based metrics assume that "good" translations tend to share the same lexical choices as the reference translations. While BLEU score performs well in capturing the translation fluency, Callison-Burch *et al.* (2006) and Koehn and Monz (2006) report cases where BLEU strongly disagree with human judgment on translation quality. The underlying reason is that lexical similarity does not adequately reflect the similarity in meaning. As MT systems improve, the shortcomings of the n-gram based evaluation metrics are becoming more apparent. State-of-the-art MT systems are often able to output fluent translations that are nearly grammatical and contain roughly the correct words, but still fail to express meaning that is close to the input.

At the same time, although HTER (Snover *et al.*, 2006) is more adequacy-oriented, it is only employed in very large scale MT system evaluation instead of day-to-day research activities. The underlying reason is that it requires rigorously trained human experts to make difficult combinatorial decisions on the minimal number of edits so as to make the MT output convey the same meaning as the reference translation—a highly labor-intensive, costly process that bottlenecks the evaluation cycle.

Instead, with MEANT, we adopt at the outset the principle that a good translation is one that is *useful*, in the sense that human readers may successfully understand at least the basic event structure—"*who* did *what* to *whom*, *when*, *where* and *why*" (Pradhan *et al.*, 2004)—representing the central meaning of the source utterances. It is true that limited tasks might exist for which inadequate translations are still useful. But for meaningful tasks, generally speaking, for a translation to be useful, at least the basic event structure must be correctly understood. Therefore, our objective is to evaluate *translation utility*: from a user's point of view, how well is

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 220–229,
Portland, Oregon, June 19-24, 2011.

the most essential semantic information being captured by machine translation systems?

In this paper, we detail the methodology that underlies MEANT, which extends and implements preliminary directions proposed in (Lo and Wu, 2010a) and (Lo and Wu, 2010b). We present the results of evaluating translation utility by measuring the accuracy within a semantic role labeling (SRL) framework. We show empirically that our proposed SRL based evaluation metric, which uses untrained monolingual humans to annotate semantic frames in MT output, correlates with human adequacy judgments as well as HTER, and far better than BLEU and other commonly used metrics. Finally, we show that replacing the human semantic role labelers with an automatic shallow semantic parser in our proposed metric yields an approximation that is about 80% as closely correlated with human judgment as HTER, at an even lower cost—and is still far better correlated than n-gram based evaluation metrics.

2 Related work

Lexical similarity based metrics BLEU (Papineni *et al.*, 2002) is the most widely used MT evaluation metric despite the fact that a number of large scale meta-evaluations (Callison-Burch *et al.*, 2006; Koehn and Monz, 2006) report cases where it strongly disagree with human judgment on translation accuracy. Other lexical similarity based automatic MT evaluation metrics, like NIST (Doddington, 2002), METEOR (Banerjee and Lavie, 2005), PER (Tillmann *et al.*, 1997), CDER (Leusch *et al.*, 2006) and WER (Nießen *et al.*, 2000), also perform well in capturing translation fluency, but share the same problem that although evaluation with these metrics can be done very quickly at low cost, their underlying assumption—that a "good" translation is one that shares the same lexical choices as the reference translation—is not justified semantically. Lexical similarity does not adequately reflect similarity in meaning. State-of-the-art MT systems are often able to output translations containing roughly the correct words, yet expressing meaning that is not close to that of the input.

We argue that a translation metric that reflects meaning similarity is better based on similarity in semantic structure, rather than simply flat lexical similarity.

HTER (non-automatic) Despite the fact that Human-targeted Translation Edit Rate (HTER) as proposed by Snover *et al.* (2006) shows a high correlation with human judgment on translation adequacy, it is not widely used in day-to-day machine translation evaluation because of its high labor cost. HTER not only requires human experts to understand the meaning expressed in both the reference translation and the machine translation, but also requires them to propose the minimum number of edits to the MT output such that the post-edited MT output conveys the same meaning as the reference translation. Requiring such heavy manual decision making greatly increases the cost of evaluation, bottlenecking the evaluation cycle.

To reduce the cost of evaluation, we aim to reduce any human decisions in the evaluation cycle to be as simple as possible, such that even untrained humans can quickly complete the evaluation. The human decisions should also be defined in a way that can be closely approximated by automatic methods, so that similar objective functions might potentially be used for tuning in MT system development cycles.

Task based metrics (non-automatic) Voss and Tate (2006) proposed a task-based approach to MT evaluation that is in some ways similar in spirit to ours, but rather than evaluating how well people understand the meaning as a whole conveyed by a sentence translation, they measured the recall with which humans can extract *one* of the *who*, *when*, or *where* elements from MT output—and without attaching them to any predicate or frame. A large number of human subjects were instructed to extract only *one* particular type of *wh*-item from each sentence. They evaluated only whether the role fillers were correctly identified, without checking whether the roles were appropriately attached to the correct predicate. Also, the actor, experiencer, and patient were all conflated into the undistinguished *who* role, while other crucial elements, like the action, purpose, manner, were ignored.

Instead, we argue, evaluating meaning similarity should be done by evaluating the semantic structure as a whole: (a) *all* core semantic roles should be checked, and (b) not only should we evaluate the presence of semantic role fillers in isolation, but also their relations to the frames' predicates.

Syntax based metrics Unlike Voss and Tate, Liu and Gildea (2005) proposed a structural approach, but it was based on syntactic rather than semantic structure, and focused on checking the correctness of the role *structure* without checking the correctness of the role *fillers*. Their subtree metric (STM) and headword chain metric (HWC) address the failure of BLEU to evaluate translation *grammaticality*; however, the problem remains that a grammatical translation can achieve a high syntax-based score even if contains meaning errors arising from confusion of semantic roles.

STM was the first proposed metric to incorporate syntactic features in MT evaluation, and STM underlies most other recently proposed syntactic MT evaluation metrics, for example the evaluation metric based on lexical-functional grammar of Owczarzak *et al.* (2008). STM is a precision-based metric that measures what fraction of subtree *structures* are shared between the parse trees of

machine translations and reference translations (averaging over subtrees up to some depth threshold). Unlike Voss and Tate, however, STM does not check whether the role *fillers* are correctly translated.

HWC is similar, but is based on dependency trees containing lexical as well as syntactic information. HWC measures what fraction of headword chains (a sequence of words corresponding to a path in the dependency tree) also appear in the reference dependency tree. This can be seen as a similarity measure on n-grams of dependency chains. Note that the HWC's notion of lexical similarity still requires exact word match.

Although STM-like syntax-based metrics are an improvement over flat lexical similarity metrics like BLEU, they are still more fluency-oriented than adequacy-oriented. Similarity of syntactic rather than semantic structure still inadequately reflects meaning preservation. Moreover, properly measuring translation *utility* requires verifying whether role *fillers* have been correctly translated—verifying only the abstract structures fails to penalize when role fillers are confused.

Semantic roles as features in aggregate metrics Giménez and Màrquez (2007, 2008) introduced ULC, an automatic MT evaluation metric that aggregates many types of features, including several shallow semantic similarity features: semantic role overlapping, semantic role matching, and semantic structure overlapping. Unlike Liu and Gildea (2007) who use discriminative training to tune the weight on each feature, ULC uses uniform weights. Although the metric shows an improved correlation with human judgment of translation quality (Callison-Burch *et al.*, 2007; Giménez and Màrquez, 2007; Callison-Burch *et al.*, 2008; Giménez and Màrquez, 2008), it is not commonly used in large-scale MT evaluation campaigns, perhaps due to its high time cost and/or the difficulty of interpreting its score because of its highly complex combination of many heterogenous types of features.

Specifically, note that the feature based representations of semantic roles used in these aggregate metrics do not actually capture the structural predicate-argument relations. "Semantic structure overlapping" can be seen as the shallow semantic version of STM: it only measures the similarity of the tree structure of the semantic roles, without considering the lexical realization. "Semantic role overlapping" calculates the degree of lexical overlap between semantic roles of the same type in the machine translation and its reference translation, using simple bag-of-words counting; this is then aggregated into an average over all semantic role types. "Semantic role matching" is just like "semantic role overlapping", except that bag-of-words degree of similarity is replaced (rather harshly) by a boolean indicating whether the role fillers are an exact string match. It is important to note that "semantic role overlapping" and "semantic role matching" both use flat feature based representations which do not capture the structural relations in semantic frames, i.e., the predicate-argument relations.

Like system combination approaches, ULC is a vastly more complex aggregate metric compared to widely used metrics like BLEU or STM. We believe it is important to retain a focus on developing *simpler* metrics which not only correlate well with human adequacy judgments, but nevertheless still directly provide *representational transparency* via simple, clear, and transparent scoring schemes that are (a) easily human readable to support error analysis, and (b) potentially directly usable for automatic credit/blame assignment in tuning tree-structured SMT systems. We also believe that to provide a foundation for better design of efficient automated metrics, making use of *humans* for annotating semantic roles and judging the role translation accuracy in MT output is an essential step that should not be bypassed, in order to adequately understand the upper bounds of such techniques.

We agree with Przybocki *et al.* (2010), who observe in the NIST MetricsMaTr 2008 report that "human [adequacy] assessments only pertain to the translations evaluated, and are of no use even to updated translations from the same systems". Instead, we aim for MT evaluation metrics that provide fine-grained scores in a way that also directly reflects interpretable insights on the strengths and weaknesses of MT systems rather than simply replicating human assessments.

3 MEANT: SRL for MT evaluation

A good translation is one from which human readers may successfully understand at least the basic event structure—"*who* did *what* to *whom*, *when*, *where* and *why*" (Pradhan *et al.*, 2004)—which represents the most essential meaning of the source utterances.

MEANT measures this as follows. First, semantic role labeling is performed (either manually or automatically) on both the reference translation and the machine translation. The semantic frame structures thus obtained for the MT output are compared to those in the reference translations, frame by frame, argument by argument. The frame translation accuracy is a weighted sum of the number of correctly translated arguments. Conceptually, MEANT is defined in terms of f-score, with respect to the precision/recall for sentence translation accuracy as calculated by averaging the translation accuracy for all frames in the MT output across the number of frames in the MT output/reference translations. Details are given below.

3.1 Annotating semantic frames

In designing a semantic MT evaluation metric, one important issue that should be addressed is how to evaluate the similarity of meaning objectively and systematically

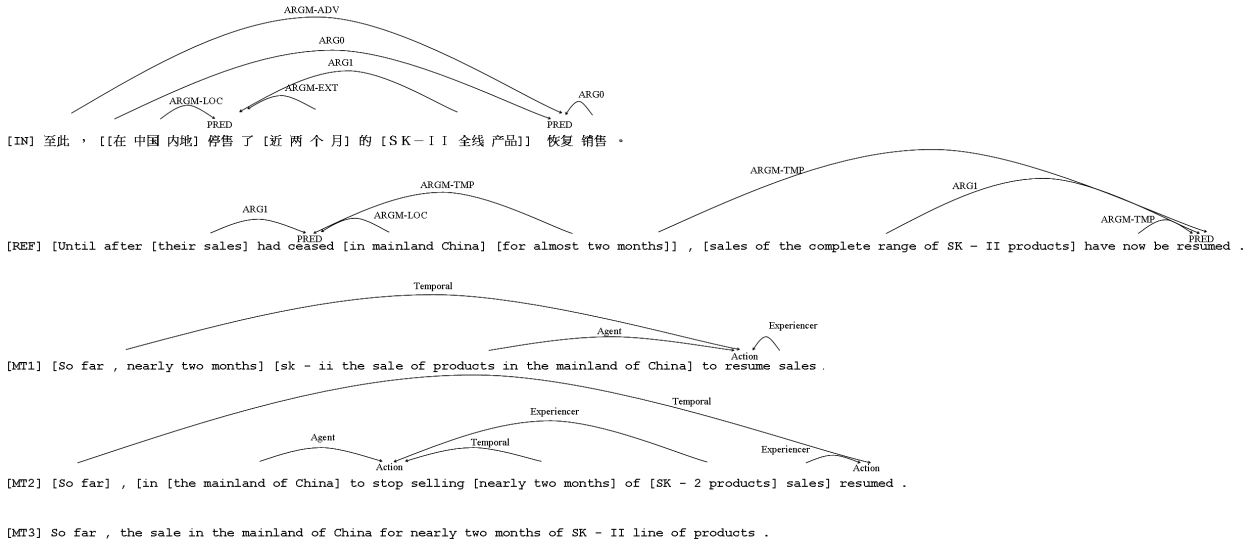


Figure 1: Example of source sentence and reference translation with reconstructed semantic frames in Propbank format and MT output with reconstructed semantic frames by minimal trained human annotators. Following Propbank, there are no semantic frames for MT3 because there is no predicate.

using fine-grained measures. We adopted the Propbank SRL style predicate-argument framework, which captures the basic event structure in a sentence in a way that clearly indicates many strengths and weaknesses of MT. Figure 1 shows the reference translation with reconstructed semantic frames in Propbank format and the corresponding MT output with reconstructed semantic frames by minimal trained human annotators.

3.2 Comparing semantic frames

After annotating the semantic frames, we must determine the translation accuracy for each semantic role filler in the reference and machine translations. Although ultimately it would be nice to do this automatically, it is essential to first understand extremely well the upper bound of accuracy for MT evaluation via semantic frame theory. Thus, instead of resorting to excessively permissive bag-of-words matching or excessively restrictive exact string matching, for the experiments reported here we employed a group of human judges to evaluate the correctness of each role filler translation between the reference and machine translations.

In order to facilitate a finer-grained measurement of utility, the human judges were not only allowed to mark each role filler translation as "correct" or "incorrect", but also "partial". Translations of role fillers are judged "correct" if they express the same meaning as that of the reference translations (or the original source input, in the bilinguals experiment discussed later). Translations may also be judged "partial" if only part of the meaning is correctly translated. Extra meaning in a role filler is not penalized unless it belongs in another role. We also assume that a wrongly translated predicate means that the entire semantic frame is incorrect; therefore, the "correct" and "partial" argument counts are collected only if their associated predicate is correctly translated in the first place.

Table 1 shows an example of SRL annotation of MT1 in Figure 1 by one of the annotators, along with the human judgment on translation accuracy of each argument. The predicate ceased in the reference translation did not match with any predicate annotated in MT1, while the predicate resumed matched with the predicate resume annotated in MT1. All arguments of the untranslated ceased are automatically considered incorrect (with no need to consider each argument individually), under our assumption that a wrongly translated predicate causes the entire event frame to be considered mistranslated. The ARGM-TMP argument, Until after their sales had ceased in mainland China for almost two months, in the reference translation is partially translated to ARGM-TMP argument, So far , nearly two months, in MT1. Similar decisions are made for the ARG1 argument and the other ARGM-TMP argument; now in the reference translation is missing in MT1.

3.3 Quantifying semantic frame match

To quantify the above in a summary metric, we define MEANT in terms of an f-score that balances the precision and recall analysis of the comparative matrices collected from the human judges, as follows.

$C_{i,j}$ = # correct fillers of ARG j for PRED i in MT
$P_{i,j}$ = # partial fillers of ARG j for PRED i in MT
$M_{i,j}$ = total # fillers of ARG j for PRED i in MT
$R_{i,j}$ = total # fillers of ARG j of PRED i in REF

Table 1: SRL annotation of MT1 in Figure 1 and the human judgment of translation accuracy for each argument (see text).

SRL	REF	MT1	Decision
PRED (Action)	ceased	–	no match
PRED (Action)	resumed	resume	match
ARG0 (Agent)	–	sk - ii the sale of products in the mainland of China	incorrect
ARG1 (Experiencer)	sales of complete range of SK - II products	sales	partial
ARGM-TMP (Temporal)	Until after , their sales had ceased in mainland China for almost two months	So far , nearly two months	partial
ARGM-TMP (Temporal)	now	–	incorrect

$$C_{\text{precision}} = \sum_{\text{matched i}} \frac{w_{\text{pred}} + \sum_j w_{\text{j}} C_{\text{i,j}}}{w_{\text{pred}} + \sum_j w_{\text{j}} M_{\text{i,j}}}$$

$$C_{\text{recall}} = \sum_{\text{matched i}} \frac{w_{\text{pred}} + \sum_j w_{\text{j}} C_{\text{i,j}}}{w_{\text{pred}} + \sum_j w_{\text{j}} R_{\text{i,j}}}$$

$$P_{\text{precision}} = \sum_{\text{matched i}} \frac{\sum_j w_{\text{j}} P_{\text{i,j}}}{w_{\text{pred}} + \sum_j w_{\text{j}} M_{\text{i,j}}}$$

$$P_{\text{recall}} = \sum_{\text{matched i}} \frac{\sum_j w_{\text{j}} P_{\text{i,j}}}{w_{\text{pred}} + \sum_j w_{\text{j}} R_{\text{i,j}}}$$

$$\text{precision} = \frac{C_{\text{precision}} + (w_{\text{partial}} \times P_{\text{precision}})}{\text{total \# predicates in MT}}$$

$$\text{recall} = \frac{C_{\text{recall}} + (w_{\text{partial}} \times P_{\text{recall}})}{\text{total \# predicates in REF}}$$

$$\text{f-score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

$C_{\text{precision}}$, $P_{\text{precision}}$, C_{recall} and P_{recall} are the sum of the fractional counts of correctly or partially translated semantic frames in the MT output and the reference, respectively, which can be viewed as the true positive for precision and recall of the whole semantic structure in one source utterence. Therefore, the SRL based MT evaluation metric is equivalent to the f-score, i.e., the translation accuracy for the whole predicate-argument structure.

Note that w_{pred}, w_{j} and w_{partial} are the weights for the matched predicate, arguments of type j, and partial translations. These weights can be viewed as the importance of meaning preservation for each different category of semantic roles, and the penalty for partial translations. We will describe below how these weights are estimated.

If all the reconstructed semantic frames in the MT output are completely identical to those annotated in the reference translation, and all the arguments in the reconstructed frames express the same meaning as the corresponding arguments in the reference translations, then the f-score will be equal to 1.

For instance, consider MT1 in Figure 1. The number of frames in MT1 and the reference translation are 1 and 2, respectively. The total number of participants (including both predicates and arguments) of the resume frame in both MT1 and the reference translation is 4 (one predicate and three arguments), with 2 of the arguments (one ARG1/experiencer and one ARGM-TMP/temporal) only partially translated. Assuming for now that the metric aggregates ten types of semantic roles with uniform weight for each role (optimization of weights will be discussed later), then $w_{\text{pred}} = w_j = 0.1$, and so $C_{\text{precision}}$ and C_{recall} are both zero while $P_{\text{precision}}$ and $P_{\text{recall}}$are both 0.5. If we further assume that $w_{\text{partial}} = 0.5$, then precison and recall are 0.25 and 0.125 respectively. Thus the f-score for this example is 0.17.

Both human and semi-automatic variants of the MEANT translation evaluation metric were meta-evaluated, as described next.

4 Meta-evaluation methodology

4.1 Evaluation Corpus

We leverage work from Phase 2.5 of the DARPA GALE program in which both a subset of the Chinese source sentences, as well as their English reference, are being annotated with semantic role labels in Propbank style. The corpus also includes three participating state-of-the-art MT systems' output. For present purposes, we randomly drew 40 sentences from the newswire genre of the corpus to form a meta-evaluation corpus. To maintain a controlled environment for experiments and consistent comparison, the evaluation corpus is fixed throughout this work.

4.2 Correlation with human judgements on adequacy

We followed the benchmark assessment procedure in WMT and NIST MetricsMaTr (Callison-Burch *et al.*, 2008, 2010), assessing the performance of the proposed evaluation metric at the sentence level using ranking preference consistency, which also known as Kendall's τ rank correlation coefficient, to evaluate the correlation of the proposed metric with human judgments on translation adequacy ranking. A higher value for τ indicates more similarity to the ranking by the evaluation metric to the human judgment. The range of possible values of correlation coefficient is [-1,1], where 1 means the systems are ranked

Table 2: List of semantic roles that human judges are requested to label.

Label	Event	Label	Event
Agent	who	Location	where
Action	did	Purpose	why
Experiencer	what	Manner	how
Patient	whom	Degree or Extent	how
Temporal	when	Other adverbial arg.	how

in the same order as the human judgment and -1 means the systems are ranked in the reverse order as the human judgment.

5 Experiment: Using human SRL

The first experiment aims to provide a more concrete understanding of one of the key questions as to the upper bounds of the proposed evaluation metric: how well can human annotators perform in reconstructing the semantic frames in MT output? This is important since MT output is still not close to perfectly grammatical for a good syntactic parsing—applying automatic shallow semantic parsers, which are trained on grammatical input and valid syntactic parse trees, on MT output may significantly underestimate translation utility.

5.1 Experimental setup

We thus introduce HMEANT, a variant of MEANT based on the idea that semantic role labeling can be simplified into a task that is easy and fast even for untrained humans. The human annotators are given only very simple instructions of less than half a page, along with two examples. Table 2 shows the list of labels annotators are requested to annotate, where the semantic role labeling instructions are given in the intuitive terms of "who did what to whom, when, where, why and how". To facilitate the inter-annotator agreement experiments discussed later, each sentence is independently assigned to at least two annotators.

After calculating the SRL scores based on the confusion matrix collected from the annotation and evaluation, we estimate the weights using grid search to optimize correlation with human adequacy judgments.

5.2 Results: Correlation with human judgement

Table 3 shows results indicating that HMEANT correlates with human judgment on adequacy as well as HTER does (0.432), and is far superior to BLEU (0.198) or other surface-oriented metrics.

Inspection of the cross validation results shown in Table 4 indicates that the estimated weights are not overfitting. Recall that the weights used in HMEANT are globally estimated (by grid search) using the evaluation corpus. To analyze stability, the corpus is also partitioned randomly into four folds of equal size. For each fold, another grid search is also run. R_{HMEANT} is the rank at which the Kendall's correlation for HMEANT is found, if the Kendall's correlations for all points in the grid search space are sorted. Many similar weight-vectors produce the same Kendall's correlation score, so "distinct R" shows how many distinct Kendall's correlation scores exist in each case—between 16 and 29. HMEANT's weight settings always produce Kendall's correlation scores among the top 5, regardless of which fold is chosen, indicating good stability of HMEANT's weight-vector.

Table 3: Sentence-level correlation with human adequacy judgments, across the evaluation metrics.

Metrics	Kendall τ
HMEANT	**0.4324**
HTER	**0.4324**
NIST	0.2883
BLEU	0.1982
METEOR	0.1982
TER	0.1982
PER	0.1982
CDER	0.1171
WER	0.0991

Table 4: Analysis of stability for HMEANT's weight settings, with R_{HMEANT} rank and Kendall's τ correlation scores (see text).

	Fold 0	Fold 1	Fold 2	Fold 3
R_{HMEANT}	3	1	3	5
distinct R	16	29	19	17
τ_{HMEANT}	0.33	0.48	0.48	0.40
τ_{HTER}	0.59	0.41	0.44	0.30
τ_{CV} train	0.45	0.42	0.40	0.43
τ_{CV} test	0.33	0.37	0.48	0.40

Next, Kendall's τ correlation scores are shown for HMEANT on each fold. They vary from 0.33 to 0.48, and are at least as stable as those shown for HTER, where τ varies from 0.30 to 0.59.

Finally, τ_{CV} shows Kendall's correlations if the weight-vector is instead subjected to full cross-validation training and testing, again demonstrating good stability. In fact, the correlations for the training set in three of the folds (0, 2, and 3) are identical to those for HMEANT.

5.3 Results: Cost of evaluating

The time needed for training non-expert humans to carry out our annotation protocol is significantly less than HTER and gold standard Propbank annotation. The half-page instructions given to annotators required only between 5 to 15 minutes for all annotators, including time

for asking questions if necessary. Aside from providing two annotated examples, no further training was given.

Similarly, the time needed for running the evaluation metric is also significantly less than HTER—under at most 5 minutes per sentence, even for non-expert humans using no computer-assisted UI tools. The average time used for annotating each sentence was lower bounded by 2 minutes and upper bounded by 3 minutes, and the time used for determing the translation accuracy of role fillers averaged under 2 minutes.

Note that these figures are for unskilled non-experts. These times tend to diminish significantly after annotators acquire experience.

6 Experiment: Monolinguals vs. bilinguals

We now show that using monolingual annotators is essentially just as effective as using more expensive bilingual annotators. We study the cost/benefit trade-off of using human annotators from different language backgrounds for the proposed evaluation metric, and compare whether providing the original source text helps. Note that this experiment focuses on the SRL annotation step, rather than the judgments of role filler paraphrasing accuracy, because the latter is only a simple three-way decision between "correct", "partial", and "incorrect" that is far less sensitive to the annotators' language backgrounds.

MT output is typically poor. Therefore, readers of MT output often guess the original meaning in the source input using their own language background knowledge. Readers' language background thus affects their understanding of the translation, which could affect the accuracy of capturing the key semantic roles in the translation.

6.1 Experimental Setup

Both English monolinguals and Chinese-English bilinguals (Chinese as first language and English as second language) were employed to annotate the semantic roles. For bilinguals, we also experimented with the difference in guessing constraints by optionally providing the original source input together with the translation. Therefore, there are three variations in the experiment setup: monolinguals seeing translation output only; bilinguals seeing translation output only; and bilinguals seeing both input and output.

The aim here is to do a rough sanity check on the effect of the variation of language background of the annotators; thus for these experiments we have not run the weight estimation step after SRL based f-score calculation. Instead, we simply assigned a uniform weight to all the semantic elements, and evaluated the variation under the same weight settings. (The correlation scores reported in this section are thus expected to be lower than that reported in the last section.)

Table 5: Sentence-level correlation with human adequacy judgments, for monolinguals vs. bilinguals. Uniform rather than optimized weights are used.

Metrics	**Kendall τ**
HMEANT - bilinguals	0.3514
HMEANT - monolinguals	0.3153
HMEANT - bilinguals with input	0.3153

6.2 Results

Table 5 of our results shows that using more expensive bilinguals for SRL annotation instead of monolinguals improves the correlation only slightly. The correlation coefficient of the SRL based evaluation metric driven by bilingual human annotators (0.351) is slightly better than that driven by monolingual human annotators (0.315); however, using bilinguals in the evaluation process is more costly than using monolinguals.

The results show that even allowing the bilinguals to see the input as well as the translation output for SRL annotation does not help the correlation. The correlation coefficient of the SRL based evaluation metric driven by bilingual human annotators who see also the source input sentences is 0.315 which is the same as that driven by monolingual human annotators. We find that the correlation coefficient of the proposed with human judgment on adequacy drops when bilinguals are shown to the source input sentence during annotation. Error analyses lead us to believe that annotators will drop some parts of the meaning in the translations when trying to align them to the source input.

This suggests that HMEANT requires only monolingual English annotators, who can be employed at low cost.

7 Inter-annotator agreement

One of the concerns of the proposed metric is that, given only minimal training on the task, humans would annotate the semantic roles so inconsistently as to reduce the reliability of the evaluation metric. Inter-annotator agreement (IAA) measures the consistency of human in performing the annotation task. A high IAA suggests that the annotation is consistent and the evaluation results are reliable and reproducible.

To obtain a clear analysis on where any inconsistency might lie, we measured IAA in two steps: role identification and role classification.

7.1 Experimental setup

Role identification Since annotators are not consistent in handling articles or punctuation at the beginning or the end of the annotated arguments, the agreement of semantic role identification is counted over the matching of

Table 6: Inter-annotator agreement rate on role identification (matching of word span)

Experiments	REF	MT
bilinguals working on output only	76%	72%
monolinguals working on output only	93%	75%
bilinguals working on input-output	75%	73%

Table 7: Inter-annotator agreement rate on role classification (matching of role label associated with matched word span)

Experiments	Ref	MT
bilinguals working on output only	69%	65%
monolinguals working on output only	88%	70%
bilinguals working on input-output	70%	69%

word span in the annotated role fillers with a tolerance of ±1 word in mismatch. The inter-annotator agreement rate (IAA) on the role identification task is calculated as follows. A_1 and A_2 denote the number of annotated predicates and arguments by annotator 1 and annotator 2 respectively. M_{span} denotes the number of annotated predicates and arguments with matching word span between annotators.

$$P_{\text{identification}} = \frac{M_{\text{span}}}{A_1}$$
$$R_{\text{identification}} = \frac{M_{\text{span}}}{A_2}$$
$$\text{IAA}_{\text{identification}} = \frac{2 * P_{\text{identification}} * R_{\text{identification}}}{P_{\text{identification}} + R_{\text{identification}}}$$

Role classification The agreement of classified roles is counted over the matching of the semantic role labels within two aligned word spans. The IAA on the role classification task is calculated as follows. M_{label} denotes the number of annotated predicates and arguments with matching role label between annotators.

$$P_{\text{classification}} = \frac{M_{\text{label}}}{A_1}$$
$$R_{\text{classification}} = \frac{M_{\text{label}}}{A_2}$$
$$\text{IAA}_{\text{classification}} = \frac{2 * P_{\text{classification}} * R_{\text{classification}}}{P_{\text{classification}} + R_{\text{classification}}}$$

7.2 Results

The high inter-annotator agreement suggests that the annotation instructions provided to the annotators are in general sufficient and the evaluation is repeatable and could be automated in the future. Table 6 and 7 show the annotators reconstructed the semantic frames quite consistently, even they were given only simple and minimal training.

We have noticed that the agreement on role identification is higher than that on role classification. This suggests that there are role confusion errors among the annotators. We expect a slightly more detailed instructions and explanations on different roles will further improve the IAA on role classification.

The results also show that monolinguals seeing output only have the highest IAA in semantic frame reconstruction. Data analyses lead us to believe the monolinguals are the most constrained group in the experiments. The monolingual annotators can only guess the meaning in the MT output using their English language knowledge. Therefore, they all understand the translation almost the same way, even if the translation is incorrect.

On the other hand, bilinguals seeing both the input and output discover the mistranslated portions, and often unconsciously try to compensate by re-interpreting the MT output with information not necessarily appearing in the translation, in order to better annotate what they think it should have conveyed. Since there are many degrees of freedom in this sort of compensatory re-interpretation, this group achieved a lower IAA than the monolinguals.

Bilinguals seeing only output appear to take this even a step further: confronted with a poor translation, they often unconsciously try to guess what the original input might have been. Consequently, they agree the least, because they have the most freedom in applying their own knowledge of the unseen input language, when compensating for poor translations.

8 Experiment: Using automatic SRL

In the previous experiment, we showed that the proposed evaluation metric driven by human semantic role annotators performed as well as HTER. It is now worth asking a deeper question: can we further reduce the labor cost of MEANT by using automatic shallow semantic parsing instead of humans for semantic role labeling?

Note that this experiment focuses on understanding the cost/benefit trade-off for the semantic frame reconstruction step. For SRL annotation, we replace humans with automatic shallow semantic parsing. We decouple this from the ternary judgments of role filler accuracy, which are still made by humans. However, we believe the evaluation of role filler accuracy will also be automatable.

8.1 Experimental setup

We performed three variations of the experiments to assess the performance degradation from the automatic approximation of semantic frame reconstruction in each translation (reference translation and MT output): we applied automatic shallow semantic parsing on the MT output only; on the reference translation only; and on both reference translation and MT output. For the semantic

Table 8: Sentence-level correlation with human adequacy judgments. *The weights for individual roles in the metric are tuned by optimizing the correlation.

Metrics	Kendall τ
HTER	0.4324
HMEANT gold - monolinguals *	0.4324
HMEANT auto - monolinguals *	0.3964
MEANT gold - auto *	0.3694
MEANT auto - auto *	0.3423
NIST	0.2883
BLEU / METEOR / TER / PER	0.1982
CDER	0.1171
WER	0.0991

parser, we used ASSERT (Pradhan *et al.*, 2004) which achieves roughly 87% semantic role labeling accuracy.

8.2 Results

Table 8 shows that the proposed SRL based evaluation metric correlates slightly worse than HTER with a much lower labor cost. The correlation with human judgment on adequacy of the fully automated SRL annotation version, i.e., applying ASSERT on both the reference translation and the MT output, of the SRL based evaluation metric is about 80% of that of HTER. The results also show that the correlation with human judgment on adequacy of either one side of translation using automatic SRL is in the 85% to 95% range of that HTER.

9 Conclusion

We have presented MEANT, a novel semantic MT evaluation metric that assesses the translation accuracy via Propbank-style semantic predicates, roles, and fillers. MEANT provides an intuitive picture on how much information is correctly translated in the MT output.

MEANT can be run using inexpensive untrained monolinguals and yet correlates with human judgments on adequacy as well as HTER with a lower labor cost. In contrast to HTER, which requires rigorous training of human experts to find a minimum edit of the translation (an exponentially large search space), MEANT requires untrained humans to make well-defined, bounded decisions on annotating semantic roles and judging translation correctness. The process by which MEANT reconstructs the semantic frames in a translation and then judges translation correctness of the role fillers conceptually models how humans read and understand translation output.

We also showed that using automatic shallow semantic parser to further reduce the labor cost of the proposed metric successfully approximates roughly 80% of the correlation with human judgment on adequacy. The results suggest future potential for a fully automatic variant of MEANT that could out-perform current automatic MT evaluation metrics and still perform near the level of HTER.

Numerous intriguing questions arise from this work. A further investigation into the correlation of each of the individual roles to human adequacy judgments is detailed elsewhere, along with additional improvements to the MEANT family of metrics (Lo and Wu, 2011). Another interesting investigation would then be to similarly replicate this analysis of the impact of each individual role, but using automatically rather than manually labeled semantic roles, in order to ascertain whether the more difficult semantic roles for automatic semantic parsers might also correspond to the less important aspects of end-to-end MT utility.

Acknowledgments

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under GALE Contract Nos. HR0011-06-C-0022 and HR0011-06-C-0023 and by the Hong Kong Research Grants Council (RGC) research grants GRF621008, GRF612806, DAG03/04.EG09, RGC6256/00E, and RGC6083/99E. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency.

References

Satanjeev Banerjee and Alon Lavie. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the 43th Annual Meeting of the Association of Computational Linguistics (ACL-05)*, pages 65–72, 2005.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. Re-evaluating the role of BLEU in Machine Translation Research. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, pages 249–256, 2006.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. (Meta-) evaluation of Machine Translation. In *Proceedings of the 2nd Workshop on Statistical Machine Translation*, pages 136–158, 2007.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. Further Meta-evaluation of Machine Translation. In *Proceedings of the 3rd Workshop on Statistical Machine Translation*, pages 70–106, 2008.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Pryzbocki, and Omar Zaidan.

Findings of the 2010 Joint Workshop on Statistical Machine Translation and Metrics for Machine Translation. In *Proceedings of the Joint 5th Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53, Uppsala, Sweden, 15-16 July 2010.

G. Doddington. Automatic Evaluation of Machine Translation Quality using N-gram Co-occurrence Statistics. In *Proceedings of the 2nd International Conference on Human Language Technology Research (HLT-02)*, pages 138–145, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

Jesús Giménez and Lluís Màrquez. Linguistic Features for Automatic Evaluation of Heterogenous MT Systems. In *Proceedings of the 2nd Workshop on Statistical Machine Translation*, pages 256–264, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Jesús Giménez and Lluís Màrquez. A Smorgasbord of Features for Automatic MT Evaluation. In *Proceedings of the 3rd Workshop on Statistical Machine Translation*, pages 195–198, Columbus, OH, June 2008. Association for Computational Linguistics.

Philipp Koehn and Christof Monz. Manual and Automatic Evaluation of Machine Translation between European Languages. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 102–121, 2006.

Gregor Leusch, Nicola Ueffing, and Hermann Ney. CDer: Efficient MT Evaluation Using Block Movements. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, 2006.

Ding Liu and Daniel Gildea. Syntactic Features for Evaluation of Machine Translation. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, page 25, 2005.

Ding Liu and Daniel Gildea. Source-Language Features and Maximum Correlation Training for Machine Translation Evaluation. In *Proceedings of the 2007 Conference of the North American Chapter of the Association of Computational Linguistics (NAACL-07)*, 2007.

Chi-kiu Lo and Dekai Wu. Evaluating machine translation utility via semantic role labels. In *Seventh International Conference on Language Resources and Evaluation (LREC-2010)*, pages 2873–2877, Malta, May 2010.

Chi-kiu Lo and Dekai Wu. Semantic vs. syntactic vs. n-gram structure for machine translation evaluation. In Dekai Wu, editor, *Proceedings of SSST-4, Fourth Workshop on Syntax and Structure in Statistical Translation (at COLING 2010)*, pages 52–60, Beijing, Aug 2010.

Chi-kiu Lo and Dekai Wu. SMT vs. AI redux: How semantic frames evaluate MT more accurately. In *22nd International Joint Conference on Artificial Intelligence (IJCAI-11)*, Barcelona, Jul 2011. To appear.

Sonja Nießen, Franz Josef Och, Gregor Leusch, and Hermann Ney. A Evaluation Tool for Machine Translation: Fast Evaluation for MT Research. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC-2000)*, 2000.

Karolina Owczarzak, Josef van Genabith, and Andy Way. Evaluating machine translation with LFG dependencies. *Machine Translation*, 21:95–119, 2008.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, pages 311–318, 2002.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James H. Martin, and Dan Jurafsky. Shallow Semantic Parsing Using Support Vector Machines. In *Proceedings of the 2004 Conference on Human Language Technology and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL-04)*, 2004.

Mark Przybocki, Kay Peterson, Sébastien Bronsart, and Gregory Sanders. The NIST 2008 Metrics for Machine Translation Challenge - Overview, Methodology, Metrics, and Results. *Machine Tr*, 23:71–103, 2010.

Matthew Snover, Bonnie J. Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA-06)*, pages 223–231, 2006.

Christoph Tillmann, Stephan Vogel, Hermann Ney, Arkaitz Zubiaga, and Hassan Sawaf. Accelerated DP Based Search For Statistical Translation. In *Proceedings of the 5th European Conference on Speech Communication and Technology (EUROSPEECH-97)*, 1997.

Clare R. Voss and Calandra R. Tate. Task-based Evaluation of Machine Translation (MT) Engines: Measuring How Well People Extract Who, When, Where-Type Elements in MT Output. In *Proceedings of the 11th Annual Conference of the European Association for Machine Translation (EAMT-2006)*, pages 203–212, Oslo, Norway, June 2006.

An exponential translation model for target language morphology

Michael Subotin
Paxfire, Inc.
Department of Linguistics & UMIACS, University of Maryland
msubotin@gmail.com

Abstract

This paper presents an exponential model for translation into highly inflected languages which can be scaled to very large datasets. As in other recent proposals, it predicts target-side phrases and can be conditioned on source-side context. However, crucially for the task of modeling morphological generalizations, it estimates feature parameters from the entire training set rather than as a collection of separate classifiers. We apply it to English-Czech translation, using a variety of features capturing potential predictors for case, number, and gender, and one of the largest publicly available parallel data sets. We also describe generation and modeling of inflected forms unobserved in training data and decoding procedures for a model with non-local target-side feature dependencies.

1 Introduction

Translation into languages with rich morphology presents special challenges for phrase-based methods. Thus, Birch et al (2008) find that translation quality achieved by a popular phrase-based system correlates significantly with a measure of target-side, but not source-side morphological complexity. Recently, several studies (Bojar, 2007; Avramidis and Koehn, 2009; Ramanathan et al., 2009; Yeniterzi and Oflazer, 2010) proposed modeling target-side morphology in a phrase-based factored models framework (Koehn and Hoang, 2007). Under this approach linguistic annotation of source sentences is analyzed using heuristics to identify relevant structural phenomena, whose occurrences are in turn used to compute additional relative frequency (maximum likelihood) estimates predicting target-side inflections. This approach makes it difficult to handle the complex interplay between different predictors for inflections. For example, the accusative case is usually preserved in translation, so that nouns appearing in the direct object position of English clauses tend to be translated to words with accusative case markings in languages with richer morphology, and vice versa. However, there are exceptions. For example, some verbs that place their object in the accusative case in Czech may be rendered as prepositional constructions in English (Naughton, 2005):

David was looking for Jana
David hledal Janu
David searched Jana-ACC

Conversely, direct objects of some English verbs can be translated by nouns with genitive case markings in Czech:

David asked Jana where Karel was
David zeptal se Jany kde je Karel
David asked SELF Jana-GEN where is Karel

Furthermore, English noun modifiers are often rendered by Czech possessive adjectives and a verbal complement in one language is commonly translated by a nominalizing complement in another language, so that the part of speech (POS) of its head need not be preserved. These complications make it difficult to model morphological phenomena using

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 230–238,
Portland, Oregon, June 19-24, 2011.

closed-form estimates. This paper presents an alternative approach based on exponential phrase models, which can straightforwardly handle feature sets with arbitrarily elaborate source-side dependencies.

2 Hierarchical phrase-based translation

We take as our starting point David Chiang's Hiero system, which generalizes phrase-based translation to substrings with gaps (Chiang, 2007). Consider for instance the following set of context-free rules with a single non-terminal symbol:

$\langle A\,,\, A\rangle \rightarrow \langle A_1\, A_2\,,\, A_1\, A_2\rangle$
$\langle A\,,\, A\rangle \rightarrow \langle\ d'\, A_1\, idées\, A_2\,,\, A_1\, A_2\, ideas\rangle$
$\langle A\,,\, A\rangle \rightarrow \langle incolores\,,\, colorless\rangle$
$\langle A\,,\, A\rangle \rightarrow \langle vertes\,,\, green\rangle$
$\langle A\,,\, A\rangle \rightarrow \langle dorment\, A\,,\, sleep\, A\rangle$
$\langle A\,,\, A\rangle \rightarrow \langle furieusement\,,\, furiously\rangle$

It is one of many rule sets that would suffice to generate the English translation 1b for the French sentence 1a.

1a. *d' incolores idées vertes dorment furieusement*
1b. *colorless green ideas sleep furiously*

As shown by Chiang (2007), a weighted grammar of this form can be collected and scored by simple extensions of standard methods for phrase-based translation and efficiently combined with a language model in a CKY decoder to achieve large improvements over a state-of-the-art phrase-based system. The translation is chosen to be the target-side yield of the highest-scoring synchronous parse consistent with the source sentence. Although a variety of scores interpolated into the decision rule for phrase-based systems have been investigated over the years, only a handful have been discovered to be consistently useful. In this work we concentrate on extending the target-given-source phrase model[1].

3 Exponential phrase models with shared features

The model used in this work is based on the familiar equation for conditional exponential models:

$$p(Y|X) = \frac{e^{\vec{w}\cdot\vec{f}(X,Y)}}{\sum_{Y'\in GEN(X)} e^{\vec{w}\cdot\vec{f}(X,Y')}}$$

where $\vec{f}(X,Y)$ is a vector of feature functions, $\vec{w}$ is a corresponding weight vector, so that $\vec{w}\cdot\vec{f}(X,Y) = \sum_i w_i f_i(X,Y)$, and $GEN(X)$ is a set of values corresponding to Y. For a target-given-source phrase model the predicted outcomes are target-side phrases r^y, the model is conditioned on a source-side phrase r^x together with some context, and each $GEN(X)$ consists of target phrases r^y co-occurring with a given source phrase r^x in the grammar.

Maximum likelihood estimation for exponential model finds the values of weights that maximize the likelihood of the training data, or, equivalently, its logarithm:

$$LL(\vec{w}) = \log \prod_{m=1}^{M} p(Y_m|X_m) = \sum_{m=1}^{M} \log\, p(Y_m|X_m)$$

where the expressions range over all training instances $\{m\}$. In this work we extend the objective using an ℓ_2 regularizer (Ng, 2004; Gao et al., 2007). We obtain the counts of instances and features from the standard heuristics used to extract the grammar from a word-aligned parallel corpus.

Exponential models and other classifiers have been used in several recent studies to condition phrase model probabilities on source-side context (Chan et al 2007; Carpuat and Wu 2007a; Carpuat and Wu 2007b). However, this has been generally accomplished by training independent classifiers associated with different source phrases. This approach is not well suited to modeling target-language inflections, since parameters for the features associated with morphological markings and their predictors would be estimated separately from many, generally very small training sets, thereby preventing the model from making precisely the kind of generalization beyond specific phrases that we seek to obtain. Instead we continue the approach proposed in Subotin (2008), where a single model defined by the equations above is trained on all of the data, so that parameters for features that are shared by rule sets with difference source sides reflect cumulative feature counts, while the standard relative

[1]To avoid confusion with features of the exponential models described below we shall use the term "model" rather than "feature" for the terms interpolated using MERT.

frequency model can be obtained as a special case of maximum likelihood estimation for a model containing only the features for rules.[2] Recently, Jeong et al (2010) independently proposed an exponential model with shared features for target-side morphology in application to lexical scores in a treelet-based system.

4 Features

The feature space for target-side inflection models used in this work consists of features tracking the source phrase and the corresponding target phrase together with its complete morphological tag, which will be referred to as *rule features* for brevity. The feature space also includes features tracking the source phrase together with the lemmatized representation of the target phrase, called *lemma features* below. Since there is little ambiguity in lemmatization for Czech, the lemma representations were for simplicity based on the most frequent lemma for each token. Finally, we include features associating aspects of source-side annotation with inflections of aligned target words. The models include features for three general classes of morphological types: number, case, and gender. We add inflection features for all words aligned to at least one English verb, adjective, noun, pronoun, or determiner, excepting definite and indefinite articles. A separate feature type marks cases where an intended inflection category is not applicable to a target word falling under these criteria due to a POS mismatch between aligned words.

4.1 Number

The inflection for number is particularly easy to model in translating from English, since it is generally marked on the source side, and POS taggers based on the Penn treebank tag set attempt to infer it in cases where it is not. For word pairs whose source-side word is a verb, we add a feature marking the number of its subject, with separate features for noun and pronoun subjects. For word pairs whose source side is an adjective, we add a feature marking the number of the head of the smallest noun phrase that contains it.

[2]Note that this model is estimated from the *full* parallel corpus, rather than a held-out development set.

4.2 Case

Among the inflection types of Czech nouns, the only type that is not generally observed in English and does not belong to derivational morphology is inflection for case. Czech marks seven cases: nominal, genitive, dative, accusative, vocative, locative, and instrumental. Not all of these forms are overtly distinguished for all lexical items, and some words that function syntactically as nouns do not inflect at all. Czech adjectives also inflect for case and their case has to match the case of their governing noun. However, since the source sentence and its annotation contain a variety of predictors for case, we model it using only source-dependent features. The following feature types for case were included:

- The structural role of the aligned source word or the head of the smallest noun phrase containing the aligned source word. Features were included for the roles of subject, direct object, and nominal predicate.
- The preposition governing the smallest noun phrase containing the aligned source word, if it is governed by a preposition.
- An indicator for the presence of a possessive marker modifying the aligned source word or the head of the smallest noun phrase containing the aligned source word.
- An indicator for the presence of a numeral modifying the aligned source word or the head of the smallest noun phrase containing the aligned source word.
- An indication that aligned source word modified by quantifiers *many*, *most*, *such*, or *half*. These features would be more properly defined based on the identity of the target word aligned to these quantifiers, but little ambiguity seems to arise from this substitution in practice.
- The lemma of the verb governing the aligned source word or the head of the smallest noun phrase containing the aligned source word. This is the only lexicalized feature type used in the model and we include only those features which occur over 1,000 times in the training data.

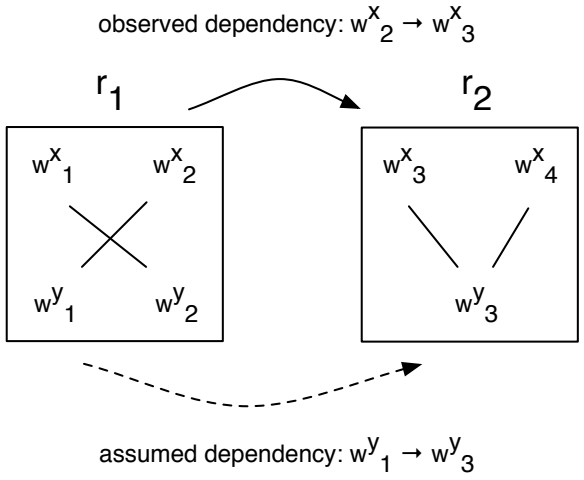


Figure 1: Inferring syntactic dependencies.

Features corresponding to aspects of the source word itself and features corresponding to aspects of the head of a noun phrase containing it were treated as separate types.

4.3 Gender

Czech nouns belong to one of three cases: feminine, masculine, and neuter. Verbs and adjectives have to agree with nouns for gender, although this agreement is not marked in some forms of the verb. In contrast to number and case, Czech gender generally cannot be predicted from any aspect of the English source sentence, which necessitates the use of features that depend on another target-side word. This, in turn, requires a more elaborate decoding procedure, described in the next section. For verbs we add a feature associating the gender of the verb with the gender of its subject. For adjectives, we add a feature tracking the gender of the governing nouns. These dependencies are inferred from source-side annotation via word alignments, as depicted in figure 1, without any use of target-side dependency parses.

5 Decoding with target-side model dependencies

The procedure for decoding with non-local target-side feature dependencies is similar in its general outlines to the standard method of decoding with a language model, as described in Chiang (2007). The search space is organized into arrays called *charts*, each containing a set of items whose scores can be compared with one another for the purposes of pruning. Each rule that has matched the source sentence belongs to a *rule chart* associated with its location-anchored sequence of non-terminal and terminal source-side symbols and any of its aspects which may affect the score of a translation hypothesis when it is combined with another rule. In the case of the language model these aspects include any of its target-side words that are part of still incomplete n-grams. In the case of non-local target-side dependencies this includes any information about features needed for this rule's estimate and tracking some target-side inflection beyond it or features tracking target-side inflections within this rule and needed for computation of another rule's estimate. We shall refer to both these types of information as *messages*, alluding to the fact that it will need to be conveyed to another point in the derivation to finish the computation. Thus, a rule chart for a rule with one non-terminal can be denoted as as $\left\langle x_{i+1}^{i_1} A x_{j_1+1}^{j}, \mu \right\rangle$, where we have introduced the symbol μ to represent the set of messages associated with a given item in the chart. Each item in the chart is associated with a score s, based on any submodels and heuristic estimates that can already be computed for that item and used to arrange the chart items into a priority queue. Combinations of one or more rules that span a substring of terminals are arranged into a different type of chart which we shall call *span charts*. A span chart has the form $[i_1, j_1; \mu_1]$, where μ_1 is a set of messages, and its items are likewise prioritized by a partial score s_1.

The decoding procedure used in this work is based on the *cube pruning* method, fully described in Chiang (2007). Informally, whenever a rule chart is combined with one or more span charts corresponding to its non-terminals, we select best-scoring items from each chart and update derivation scores by performing any model computations that become possible once we combine the corresponding items. Crucially, whenever an item in one of the charts crosses a pruning threshold, we discard the rest of that chart's items, even though one of them could generate a better-scoring partial derivation in com-

bination with an item from another chart. It is therefore important to estimate incomplete model scores as well as we can. We estimate these scores by computing exponential models using all features without non-local dependencies.

Schematically, our decoding procedure can be illustrated by three elementary cases. We take the example of computing an estimate for a rule whose only terminal on both sides is a verb and which requires a feature tracking the target-side gender inflection of the subject. We make use of a cache storing all computed numerators and denominators of the exponential model, which makes it easy to recompute an estimate given an additional feature and use the difference between it and the incomplete estimate to update the score of the partial derivation. In the simplest case, illustrated in figure 2, the non-local feature depends on the position within the span of the rule's non-terminal symbol, so that its model estimate can be computed when its rule chart is combined with the span chart for its non-terminal symbol. This is accomplished using a *feature message*, which indicates the gender inflection for the subject and is denoted as $m_f(i)$, where the index i refers to the position of its "recipient". Figure 3 illustrates the case where the non-local feature lies outside the rule's span, but the estimated rule lies inside a non-terminal of the rule which contains the feature dependency. This requires sending a *rule message* $m_r(i)$, which includes information about the estimated rule (which also serves as a pointer to the score cache) and its feature dependency. The final example, shown in figure 4, illustrates the case where both types of messages need to be propagated until we reach a rule chart that spans both ends of the dependency. In this case, the full estimate for a rule is computed while combining charts neither of which corresponds directly to that rule.

A somewhat more formal account of the decoding procedure is given in figure 5, which shows a partial set of inference rules, generally following the formalism used in Chiang (2007), but simplifying it in several ways for brevity. Aside from the notation introduced above, we also make use of two updating functions. The message-updating function $u_m(\mu)$ takes a set of messages and outputs another set that includes those messages $m_r(k)$ and $m_f(k)$ whose destination k lies outside the span i, j of the

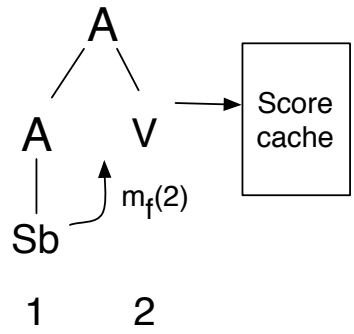


Figure 2: Non-local dependency, case A.

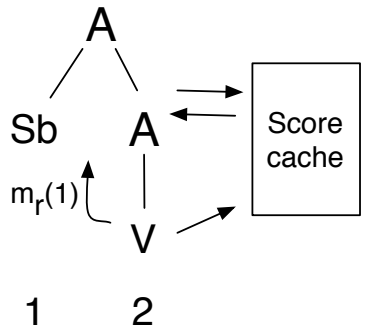


Figure 3: Non-local dependency, case B.

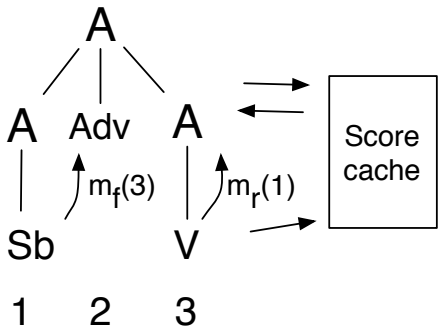


Figure 4: Non-local dependency, case C.

$$\frac{}{\langle r^x, r^y \rangle : s}$$

$$\frac{\left\langle x_{i+1}^{j}, \mu \right\rangle : s}{[i, j; u_m(\mu)] : s u_s(\mu)}$$

$$\frac{\left\langle x_{i+1}^{i_1} A x_{j_1+1}^{j}, \mu \right\rangle : s \quad [i_1, j_1; \mu_1] : s_1}{[i, j; u_m(\mu \cup \mu_1)] : s s_1 u_s(\mu \cup \mu_1)}$$

Figure 5: Simplified set of inference rules for decoding with target-side model dependencies.

chart. The score-updating function $u_s(\mu)$ computes those model estimates which can be completed using a message in the set μ and returns the difference between the new and old scores.

6 Modeling unobserved target inflections

As a consequence of translating into a morphologically rich language, some inflected forms of target words are unobserved in training data and cannot be generated by the decoder under standard phrase-based approaches. Exponential models with shared features provide a straightforward way to estimate probabilities of unobserved inflections. This is accomplished by extending the sets of target phrases $GEN(X)$ over which the model is normalized by including some phrases which have not been observed in the original sets. When additional rule features with these unobserved target phrases are included in the model, their weights will be estimated even though they never appear in the training examples (i.e, in the numerator of their likelihoods).

We generate unobserved morphological variants for target phrases starting from a generation procedure for target words. Morphological variants for words were generated using the ÚFAL MORPHO tool (Kolovratník and Přikryl, 2008). The forms produced by the tool from the lemma of an observed inflected word form were subjected to several restrictions:

- For nouns, generated forms had to match the original form for number.
- For verbs, generated forms had to match the original form for tense and negation.
- For adjectives, generated forms had to match the original form for degree of comparison and negation.
- For pronouns, excepting relative and interrogative pronouns, generated forms had to match the original form for number, case, and gender.
- Non-standard inflection forms for all POS were excluded.

The following criteria were used to select rules for which expanded inflection sets were generated:

- The target phrase had to contain exactly one word for which inflected forms could be generated according to the criteria given above.
- If the target phrase contained prepositions or numerals, they had to be in a position not adjacent to the inflected word. The rationale for this criterion was the tendency of prepositions and numerals to determine the inflection of adjacent words.
- The lemmatized form of the phrase had to account for at least 25% of target phrases extracted for a given source phrase.

The standard relative frequency estimates for the $p(X|Y)$ phrase model and the lexical models do not provide reasonable values for the decoder scores for unobserved rules and words. In contrast, exponential models with surface and lemma features can be straightforwardly trained for all of them. For the experiments described below we trained an exponential model for the $p(Y|X)$ lexical model. For greater speed we estimate the probabilities for the other two models using interpolated Kneser-Ney smoothing (Chen and Goodman, 1998), where the surface form of a rule or an aligned word pair plays to role of a trigram, the pairing of the source surface form with the lemmatized target form plays the role of a bigram, and the source surface form alone plays the role of a unigram.

7 Corpora and baselines

We investigate the models using the 2009 edition of the parallel treebank from ÚFAL (Bojar and Žabokrtský, 2009), containing 8,029,801 sentence pairs from various genres. The corpus comes with automatically generated annotation and a randomized split into training, development, and testing sets. Thus, the annotation for the development and testing sets provides a realistic reflection of what could be obtained for arbitrary source text. The English-side annotation follows the standards of the Penn Treebank and includes dependency parses and structural role labels such as subject and object. The Czech side is labeled with several layers of annotation, of which only the morphological tags and lemmas are used in this study. The Czech tags follow the standards of the Prague Dependency Treebank 2.0.

The impact of the models on translation accuracy was investigated for two experimental conditions:

- Small data set: trained on the news portion of the data, containing 140,191 sentences; development and testing sets containing 1500 sentences of news text each.
- Large data set: trained on all the training data; developing and testing sets each containing 1500 sentences of EU, news, and fiction data in equal portions. The other genres were excluded from the development and testing sets because manual inspection showed them to contain a considerable proportion of non-parallel sentences pairs.

All conditions use word alignments produced by sequential iterations of IBM model 1, HMM, and IBM model 4 in GIZA++, followed by "diag-and" symmetrization (Koehn et al., 2003). Thresholds for phrase extraction and decoder pruning were set to values typical for the baseline system (Chiang, 2007). Unaligned words at the outer edges of rules or gaps were disallowed. A 5-gram language model with modified interpolated Kneser-Ney smoothing (Chen and Goodman, 1998) was trained by the SRILM toolkit (Stolcke, 2002) on a set of 208 million running words of text obtained by combining the monolingual Czech text distributed by the 2010 ACL MT workshop with the Czech portion of the training data. The decision rule was based on the standard log-linear interpolation of several models, with weights tuned by MERT on the development set (Och, 2003). The baselines consisted of the language model, two phrase translation models, two lexical models, and a brevity penalty.

The proposed exponential phrase model contains several modifications relative to a standard phrase model (called *baseline A* below) with potential to improve translation accuracy, including smoothed estimates and estimates incorporating target-side tags. To gain better insight into the role played by different elements of the model, we also tested a second baseline phrase model (baseline B), which attempted to isolate the exponential model itself from auxiliary modifications. Baseline B was different from the experimental condition in using a grammar limited to observed inflections and in replacing the exponential $p(Y|X)$ phrase model by a relative frequency phrase model. It was different from baseline A in computing the frequencies for the $p(Y|X)$ phrase model based on counts of *tagged* target phrases and in using the same smoothed estimates in the other models as were used in the experimental condition.

8 Parameter estimation

Parameter estimation was performed using a modified version of the maximum entropy module from SciPy (Jones et al., 2001) and the LBFGS-B algorithm (Byrd et al., 1995). The objective included an ℓ_2 regularizer with the regularization trade-off set to 1. The amount of training data presented a practical challenge for parameter estimation. Several strategies were pursued to reduce the computational expenses. Following the approach of Mann et al (2009), the training set was split into many approximately equal portions, for which parameters were estimated separately and then averaged for features observed in multiple portions. The sets of target phrases for each source phrase prior to generation of additional inflected variants were truncated by discarding extracted rules which were observed with frequency less than the 200-th most frequent target phrase for that source phrase.

Additional computational challenges remained

due to an important difference between models with shared features and usual phrase models. Features appearing with source phrases found in development and testing data share their weights with features appearing with other source phrases, so that filtering the training set for development and testing data affects the solution. Although there seems to be no reason why this would positively affect translation accuracy, to be methodologically strict we estimate parameters for rule and lemma features without inflection features for larger models, and then combine them with weights for inflection feature estimated from a smaller portion of training data. This should affect model performance negatively, since it precludes learning trade-offs between evidence provided by the different kinds of features, and therefore it gives a conservative assessment of the results that could be obtained at greater computational costs. The large data model used parameters for the inflection features estimated from the small data set. In the runs where exponential models were used they replaced the corresponding baseline phrase translation model.

9 Results and discussion

Table 1 shows the results. Aside from the two baselines described in section 7 and the full exponential model, the table also reports results for an exponential model that excluded gender-based features (and hence non-local target-side dependencies). The highest scores were achieved by the full exponential model, although baseline B produced surprisingly disparate effects for the two data sets. This suggests a complex interplay of the various aspects of the model and training data whose exploration could further improve the scores. Inclusion of gender-based features produced small but consistent improvements. Table 2 shows a summary of the grammars.

We further illustrate general properties of these models using toy examples and the actual parameters estimated from the large data set. Table 3 shows representative rules with two different source sides. The column marked "no infl." shows model estimates computed without inflection features. One can see that for both rule sets the estimated probabilities for rules observed a single time is only slightly higher than probabilities for generated unobserved rules. However, rules with relatively high counts in the second set receive proportionally higher estimates, while the difference between the singleton rule and the most frequent rule in the second set, which was observed 3 times, is smoothed away to an even greater extent. The last two columns show model estimates when various inflection features are included. There is a grammatical match between nominative case for the target word and subject position for the aligned source word and between accusative case for the target word and direct object role for the aligned source word. The other pairings represent grammatical mismatches. One can see that the probabilities for rules leading to correct case matches are considerably higher than the alternatives with incorrect case matches.

Condition	Small set	Large set
Baseline A	0.1964	0.2562
Baseline B	0.2067	0.2522
Expon-gender	0.2114	0.2598
Expon+gender	0.2128	0.2615

Table 1: BLUE scores on testing. See section 7 for a description of the baselines.

Condition	Total rules	Observed rules
Small set	17,089,850	3,983,820
Large set	39,349,268	23,679,101

Table 2: Grammar sizes after and before generation of unobserved inflections (all filtered for dev/test sets).

r^x	Count	Case	No infl.	Sb	Obj
1	1	Dat	0.085	0.037	0.035
1	3	Acc	0.086	0.092	0.204
1	0	Nom	0.063	0.416	0.063
2	1	Instr	0.007	0.002	0.003
2	31	Nom	0.212	0.624	0.169
2	0	Acc	0.005	0.002	0.009

Table 3: The effect of inflection features on estimated probabilities.

10 Conclusion

This paper has introduced a scalable exponential phrase model for target languages with complex morphology that can be trained on the full parallel corpus. We have showed how it can provide estimates for inflected forms unobserved in the training data and described decoding procedures for features with non-local target-side dependencies. The results suggest that the model should be especially useful for languages with sparser resources, but that performance improvements can be obtained even for a very large parallel corpus.

Acknowledgments

I would like to thank Philip Resnik, Amy Weinberg, Hal Daumé III, Chris Dyer, and the anonymous reviewers for helpful comments relating to this work.

References

E. Avramidis and P. Koehn. 2008. Enriching Morphologically Poor Languages for Statistical Machine Translation. In *Proc. ACL 2008*.

A. Birch, M. Osborne and P. Koehn. 2008. Predicting Success in Machine Translation. The Conference on Empirical Methods in Natural Language Processing (EMNLP), 2008.

O. Bojar. 2007. English-to-Czech Factored Machine Translation. In Proceedings of the Second Workshop on Statistical Machine Translation.

O. Bojar and Z. Žabokrtský. 2009. Large Parallel Treebank with Rich Annotation. Charles University, Prague. http://ufal.mff.cuni.cz/czeng/czeng09/, 2009.

R. H. Byrd, P. Lu and J. Nocedal. 1995. A Limited Memory Algorithm for Bound Constrained Optimization. SIAM Journal on Scientific and Statistical Computing, 16(5), pp. 1190-1208.

M. Carpuat and D. Wu. 2007a. Improving Statistical Machine Translation using Word Sense Disambiguation. Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007).

M. Carpuat and D. Wu. 2007b. How Phrase Sense Disambiguation outperforms Word Sense Disambiguation for Statistical Machine Translation. 11th Conference on Theoretical and Methodological Issues in Machine Translation (TMI 2007)

Y.S. Chan, H.T. Ng, and D. Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proc. ACL 2007*.

S.F. Chen and J.T. Goodman. 1998. An Empirical Study of Smoothing Techniques for Language Modeling. *Technical Report TR-10-98, Computer Science Group, Harvard University*.

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

J. Gao, G. Andrew, M. Johnson and K. Toutanova. 2007. A Comparative Study of Parameter Estimation Methods for Statistical Natural Language Processing. In *Proc. ACL 2007*.

M. Jeong, K. Toutanova, H. Suzuki, and C. Quirk. 2010. A Discriminative Lexicon Model for Complex Morphology. The Ninth Conference of the Association for Machine Translation in the Americas (AMTA-2010).

E. Jones, T. Oliphant, P. Peterson and others. SciPy: Open source scientific tools for Python. http://www.scipy.org/

P. Koehn and H. Hoang. 2007. Factored translation models. The Conference on Empirical Methods in Natural Language Processing (EMNLP), 2007.

P. Koehn, F.J. Och, and D. Marcu. 2003. Statistical Phrase-Based Translation. In Proceedings of the Human Language Technology Conference (HLT-NAACL 2003).

D. Kolovratník and L. Přikryl. 2008. Programátorská dokumentace k projektu Morfo. http://ufal.mff.cuni.cz/morfo/, 2008.

G. Mann, R. McDonald, M. Mohri, N. Silberman, D. Walker. 2009. Efficient Large-Scale Distributed Training of Conditional Maximum Entropy Models. Advances in Neural Information Processing Systems (NIPS), 2009.

J. Naughton. 2005. *Czech. An Essential Grammar.* Routledge, 2005.

A.Y. Ng. 2004. Feature selection, L1 vs. L2 regularization, and rotational invariance. In Proceedings of the Twenty-first International Conference on Machine Learning.

F.J. Och. 2003. Minimum Error Rate Training for Statistical Machine Translation. In *Proc. ACL 2003*.

A. Ramanathan, H. Choudhary, A. Ghosh, P. Bhattacharyya. 2009. Case markers and Morphology: Addressing the crux of the fluency problem in English-Hindi SMT. In *Proc. ACL 2009*.

A. Stolcke. 2002. SRILM – An Extensible Language Modeling Toolkit. International Conference on Spoken Language Processing, 2002.

M. Subotin. 2008. Generalizing Local Translation Models. Proceedings of SSST-2, Second Workshop on Syntax and Structure in Statistical Translation.

R. Yeniterzi and K. Oflazer. 2010. Syntax-to-Morphology Mapping in Factored Phrase-Based Statistical Machine Translation from English to Turkish. In *Proc. ACL 2010*.

Bayesian Inference for Zodiac and Other Homophonic Ciphers

Sujith Ravi and Kevin Knight
University of Southern California
Information Sciences Institute
Marina del Rey, California 90292
{sravi,knight}@isi.edu

Abstract

We introduce a novel Bayesian approach for deciphering complex substitution ciphers. Our method uses a decipherment model which combines information from letter n-gram language models as well as word dictionaries. Bayesian inference is performed on our model using an efficient sampling technique. We evaluate the quality of the Bayesian decipherment output on simple and homophonic letter substitution ciphers and show that unlike a previous approach, our method consistently produces almost 100% accurate decipherments. The new method can be applied on more complex substitution ciphers and we demonstrate its utility by cracking the famous Zodiac-408 cipher in a fully automated fashion, which has never been done before.

1 Introduction

Substitution ciphers have been used widely in the past to encrypt secrets behind messages. These ciphers replace (English) plaintext letters with cipher symbols in order to generate the ciphertext sequence.

There exist many published works on automatic decipherment methods for solving simple letter-substitution ciphers. Many existing methods use dictionary-based attacks employing huge word dictionaries to find plaintext patterns within the ciphertext (Peleg and Rosenfeld, 1979; Ganesan and Sherman, 1993; Jakobsen, 1995; Olson, 2007). Most of these methods are heuristic in nature and search for the best deterministic key during decipherment. Others follow a probabilistic decipherment approach. Knight et al. (2006) use the Expectation Maximization (EM) algorithm (Dempster et al., 1977) to search for the best probabilistic key using letter n-gram models. Ravi and Knight (2008) formulate decipherment as an integer programming problem and provide an exact method to solve simple substitution ciphers by using letter n-gram models along with deterministic key constraints. Corlett and Penn (2010) work with large ciphertexts containing thousands of characters and provide another exact decipherment method using an A* search algorithm. Diaconis (2008) presents an analysis of Markov Chain Monte Carlo (MCMC) sampling algorithms and shows an example application for solving simple substitution ciphers.

Most work in this area has focused on solving simple substitution ciphers. But there are variants of substitution ciphers, such as homophonic ciphers, which display increasing levels of difficulty and present significant challenges for decipherment. The famous Zodiac serial killer used one such cipher system for communication. In 1969, the killer sent a three-part cipher message to newspapers claiming credit for recent shootings and crimes committed near the San Francisco area. The 408-character message (Zodiac-408) was manually decoded by hand in the 1960's. Oranchak (2008) presents a method for solving the Zodiac-408 cipher automatically with a dictionary-based attack using a genetic algorithm. However, his method relies on using plaintext words from the known solution to solve the cipher, which departs from a strict decipherment scenario.

In this paper, we introduce a novel method for

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 239–247,
Portland, Oregon, June 19-24, 2011.

solving substitution ciphers using Bayesian learning. Our novel contributions are as follows:

- We present a new probabilistic decipherment approach using Bayesian inference with sparse priors, which can be used to solve different types of substitution ciphers.
- Our new method combines information from word dictionaries along with letter n-gram models, providing a robust decipherment model which offsets the disadvantages faced by previous approaches.
- We evaluate the Bayesian decipherment output on three different types of substitution ciphers and show that unlike a previous approach, our new method solves all the ciphers completely.
- Using the Bayesian decipherment, we show for the first time a truly automated system that successfully solves the Zodiac-408 cipher.

2 Letter Substitution Ciphers

We use natural language processing techniques to attack letter substitution ciphers. In a letter substitution cipher, every letter p in the natural language (plaintext) sequence is replaced by a cipher token c, according to some substitution key.

For example, an English plaintext

"H E L L O _ W O R L D ..."

may be enciphered as:

"N O E E I _ T I M E L ..."

according to the key:

p: ABCDEFGHIJKLMNOPQRSTUVWXYZ_
c: XYZLOHANBCDEFGIJKMPQRSTUVW_

where, "_" represents the space character (word boundary) in the English and ciphertext messages.

If the recipients of the ciphertext message have the substitution key, they can use it (in reverse) to recover the original plaintext. The sender can encrypt the message using one of many different cipher systems. The particular type of cipher system chosen determines the properties of the key. For example, the substitution key can be deterministic in both the encipherment and decipherment directions as shown in the above example—i.e., there is a 1-to-1 correspondence between the plaintext letters and ciphertext symbols. Other types of keys exhibit nondeterminism either in the encipherment (or decipherment) or both directions.

2.1 Simple Substitution Ciphers

The key used in a simple substitution cipher is deterministic in both the encipherment and decipherment directions, i.e., there is a 1-to-1 mapping between plaintext letters and ciphertext symbols. The example shown earlier depicts how a simple substitution cipher works.

Data: In our experiments, we work with a 414-letter simple substitution cipher. We encrypt an original English plaintext message using a randomly generated simple substitution key to create the ciphertext. During the encipherment process, we preserve spaces between words and use this information for decipherment—i.e., plaintext character "_" maps to ciphertext character "_". Figure 1 (top) shows a portion of the ciphertext along with the original plaintext used to create the cipher.

2.2 Homophonic Ciphers

A homophonic cipher uses a substitution key that maps a plaintext letter to more than one cipher symbol.

For example, the English plaintext:

"H E L L O _ W O R L D ..."

may be enciphered as:

"65 82 51 84 05 _ 60 54 42 51 45 ..."

according to the key:

A: 09 12 33 47 53 67 78 92
B: 48 81
...
E: 14 16 24 44 46 55 57 64 74 82 87
...
L: 51 84
...
Z: 02

Here, "_" represents the space character in both English and ciphertext. Notice the non-determinism involved in the enciphering direction—the English

letter "L" is substituted using different symbols (51, 84) at different positions in the ciphertext.

These ciphers are more complex than simple substitution ciphers. Homophonic ciphers are generated via a non-deterministic encipherment process—the key is 1-to-many in the enciphering direction. The number of potential cipher symbol substitutes for a particular plaintext letter is often proportional to the frequency of that letter in the plaintext language—for example, the English letter "E" is assigned more cipher symbols than "Z". The objective of this is to flatten out the frequency distribution of ciphertext symbols, making a frequency-based cryptanalysis attack difficult.

The substitution key is, however, deterministic in the decipherment direction—each ciphertext symbol maps to a single plaintext letter. Since the ciphertext can contain more than 26 types, we need a larger alphabet system—we use a numeric substitution alphabet in our experiments.

Data: For our decipherment experiments on homophonic ciphers, we use the same 414-letter English plaintext used in Section 2.1. We encrypt this message using a homophonic substitution key (available from *http://www.simonsingh.net/The_Black_Chamber/homophoniccipher.htm*). As before, we preserve spaces between words in the ciphertext. Figure 1 (middle) displays a section of the homophonic cipher (with spaces) and the original plaintext message used in our experiments.

2.3 Homophonic Ciphers without spaces (Zodiac-408 cipher)

In the previous two cipher systems, the word-boundary information was preserved in the cipher. We now consider a more difficult homophonic cipher by removing space characters from the original plaintext.

The English plaintext from the previous example now looks like this:

"HELLOWORLD ..."

and the corresponding ciphertext is:

"65 82 51 84 05 60 54 42 51 45 ..."

Without the word boundary information, typical dictionary-based decipherment attacks fail on such ciphers.

Zodiac-408 cipher: Homophonic ciphers without spaces have been used extensively in the past to encrypt secret messages. One of the most famous homophonic ciphers in history was used by the infamous Zodiac serial killer in the 1960's. The killer sent a series of encrypted messages to newspapers and claimed that solving the ciphers would reveal clues to his identity. The identity of the Zodiac killer remains unknown to date. However, the mystery surrounding this has sparked much interest among cryptanalysis experts and amateur enthusiasts.

The Zodiac messages include two interesting ciphers: (1) a 408-symbol homophonic cipher without spaces (which was solved manually by hand), and (2) a similar looking 340-symbol cipher that has yet to be solved.

Here is a sample of the Zodiac-408 cipher message:

...

and the corresponding section from the original English plaintext message:

I L I K E K I L L I N G P E O P L
E B E C A U S E I T I S S O M U C
H F U N I T I S M O R E F U N T H
A N K I L L I N G W I L D G A M E
I N T H E F O R R E S T B E C A U
S E M A N I S T H E M O S T D A N
G E R O U E A N A M A L O F A L L
T O K I L L S O M E T H I N G G I

...

Besides the difficulty with missing word boundaries and non-determinism associated with the key, the Zodiac-408 cipher poses several additional challenges which makes it harder to solve than any standard homophonic cipher. There are spelling mistakes in the original message (for example, the English word "PARADISE" is misspelt as

"PARADICE") which can divert a dictionary-based attack. Also, the last 18 characters of the plaintext message does not seem to make any sense ("EBEORIETEMETHHPITI").

Data: Figure 1 (bottom) displays the Zodiac-408 cipher (consisting of 408 tokens, 54 symbol types) along with the original plaintext message. We run the new decipherment method (described in Section 3.1) and show that our approach can successfully solve the Zodiac-408 cipher.

3 Decipherment

Given a ciphertext message $c_1...c_n$, the goal of decipherment is to uncover the hidden plaintext message $p_1...p_n$. The size of the keyspace (i.e., number of possible key mappings) that we have to navigate during decipherment is huge—a simple substitution cipher has a keyspace size of 26!, whereas a homophonic cipher such as the Zodiac-408 cipher has 26^{54} possible key mappings.

Next, we describe a new Bayesian decipherment approach for tackling substitution ciphers.

3.1 Bayesian Decipherment

Bayesian inference methods have become popular in natural language processing (Goldwater and Griffiths, 2007; Finkel et al., 2005; Blunsom et al., 2009; Chiang et al., 2010). Snyder et al. (2010) proposed a Bayesian approach in an archaeological decipherment scenario. These methods are attractive for their ability to manage uncertainty about model parameters and allow one to incorporate prior knowledge during inference. A common phenomenon observed while modeling natural language problems is *sparsity*. For simple letter substitution ciphers, the original substitution key exhibits a 1-to-1 correspondence between the plaintext letters and cipher types. It is not easy to model such information using conventional methods like EM. But we can easily specify priors that favor sparse distributions within the Bayesian framework.

Here, we propose a novel approach for deciphering substitution ciphers using Bayesian inference. Rather than enumerating all possible keys (26! for a simple substitution cipher), our Bayesian framework requires us to sample only a small number of keys during the decipherment process.

Probabilistic Decipherment: Our decipherment method follows a noisy-channel approach. We are faced with a ciphertext sequence $c = c_1...c_n$ and we want to find the (English) letter sequence $p = p_1...p_n$ that maximizes the probability $P(p|c)$.

We first formulate a generative story to model the process by which the ciphertext sequence is generated.

1. Generate an English plaintext sequence $p = p_1...p_n$, with probability $P(p)$.
2. Substitute each plaintext letter p_i with a ciphertext token c_i, with probability $P(c_i|p_i)$ in order to generate the ciphertext sequence $c = c_1...c_n$.

We build a statistical English language model (LM) for the plaintext source model $P(p)$, which assigns a probability to any English letter sequence. Our goal is to estimate the channel model parameters θ in order to maximize the probability of the observed ciphertext c:

$$\arg\max_{\theta} P(c) = \arg\max_{\theta} \sum_{p} P_\theta(p, c) \quad (1)$$

$$= \arg\max_{\theta} \sum_{p} P(p) \cdot P_\theta(c|p) \quad (2)$$

$$= \arg\max_{\theta} \sum_{p} P(p) \cdot \prod_{i=1}^{n} P_\theta(c_i|p_i) \quad (3)$$

We estimate the parameters θ using Bayesian learning. In our decipherment framework, a Chinese Restaurant Process formulation is used to model both the source and channel. The detailed generative story using CRPs is shown below:

1. $i \leftarrow 1$
2. Generate the English plaintext letter p_1, with probability $P_0(p_1)$
3. Substitute p_1 with cipher token c_1, with probability $P_0(c_1|p_1)$
4. $i \leftarrow i + 1$
5. Generate English plaintext letter p_i, with probability

$$\frac{\alpha \cdot P_0(p_i|p_{i-1}) + C_1^{i-1}(p_{i-1}, p_i)}{\alpha + C_1^{i-1}(p_{i-1})}$$

Plaintext:	D E C I P H E R M E N T _ I S _ T H E _ A N A L Y S I S _ O F _ D O C U M E N T S _ W R I T T E N _ I N _ A N C I E N T _ L A N G U A G E S _ W H E R E _ T H E _ ...
Ciphertext:	i n g c m p n q s n w f _ c v _ f p n _ o w o k t v c v _ h u _ i h g z s n w f v _ r q c f f n w _ c w _ o w g c n w f _ k o w a z o a n v _ r p n q n _ f p n _ ...
Bayesian solution:	D E C I P H E R M E N T _ I S _ T H E _ A N A L Y S I S _ O F _ D O C U M E N T S _ W R I T T E N _ I N _ A N C I E N T _ L A N G U A G E S _ W H E R E _ T H E _ ...

Plaintext:	D E C I P H E R M E N T _ I S _ T H E _ A N A L Y S I S _ O F _ D O C U M E N T S _ W R I T T E N _ I N _ ...
Ciphertext:	79 57 62 93 95 68 44 77 22 74 59 97 _ 32 86 _ 85 56 82 _ 67 59 67 84 52 86 73 11 _ 99 10 _ 45 90 13 61 27 98 71 49 19 _ 60 80 88 85 20 55 59 _ 32 91 _ ...
Bayesian solution:	D E C I P H E R M E N T _ I S _ T H E _ A N A L Y S I S _ O F _ D O C U M E N T S _ W R I T T E N _ I N _ ...

Ciphertext:	
Plaintext:	“ I LIKE KILLING PEOPLE BECAUSE IT IS SO MUCH FUN IT IS MORE FUN THAN KILLING WILD GAME IN THE FORREST BECAUSE MAN IS THE MOST DANGEROUE ANAMAL OF ALL TO KILL SOMETHING GIVES ME THE MOST THRILLING EXPERENCE IT IS EVEN BETTER THAN GETTING YOUR ROCKS OFF WITH A GIRL THE BEST PART OF IT IS THAE WHEN I DIE I WILL BE REBORN IN PARADICE AND THEI HAVE KILLED WILL BECOME MY SLAVES I WILL NOT GIVE YOU MY NAME BECAUSE YOU WILL TRY TO SLOI DOWN OR ATOP MY COLLECTIOG OF SLAVES FOR MY AFTERLIFE EBEORIETEMETHHPITI ”
Bayesian solution (final decoding):	I L I K E K I L L I N G P E O P L E B E C A U S E I T I S S O M U C H F U N I T I **A** M O R E F U N T H A N K I L L I N G W I L D G A M E I N T H E F O R R E S T B E C A U S E M A N I S T H E M O **A** T D A N G E R **T** U E A N A M A L O F A L L ...
(with spaces shown):	I _ L I K E _ K I L L I N G _ P E O P L E _ B E C A U S E _ I T _ I S _ S O _ M U C H _ F U N _ I T _ I _ **A** _ M O R E _ F U N _ T H A N _ K I L L I N G _ W I L D _ G A M E _ I N _ T H E _ F O R _ R E S T _ B E C A U S E _ M A N _ I S _ T H E _ M O **A** T _ D A N G E R _ **T** U E _ A N _ A M _ A L _ O F _ A L L _ ...

Figure 1: Samples from the ciphertext sequence, corresponding English plaintext message and output from Bayesian decipherment (using word+3-gram LM) for three different ciphers: (a) Simple Substitution Cipher (top), (b) Homophonic Substitution Cipher with spaces (middle), and (c) Zodiac-408 Cipher (bottom).

6. Substitute p_i with cipher token c_i, with probability

$$\frac{\beta \cdot P_0(c_i|p_i) + C_1^{i-1}(p_i, c_i)}{\beta + C_1^{i-1}(p_i)}$$

7. With probability P_{quit}, quit; else go to Step 4.

This defines the probability of any given derivation, i.e., any plaintext hypothesis corresponding to the given ciphertext sequence. The base distribution P_0 represents prior knowledge about the model parameter distributions. For the plaintext source model, we use probabilities from an English language model and for the channel model, we specify a uniform distribution (i.e., a plaintext letter can be substituted with any given cipher type with equal probability). C_1^{i-1} represents the count of events occurring before plaintext letter p_i in the derivation (we call this the "cache"). α and β represent Dirichlet prior hyperparameters over the source and channel models respectively. A large prior value implies that characters are generated from the base distribution P_0, whereas a smaller value biases characters to be generated with reference to previous decisions inside the cache (favoring sparser distributions).

Efficient inference via type sampling: We use a Gibbs sampling (Geman and Geman, 1984) method for performing inference on our model. We could follow a point-wise sampling strategy, where we sample plaintext letter choices for every cipher token, one at a time. But we already know that the substitution ciphers described here exhibit determinism in the deciphering direction,[1] i.e., although we have no idea about the key mappings themselves, we do know that there exists only a single plaintext letter mapping for every cipher symbol type in the true key. So sampling plaintext choices for every cipher token separately is not an efficient strategy—our sampler may spend too much time exploring invalid keys (which map the same cipher symbol to different plaintext letters).

Instead, we use a *type sampling* technique similar to the one proposed by Liang et al. (2010). Under this scheme, we sample plaintext letter choices for each cipher symbol type. In every step, we sample a new plaintext letter for a cipher type and update the entire plaintext hypothesis (i.e., plaintext letters at all corresponding positions) to reflect this change. For example, if we sample a new choice p_{new} for a cipher symbol which occurs at positions $4, 10, 18$, then we update plaintext letters p_4, p_{10} and p_{18} with the new choice p_{new}.

Using the property of exchangeability, we derive an incremental formula for re-scoring the probability of a new derivation based on the probability of the old derivation—when sampling at position i, we pretend that the area affected (within a context window around i) in the current plaintext hypothesis occurs at the end of the corpus, so that both the old and new derivations share the same cache.[2] While we may make corpus-wide changes to a derivation in every sampling step, exchangeability allows us to perform scoring in an efficient manner.

Combining letter n-gram language models with word dictionaries: Many existing probabilistic approaches use statistical letter n-gram language models of English to assign $P(p)$ probabilities to plaintext hypotheses during decipherment. Other decryption techniques rely on word dictionaries (using words from an English dictionary) for attacking substitution ciphers.

Unlike previous approaches, our decipherment method combines information from both sources—letter n-grams and word dictionaries. We build an interpolated *word+n-gram* LM and use it to assign $P(p)$ probabilities to any plaintext letter sequence $p_1 ... p_n$.[3] The advantage is that it helps direct the sampler towards plaintext hypotheses that resemble natural language—high probability letter sequences which form valid words such as "H E L L O" instead of sequences like "T X H R T". But in addition to this, using letter n-gram information makes

[1] This assumption does not strictly apply to the Zodiac-408 cipher where a few cipher symbols exhibit non-determinism in the decipherment direction as well.

[2] The relevant context window that is affected when sampling at position i is determined by the word boundaries to the left and right of i.

[3] We set the interpolation weights for the word and n-gram LM as (0.9, 0.1). The word-based LM is constructed from a dictionary consisting of 9,881 frequently occurring words collected from Wikipedia articles. We train the letter n-gram LM on 50 million words of English text available from the Linguistic Data Consortium.

our model robust against variations in the original plaintext (for example, unseen words or misspellings as in the case of Zodiac-408 cipher) which can easily throw off dictionary-based attacks. Also, it is hard for a point-wise (or type) sampler to "find words" starting from a random initial sample, but easier to "find n-grams".

Sampling for ciphers without spaces: For ciphers without spaces, dictionaries are hard to use because we do not know where words start and end. We introduce a new sampling operator which counters this problem and allows us to perform inference using the same decipherment model described earlier. In a first sampling pass, we sample from 26 plaintext letter choices (e.g., "A", "B", "C", ...) for every cipher symbol type as before. We then run a second pass using a new sampling operator that iterates over adjacent plaintext letter pairs p_{i-1}, p_i in the current hypothesis and samples from two choices—(1) add a word boundary (space character "_") between p_{i-1} and p_i, or (2) remove an existing space character between p_{i-1} and p_i.

For example, given the English plaintext hypothesis "`... A B O Y ...`", there are two sampling choices for the letter pair `A,B` in the second step. If we decide to add a word boundary, our new plaintext hypothesis becomes "`... A _ B O Y ...`".

We compute the derivation probability of the new sample using the same efficient scoring procedure described earlier. The new strategy allows us to apply Bayesian decipherment even to ciphers without spaces. As a result, we now have a new decipherment method that consistently works for a range of different types of substitution ciphers.

Decoding the ciphertext: After the sampling run has finished,[4] we choose the final sample as our English plaintext decipherment output.

[4]For letter substitution decipherment we want to keep the language model probabilities fixed during training, and hence we set the prior on that model to be high ($\alpha = 10^4$). We use a sparse prior for the channel ($\beta = 0.01$). We instantiate a key which matches frequently occurring plaintext letters to frequent cipher symbols and use this to generate an initial sample for the given ciphertext and run the sampler for 5000 iterations. We use a linear annealing schedule during sampling decreasing the temperature from $10 \rightarrow 1$.

4 Experiments and Results

We run decipherment experiments on different types of letter substitution ciphers (described in Section 2). In particular, we work with the following three ciphers:

(a) 414-letter Simple Substitution Cipher

(b) 414-letter Homophonic Cipher (with spaces)

(c) Zodiac-408 Cipher

Methods: For each cipher, we run and compare the output from two different decipherment approaches:

1. **EM Method** using letter n-gram LMs following the approach of Knight et al. (2006). They use the EM algorithm to estimate the channel parameters θ during decipherment training. The given ciphertext c is then decoded by using the Viterbi algorithm to choose the plaintext decoding p that maximizes $P(p) \cdot P_\theta(c|p)^3$, stretching the channel probabilities.

2. **Bayesian Decipherment** method using word+n-gram LMs (novel approach described in Section 3.1).

Evaluation: We evaluate the quality of a particular decipherment as the percentage of cipher tokens that are decoded correctly.

Results: Figure 2 compares the decipherment performance for the EM method with Bayesian decipherment (using type sampling and sparse priors) on three different types of substitution ciphers. Results show that our new approach (Bayesian) outperforms the EM method on all three ciphers, solving them completely. Even with a 3-gram letter LM, our method yields a +63% improvement in decipherment accuracy over EM on the homophonic cipher with spaces. We observe that the word+3-gram LM proves highly effective when tackling more complex ciphers and cracks the Zodiac-408 cipher. Figure 1 shows samples from the Bayesian decipherment output for all three ciphers. For ciphers without spaces, our method automatically guesses the word boundaries for the plaintext hypothesis.

Method	LM	Accuracy (%) on 414-letter Simple Substitution Cipher	Accuracy (%) on 414-letter Homophonic Substitution Cipher (with spaces)	Accuracy (%) on Zodiac-408 Cipher
1. EM	2-gram	83.6	30.9	_
	3-gram	99.3	32.6	0.3* (*28.8 *with 100 restarts*)
2. Bayesian	3-gram	**100.0**	95.2	23.0
	word+2-gram	**100.0**	**100.0**	_
	word+3-gram	**100.0**	**100.0**	**97.8**

Figure 2: Comparison of decipherment accuracies for EM *versus* Bayesian method when using different language models of English on the three substitution ciphers: (a) 414-letter Simple Substitution Cipher, (b) 414-letter Homophonic Substitution Cipher (with spaces), and (c) the famous Zodiac-408 Cipher.

For the Zodiac-408 cipher, we compare the performance achieved by Bayesian decipherment under different settings:

- *Letter n-gram* versus *Word+n-gram* LMs—Figure 2 shows that using a word+3-gram LM instead of a 3-gram LM results in +75% improvement in decipherment accuracy.
- *Sparse* versus *Non-sparse* priors—We find that using a sparse prior for the channel model (β = 0.01 versus 1.0) helps for such problems and produces better decipherment results (97.8% versus 24.0% accuracy).
- *Type* versus *Point-wise* sampling—Unlike point-wise sampling, type sampling quickly converges to better decipherment solutions. After 5000 sampling passes over the entire data, decipherment output from type sampling scores 97.8% accuracy compared to 14.5% for the point-wise sampling run.[5]

We also perform experiments on shorter substitution ciphers. On a 98-letter simple substitution cipher, EM using 3-gram LM achieves 41% accuracy, whereas the method from Ravi and Knight (2009) scores 84% accuracy. Our Bayesian method performs the best in this case, achieving 100% with word+3-gram LM.

5 Conclusion

In this work, we presented a novel Bayesian decipherment approach that can effectively solve a variety of substitution ciphers. Unlike previous approaches, our method combines information from letter n-gram language models and word dictionaries and provides a robust decipherment model. We empirically evaluated the method on different substitution ciphers and achieve perfect decipherments on all of them. Using Bayesian decipherment, we can successfully solve the Zodiac-408 cipher—the first time this is achieved by a fully automatic method in a strict decipherment scenario.

For future work, there are other interesting decipherment tasks where our method can be applied. One challenge is to crack the unsolved Zodiac-340 cipher, which presents a much harder problem than the solved version.

[5]Both sampling runs were seeded with the same random initial sample.

Acknowledgements

The authors would like to thank the reviewers for their comments. This research was supported by NSF grant IIS-0904684.

References

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A Gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*, pages 782–790.

David Chiang, Jonathan Graehl, Kevin Knight, Adam Pauls, and Sujith Ravi. 2010. Bayesian inference for finite-state transducers. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL/HLT)*, pages 447–455.

Eric Corlett and Gerald Penn. 2010. An exact A* method for deciphering letter-substitution ciphers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1040–1047.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Persi Diaconis. 2008. The Markov Chain Monte Carlo revolution. *Bulletin of the American Mathematical Society*, 46(2):179–205.

Jenny Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 363–370.

Ravi Ganesan and Alan T. Sherman. 1993. Statistical techniques for language recognition: An introduction and guide for cryptanalysts. *Cryptologia*, 17(4):321–366.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741.

Sharon Goldwater and Thomas Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751.

Thomas Jakobsen. 1995. A fast method for cryptanalysis of substitution ciphers. *Cryptologia*, 19(3):265–274.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics*, pages 499–506.

Percy Liang, Michael I. Jordan, and Dan Klein. 2010. Type-based MCMC. In *Proceedings of the Conference on Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 573–581.

Edwin Olson. 2007. Robust dictionary attack of short simple substitution ciphers. *Cryptologia*, 31(4):332–342.

David Oranchak. 2008. Evolutionary algorithm for decryption of monoalphabetic homophonic substitution ciphers encoded as constraint satisfaction problems. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, pages 1717–1718.

Shmuel Peleg and Azriel Rosenfeld. 1979. Breaking substitution ciphers using a relaxation algorithm. *Comm. ACM*, 22(11):598–605.

Sujith Ravi and Kevin Knight. 2008. Attacking decipherment problems optimally with low-order n-gram models. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, pages 812–819.

Sujith Ravi and Kevin Knight. 2009. Probabilistic methods for a Japanese syllable cipher. In *Proceedings of the International Conference on the Computer Processing of Oriental Languages (ICCPOL)*, pages 270–281.

Benjamin Snyder, Regina Barzilay, and Kevin Knight. 2010. A statistical model for lost language decipherment. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1048–1057.

Interactive Topic Modeling

Yuening Hu
Department of Computer Science
University of Maryland
ynhu@cs.umd.edu

Jordan Boyd-Graber
iSchool
University of Maryland
jbg@umiacs.umd.edu

Brianna Satinoff
Department of Computer Science
University of Maryland
bsonrisa@cs.umd.edu

Abstract

Topic models have been used extensively as a tool for corpus exploration, and a cottage industry has developed to tweak topic models to better encode human intuitions or to better model data. However, creating such extensions requires expertise in machine learning unavailable to potential end-users of topic modeling software. In this work, we develop a framework for allowing users to iteratively refine the topics discovered by models such as latent Dirichlet allocation (LDA) by adding constraints that enforce that sets of words must appear together in the same topic. We incorporate these constraints interactively by selectively removing elements in the state of a Markov Chain used for inference; we investigate a variety of methods for incorporating this information and demonstrate that these interactively added constraints improve topic usefulness for simulated and actual user sessions.

1 Introduction

Probabilistic topic models, as exemplified by probabilistic latent semantic indexing (Hofmann, 1999) and latent Dirichlet allocation (LDA) (Blei et al., 2003) are unsupervised statistical techniques to discover the thematic topics that permeate a large corpus of text documents. Topic models have had considerable application beyond natural language processing in computer vision (Rob et al., 2005), biology (Shringarpure and Xing, 2008), and psychology (Landauer et al., 2006) in addition to their canonical application to text.

For text, one of the few real-world applications of topic models is corpus exploration. Unannotated, noisy, and ever-growing corpora are the norm rather than the exception, and topic models offer a way to quickly get the gist a large corpus.[1]

Contrary to the impression given by the tables shown in topic modeling papers, topics discovered by topic modeling don't always make sense to ostensible end users. Part of the problem is that the objective function of topic models doesn't always correlate with human judgements (Chang et al., 2009). Another issue is that topic models — with their bag-of-words vision of the world — simply lack the necessary information to create the topics as end-users expect.

There has been a thriving cottage industry adding more and more information to topic models to correct these shortcomings; either by modeling perspective (Paul and Girju, 2010; Lin et al., 2006), syntax (Wallach, 2006; Gruber et al., 2007), or authorship (Rosen-Zvi et al., 2004; Dietz et al., 2007). Similarly, there has been an effort to inject human knowledge into topic models (Boyd-Graber et al., 2007; Andrzejewski et al., 2009; Petterson et al., 2010).

However, these are *a priori* fixes. They don't help a frustrated **consumer** of topic models staring at a collection of topics that don't make sense. In this paper, we propose interactive topic modeling (ITM), an *in situ* method for incorporating human knowledge into topic models. In Section 2, we review prior work on creating probabilistic models that incorporate human knowledge, which we extend in Section 3 to apply to ITM sessions. Section 4 discusses the implementation of this process during the inference process. Via a motivating example in Section 5, simulated ITM sessions in Section 6, and a real interactive test in Section 7, we demonstrate that our approach is able to focus a user's desires in a topic model, better capture the key properties of a corpus, and capture diverse interests from users on the web.

[1]For examples, see Rexa http://rexa.info/, JSTOR http://showcase.jstor.org/blei/, and the NIH https://app.nihmaps.org/nih/.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 248–257,
Portland, Oregon, June 19-24, 2011.

2 Putting Knowledge in Topic Models

At a high level, topic models such as LDA take as input a number of topics K and a corpus. As output, a topic model discovers K distributions over words — the namesake topics — and associations between documents and topics. In LDA both of these outputs are multinomial distributions; typically they are presented to users in summary form by listing the elements with highest probability. For an example of topics discovered from a 20-topic model of New York Times editorials, see Table 1.

When presented with poor topics learned from data, users can offer a number of complaints:[2] these documents should have similar topics but don't (Daumé III, 2009); this topic should have syntactic coherence (Gruber et al., 2007; Boyd-Graber and Blei, 2008); this topic doesn't make any sense at all (Newman et al., 2010); this topic shouldn't be associated with this document but is (Ramage et al., 2009); these words shouldn't be the in same topic but are (Andrzejewski et al., 2009); or these words should be in the same topic but aren't (Andrzejewski et al., 2009).

Many of these complaints can be addressed by using "must-link" constraints on topics, retaining Andrzejewski et al's (2009) terminology borrowed from the database literature. A "must-link" constraint is a group of words whose probability must be correlated in the topic. For example, Figure 1 shows an example constraint: {plant, factory}. After this constraint is added, the probabilities of "plant" and "factory" in each topic are likely to both be high or both be low. It's unlikely for "plant" to have high probability in a topic and "factory" to have a low probability. In the next section, we demonstrate how such constraints can be built into a model and how they can even be added while inference is underway.

In this paper, we view constraints as transitive; if "plant" is in a constraint with "factory" and "factory" is in a constraint with "production," then "plant" is in a constraint with "production." Making this assumption can simplify inference slightly, which we take advantage of in Section 3.1, but the real reason for this assumption is because not doing so would

[2]Citations in this litany of complaints are offline solutions for addressing the problem; the papers also give motivation why such complaints might arise.

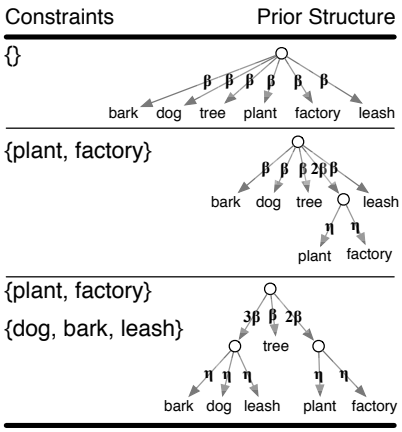


Figure 1: How adding constraints (left) creates new topic priors (right). The trees represent correlated distributions (assuming $\eta >> \beta$). After the {plant, factory} constraint is added, it is now highly unlikely for a topic drawn from the distribution to have a high probability for "plant" and a low probability for "factory" or vice versa. The bottom panel adds an additional constraint, so now dog-related words are also correlated. Notice that the two constraints themselves are uncorrelated. It's possible for both, either, or none of "bark" and "plant" (for instance) to have high probability in a topic.

introduce ambiguity over the path associated with an observed token in the generative process. As long as a word is either in a single constraint or in the general vocabulary, there is only a single path. The details of this issue are further discussed in Section 4.

3 Constraints Shape Topics

As discussed above, LDA views topics as distributions over words, and each document expresses an admixture of these topics. For "vanilla" LDA (no constraints), these are symmetric Dirichlet distributions. A document is composed of a number of observed words, which we call tokens to distinguish specific observations from the more abstract word (type) associated with each token. Because LDA assumes a document's tokens are interchangeable, it treats the document as a bag-of-words, ignoring potential relations between words.

This problem with vanilla LDA can be solved by encoding constraints, which will "guide" different words into the same topic. Constraints can be added to vanilla LDA by replacing the multinomial distribution over words for each topic with a collection of

tree-structured multinomial distributions drawn from a prior as depicted in Figure 1. By encoding word distributions as a tree, we can preserve conjugacy and relatively simple inference while encouraging correlations between related concepts (Boyd-Graber et al., 2007; Andrzejewski et al., 2009; Boyd-Graber and Resnik, 2010). Each topic has a top-level distribution over words and constraints, and each constraint in each topic has second-level distribution over the words in the constraint. Critically, the per-constraint distribution over words is engineered to be non-sparse and close to uniform. The top level distribution encodes which constraints (and unconstrained words) to include; the lower-level distribution forces the probabilities to be correlated for each of the constraints.

In LDA, a document's token is produced in the generative process by choosing a topic z and sampling a word from the multinomial distribution ϕ_z of topic z. For a constrained topic, the process now can take two steps. First, a first-level node in the tree is selected from ϕ_z. If that is an unconstrained word, the word is emitted and the generative process for that token is done. Otherwise, if the first level node is constraint l, then choose a word to emit from the constraint's distribution over words $\pi_{z,l}$.

More concretely, suppose for a corpus with M documents we have a set of constraints $\mathbf{\Omega}$. The prior structure has B branches (one branch for each word not in a constraint and one for each constraint). Then the generative process for constrained LDA is:

1. For each topic $i \in \{1, \dots K\}$:
   (a) draw a distribution over the B branches (words and constraints) $\phi_i \sim \text{Dir}(\vec{\beta})$, and
   (b) for each constraint $\Omega_j \in \mathbf{\Omega}$, draw a distribution over the words in the constraint $\pi_{i,j} \sim \text{Dir}(\eta)$, where $\pi_{i,j}$ is a distribution over the words in Ω_j
2. Then for each document $d \in \{1, \dots M\}$:
   (a) first draw a distribution over topics $\theta_d \sim \text{Dir}(\alpha)$,
   (b) then for each token $n \in \{1, \dots N_d\}$:
      i. choose a topic assignment $z_{d,n} \sim \text{Mult}(\theta_d)$, and then
      ii. choose either a constraint or word from $\text{Mult}(\phi_{z_{d,n}})$:
         A. if we chose a word, emit that word $w_{d,n}$
         B. otherwise if we chose a constraint index $l_{d,n}$, emit a word $w_{d,n}$ from the constraint's distribution over words in topic $z_{d,n}$: $w_{d,n} \sim \text{Mult}(\pi_{z_{d,n},l_{d,n}})$.

In this model, α, β, and η are Dirichlet hyperparameters set by the user; their role is explained below.

3.1 Gibbs Sampling for Topic Models

In topic modeling, collapsed Gibbs sampling (Griffiths and Steyvers, 2004) is a standard procedure for obtaining a Markov chain over the latent variables in the model. Given certain technical conditions, the stationary distribution of the Markov chain is the posterior (Neal, 1993). Given M documents the state of a Gibbs sampler for LDA consists of topic assignments for each token in the corpus and is represented as $Z = \{z_{1,1} \dots z_{1,N_1}, z_{2,1}, \dots z_{M,N_M}\}$. In each iteration, every token's topic assignment $z_{d,n}$ is resampled based on topic assignments for all the tokens except for $z_{d,n}$. (This subset of the state is denoted $\boldsymbol{Z}_{-(d,n)}$). The sampling equation for $z_{d,n}$ is

$$p(z_{d,n} = k|\mathbf{Z}_{-(d,n)}, \alpha, \beta) \propto \frac{T_{d,k} + \alpha}{T_{d,\cdot} + K\alpha} \frac{P_{k,w_{d,n}} + \beta}{P_{k,\cdot} + V\beta} \quad (1)$$

where $T_{d,k}$ is the number of times topic k is used in document d, $P_{k,w_{d,n}}$ is the number of times the type $w_{d,n}$ is assigned to topic k, and α, β are the hyperparameters of the two Dirichlet distributions, and B is the number of top-level branches (this is the vocabulary size for vanilla LDA). When a dot replaces a subscript of a count, it represents the marginal sum over all possible topics or words, e.g. $T_{d,\cdot} = \sum_k T_{d,k}$. The count statistics P and T provide summaries of the state. Typically, these only change based on assignments of latent variables in the sampler; in Section 4 we describe how changes in the model's *structure* (in addition to the latent state) can be reflected in these count statistics.

Contrasting with the above inference is the inference for a constrained model. (For a derivation, see Boyd-Graber, Blei, and Zhu (2007) for the general case or Andrzejewski, Zhu, and Craven (2009) for the specific case of constraints.) In this case the sampling equation for $z_{d,n}$ is changed to $p(z_{d,n} = k|\mathbf{Z}_{-(d,n)}, \alpha, \beta, \eta)$

$$\propto \begin{cases} \frac{T_{d,k}+\alpha}{T_{d,\cdot}+K\alpha} \frac{P_{k,w_{d,n}}+\beta}{P_{k,\cdot}+V\beta} & \text{if } \forall l, w_{d,n} \notin \Omega_l \\ \frac{T_{d,k}+\alpha}{T_{d,\cdot}+K\alpha} \frac{P_{k,l}+C_l\beta}{P_{k,\cdot}+V\beta} \frac{W_{k,l,w_{d,n}}+\eta}{W_{k,l,\cdot}+C_l\eta} & w_{d,n} \in \Omega_l \end{cases}, \quad (2)$$

where $P_{k,w_{d,n}}$ is the number of times the unconstrained word $w_{d,n}$ appears in topic k; $P_{k,l}$ is the

number of times any word of constraint Ω_l appears in topic k; $W_{k,l,w_{d,n}}$ is the number of times word $w_{d,n}$ appears in constraint Ω_l in topic k; V is the vocabulary size; C_l is the number of words in constraint Ω_l. Note the differences between these two samplers for constrained words; however, for unconstrained LDA and for unconstrained words in constrained LDA, the conditional probability is the same.

In order to make the constraints effective, we set the constraint word-distribution hyperparameter η to be much larger than the hyperparameter for the distribution over constraints and vocabulary β. This gives the constraints higher weight. Normally, estimating hyperparameters is important for topic modeling (Wallach et al., 2009). However, in ITM, sampling hyperparameters often (but not always) undoes the constraints (by making η comparable to β), so we keep the hyperparameters fixed.

4 Interactively adding constraints

For a static model, inference in ITM is the same as in previous models (Andrzejewski et al., 2009). In this section, we detail how interactively changing constraints can be accommodated in ITM, smoothly transitioning from unconstrained LDA (n.b. Equation 1) to constrained LDA (n.b. Equation 2) with one constraint, to constrained LDA with two constraints, etc.

A central tool that we will use is the strategic unassignment of states, which we call ablation (distinct from feature ablation in supervised learning). As described in the previous section, a sampler stores the topic assignment of each token. In the implementation of a Gibbs sampler, unassignment is done by setting a token's topic assignment to an invalid topic (e.g. -1, as we use here) and decrementing any counts associated with that word.

The constraints created by users implicitly signal that words in constraints don't belong in a given topic. In other models, this input is sometimes used to "fix," i.e. deterministically hold constant topic assignments (Ramage et al., 2009). Instead, we change the underlying model, using the current topic assignments as a starting position for a new Markov chain with some states strategically unassigned. How much of the existing topic assignments we use leads to four different options, which are illustrated in Figure 2.

	Previous	New
All	[bark:2, dog:3, leash:3 dog:2] [bark:2, bark:2, plant:2, tree:3] [tree:2,play:2,forest:1,leash:2]	[bark:-1, dog:-1, leash:-1 dog:-1] [bark:-1, bark:-1, plant:-1, tree:-1] [tree:-1,play:-1,forest:-1,leash:-1]
Doc	[bark:2, dog:3, leash:3 dog:2] [bark:2, bark:2, plant:2, tree:3] [tree:2,play:2,forest:1,leash:2]	[bark:-1, dog:-1, leash:-1 dog:-1] [bark:-1, bark:-1, plant:-1, tree:-1] [tree:2,play:2,forest:1,leash:2]
Term	[bark:2, dog:3, leash:3 dog:3] [bark:2, bark:2, plant:2, tree:3] [tree:2,play:2,forest:1,leash:2]	[bark:-1, dog:-1, leash:3 dog:-1] [bark:-1, bark:-1, plant:2, tree:3] [tree:2,play:2,forest:1,leash:2]
None	[bark:2, dog:3, leash:3 dog:2] [bark:2, bark:2, plant:2, tree:3] [tree:2,play:2,forest:1,leash:2]	[bark:2, dog:3, leash:3 dog:2] [bark:2, bark:2, plant:2, tree:3] [tree:2,play:2,forest:1,leash:2]

Figure 2: Four different strategies for state ablation after the words "dog" and "bark" are added to the constraint {"leash," "puppy"} to make the constraint {"dog," "bark," "leash," "puppy"}. The state is represented by showing the current topic assignment after each word (e.g. "leash" in the first document has topic 3, while "forest" in the third document has topic 1). On the left are the assignments before words were added to constraints, and on the right are the ablated assignments. Unassigned words are given the new topic assignment -1 and are highlighted in red.

All We could revoke all state assignments, essentially starting the sampler from scratch. This does not allow *interactive* refinement, as there is nothing to enforce that the new topics will be in any way consistent with the existing topics. Once the topic assignments of all states are revoked, the counts for T, P and W (as described in Section 3.1) will be zero, retaining no information about the state the user observed.

Doc Because topic models treat the document context as exchangeable, a document is a natural context for partial state ablation. Thus if a user adds a set of words $\boldsymbol{S}$ to constraints, then we have reason to suspect that all documents containing any one of $\boldsymbol{S}$ may have incorrect topic assignments. This is reflected in the state of the sampler by performing the UNASSIGN (Algorithm 1) operation for each word in any document containing a word added to a constraint.

Algorithm 1 UNASSIGN$(d, n, w_{d,n}, z_{d,n} = k)$

1: $T:\ T_{d,k} \leftarrow T_{d,k} - 1$
2: If $w_{d,n} \notin \Omega^{old}$,
  $P:\ P_{k,w_{d,n}} \leftarrow P_{k,w_{d,n}} - 1$
3: Else: suppose $w_{d,n} \in \Omega^{old}_m$,
  $P:\ P_{k,m} \leftarrow P_{k,m} - 1$
  $W:\ W_{k,m,w_{d,n}} \leftarrow W_{k,m,w_{d,n}} - 1$

This is equivalent to the Gibbs2 sampler of Yao et al. (2009) for incorporating new documents in a streaming context. Viewed in this light, a user is using words to select documents that should be treated as "new" for this refined model.

Term Another option is to perform ablation only on the topic assignments of tokens whose words have added to a constraint. This applies the unassignment operation (Algorithm 1) only to tokens whose corresponding word appears in added constraints (i.e. a subset of the Doc strategy). This makes it less likely that other tokens in similar contexts will follow the words explicitly included in the constraints to new topic assignments.

None The final option is to move words into constraints but keep the topic assignments fixed. Thus, P and W change, but not T, as described in Algorithm 2.[3] This is arguably the simplest option, and in principle is sufficient, as the Markov chain should find a stationary distribution regardless of the starting position. In practice, however, this strategy is less interactive, as users don't feel that their constraints are actually incorporated in the model, and inertia can keep the chain from reflecting the constraints.

Algorithm 2 MOVE$(d, n, w_{d,n}, z_{d,n} = k, \Omega_l)$

1: If $w_{d,n} \notin \Omega^{old}$,
$P:\ P_{k,w_{d,n}} \leftarrow P_{k,w_{d,n}} - 1,\ \ P_{k,l} \leftarrow P_{k,l} + 1$
$W:\ W_{k,l,w_{d,n}} \leftarrow W_{k,l,w_{d,n}} + 1$
2: Else, suppose $w_{d,n} \in \Omega^{old}_m$,
$P:\ P_{k,m} \leftarrow P_{k,m} - 1,\ P_{k,l} \leftarrow P_{k,l} + 1$
$W:\ W_{k,m,w_{d,n}} \leftarrow W_{k,m,w_{d,n}} - 1$
$W_{k,l,w_{d,n}} \leftarrow W_{k,l,w_{d,n}} + 1$

Regardless of what ablation scheme is used, after the state of the Markov chain is altered, the next step is to actually run inference forward, sampling assignments for the unassigned tokens for the "first" time and changing the topic assignment of previously assigned tokens. How many additional iterations are required after adding constraints is a delicate tradeoff between interactivity and effectiveness, which we investigate further in the next sections.

[3]This assumes that there is only one possible path in the constraint tree that can generate a word; in other words, this assumes that constraints are transitive, as discussed at the end of Section 2. In the more general case, when words lack a unique path in the constraint tree, an additional latent variable specifies which possible paths in the constraint tree produced the word; this would have to be sampled. All other updating strategies are immune to this complication, as the assignments are left unassigned.

5 Motivating Example

To examine the viability of ITM, we begin with a qualitative demonstration that shows the potential usefulness of ITM. For this task, we used a corpus of about 2000 New York Times editorials from the years 1987 to 1996. We started by finding 20 initial topics with no constraints, as shown in Table 1 (left).

Notice that topics 1 and 20 both deal with Russia. Topic 20 seems to be about the Soviet Union, with topic 1 about the post-Soviet years. We wanted to combine the two into a single topic, so we created a constraint with all of the clearly Russian or Soviet words (*boris, communist, gorbachev, mikhail, russia, russian, soviet, union, yeltsin*). Running inference forward 100 iterations with the **Doc** ablation strategy yields the topics in Table 1 (right). The two Russia topics were combined into Topic 20. This combination also pulled in other relevant words that not near the top of either topic before: "moscow" and "relations." Topic 1 is now more about elections in countries other than Russia. The other 18 topics changed little.

While we combined the Russian topics, other researchers analyzing large corpora might preserve the Soviet vs. post-Soviet distinction but combine topics about American government. ITM allows tuning for specific tasks.

6 Simulation Experiment

Next, we consider a process for evaluating our ITM using automatically derived constraints. These constraints are meant to simulate a user with a predefined list of categories (e.g. reviewers for journal submissions, e-mail folders, etc.). The categories grow more and more specific during the session as the simulated users add more constraint words.

To test the ability of ITM to discover relevant subdivisions in a corpus, we use a dataset with predefined, intrinsic labels and assess how well the discovered latent topic structure can reproduce the corpus's inherent structure. Specifically, for a corpus with M classes, we use the per-document topic distribution as a feature vector in a supervised classi-

Topic	Words
1	election, yeltsin, russian, political, party, democratic, russia, president, democracy, boris, country, south, years, month, government, vote, since, leader, presidential, military
2	new, york, city, state, mayor, budget, giuliani, council, cuomo, gov, plan, year, rudolph, dinkins, lead, need, governor, legislature, pataki, david
3	nuclear, arms, weapon, defense, treaty, missile, world, unite, yet, soviet, lead, secretary, would, control, korea, intelligence, test, nation, country, testing
4	president, bush, administration, clinton, american, force, reagan, war, unite, lead, economic, iraq, congress, america, iraqi, policy, aid, international, military, see
⋮	⋮
20	soviet, lead, gorbachev, union, west, mikhail, reform, change, europe, leaders, poland, communist, know, old, right, human, washington, western, bring, party

Topic	Words
1	election, democratic, south, country, president, party, africa, lead, even, democracy, leader, presidential, week, politics, minister, percent, voter, last, month, years
2	new, york, city, state, mayor, budget, council, giuliani, gov, cuomo, year, rudolph, dinkins, legislature, plan, david, governor, pataki, need, cut
3	nuclear, arms, weapon, treaty, defense, war, missile, may, come, test, american, world, would, need, lead, get, join, yet, clinton, nation
4	president, administration, bush, clinton, war, unite, force, reagan, american, america, make, nation, military, iraq, iraqi, troops, international, country, yesterday, plan
⋮	⋮
20	soviet, union, economic, reform, yeltsin, russian, lead, russia, gorbachev, leaders, west, president, boris, moscow, europe, poland, mikhail, communist, power, relations

Table 1: Five topics from a 20 topic topic model on the editorials from the New York times before adding a constraint (left) and after (right). After the constraint was added, which encouraged Russian and Soviet terms to be in the same topic, non-Russian terms gained increased prominence in Topic 1, and "Moscow" (which was not part of the constraint) appeared in Topic 20.

fier (Hall et al., 2009). The lower the classification error rate, the better the model has captured the structure of the corpus.[4]

6.1 Generating automatic constraints

We used the 20 Newsgroups corpus, which contains 18846 documents divided into 20 constituent newsgroups. We use these newsgroups as ground-truth labels.[5]

We simulate a user's constraints by ranking words in the training split by their information gain (IG).[6] After ranking the top 200 words for each class by IG, we delete words associated with multiple labels to prevent constraints for different labels from merging. The smallest class had 21 words remaining after removing duplicates (due to high overlaps of 125 words between "talk.religion.misc" and "soc.religion.christian," and 110 words between "talk.religion.misc" and "alt.atheism"), so the top 21 words for each class were the ingredients for our simulated constraints. For example, for the class "soc.religion.christian," the 21 constraint words include "catholic, scripture, resurrection, pope, sabbath, spiritual, pray, divine, doctrine, orthodox." We simulate a user's ITM session by adding a word to each of the 20 constraints until each of the constraints has 21 words.

6.2 Simulation scheme

Starting with 100 base iterations, we perform successive rounds of refinement. In each round a new constraint is added corresponding to the newsgroup labels. Next, we perform one of the strategies for state ablation, add additional iterations of Gibbs sampling, use the newly obtained topic distribution of each document as the feature vector, and perform classification on the test / train split. We do this for 21 rounds until each label has 21 constraint words. The number of LDA topics is set to 20 to match the number of newsgroups. The hyperparameters for all experiments are $\alpha = 0.1, \beta = 0.01$, and $\eta = 100$.

At 100 iterations, the chain is clearly not converged. However, we chose this number of iterations because it more closely matches the likely use case as users do not wait for convergence. Moreover, while investigations showed that the patterns shown in Fig-

[4]Our goal is to understand the phenomena of ITM, not classification, so these classification results are well below state of the art. However, adding interactively selected topics to the state of the art features (tf-idf unigrams) gives a relative error reduction of 5.1%, while just adding topics from vanilla LDA gives a relative error reduction of 1.1%. Both measurements were obtained without tuning or weighting features, so presumably better results are possible.

[5]http://people.csail.mit.edu/jrennie/20Newsgroups/
In preprocessing, we deleted short documents, leaving 15160 documents, including 9131 training documents and 6029 test documents (default split). Tokenization, lemmatization, and stopword removal was performed using the Natural Language Toolkit (Loper and Bird, 2002). Topic modeling was performed using the most frequent 5000 lemmas as the vocabulary.

[6]IG is computed by the Rainbow toolbox http://www.cs.umass.edu/ mccallum/bow/rainbow/

ure 4 were broadly consistent with larger numbers of iterations, such configurations sometimes had too much inertia to escape from local extrema. More iterations make it harder for the constraints to influence the topic assignment.

6.3 Investigating Ablation Strategies

First, we investigate which ablation strategy best allows constraints to be incorporated. Figure 3 shows the classification error of six different ablation strategies based on the number of words in each constraint, ranging from 0 to 21. Each is averaged over five different chains using 10 additional iterations of Gibbs sampling per round (other numbers of iterations are discussed in Section 6.4). The model runs forward 10 iterations after the first round, another 10 iterations after the second round, etc. In general, as the number of words per constraint increases, the error decreases as models gain more information about the classes.

Strategy **Null** is the non-interactive baseline that contains no constraints (vanilla LDA), but runs inference for a comparable number of rounds. **All Initial** and **All Full** are non-interactive baselines with all constraints known *a priori*. **All Initial** runs the model for the only the initial number of iterations (100 iterations in this experiment), while **All Full** runs the model for the total number of iterations added for the interactive version. (That is, if there were 21 rounds and each round of interactive modeling added 10 iterations, **All Full** would have 210 iterations more than **All Initial**).

While **Null** sees no constraints, it serves as an upper baseline for the error rate (lower error being better) but shows the effect of additional inference. **All Full** is a lower baseline for the error rate since it both sees the constraints at the beginning and also runs for the maximum number of total iterations. **All Initial** sees the constraints before the other ablation techniques but it has fewer total iterations.

The **Null** strategy does not perform as well as the interactive versions, especially with larger constraints. Both **All Initial** and **All Full**, however, show a larger variance (as denoted by error bands around the average trends) than the interactive schemes. This can be viewed as akin to simulated annealing, as the interactive search has more freedom to explore in early rounds. As more constraint words are added each round, the model is less free to explore.

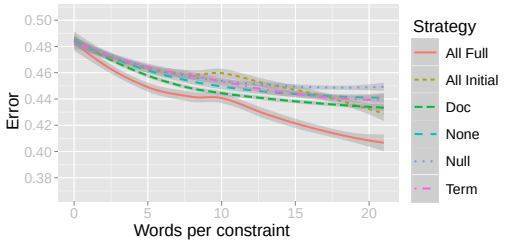


Figure 3: Error rate (y-axis, lower is better) using different ablation strategies as additional constraints are added (x-axis). **Null** represents standard LDA, as the unconstrained baseline. **All Initial** and **All Full** are non-interactive, constrained baselines. The results of **None**, **Term**, **Doc** are more stable (as denoted by the error bars), and the error rate is reduced gradually as more constraint words are added.

The error rate of each interactive ablation strategy is (as expected) between the lower and upper baselines. Generally, the constraints will influence not only the topics of the constraint words, but also the topics of the constraint words' context in the same document. **Doc** ablation gives more freedom for the constraints to overcome the inertia of the old topic distribution and move towards a new one influenced by the constraints.

6.4 How many iterations do users have to wait?

Figure 4 shows the effect of using different numbers of Gibbs sampling iterations after changing a constraint. For each of the ablation strategies, we run $\{10, 20, 30, 50, 100\}$ additional Gibbs sampling iterations. As expected, more iterations reduce error, although improvements diminish beyond 100 iterations. With more constraints, the impact of additional iterations is lessened, as the model has more *a priori* knowledge to draw upon.

For all numbers of additional iterations, while the **Null** serves as the upper baseline on the error rate in all cases, the **Doc** ablation clearly outperforms the other ablation schemes, consistently yielding a lower error rate. Thus, there is a benefit when the model has a chance to relearn the document context when constraints are added. The difference is even larger with more iterations, suggesting **Doc** needs more iterations to "recover" from unassignment.

The luxury of having hundreds or thousands of additional iterations for each constraint would be im-

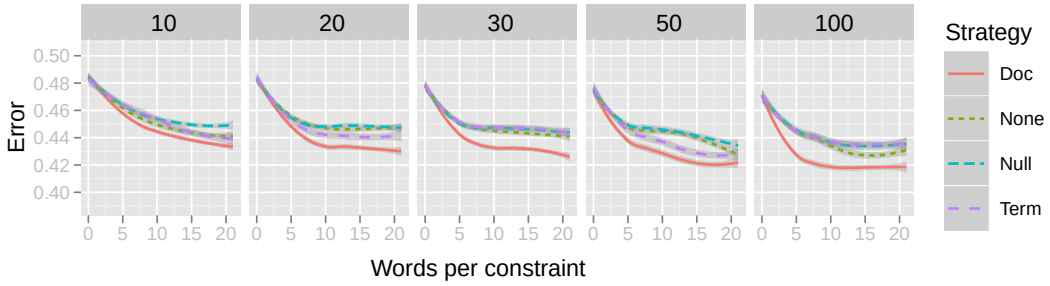


Figure 4: Classification accuracy by strategy and number of additional iterations. The **Doc** ablation strategy performs best, suggesting that the document context is important for ablation constraints. While more iterations are better, there is a tradeoff with interactivity.

practical. For even moderately sized datasets, even one iteration per second can tax the patience of individuals who want to use the system interactively. Based on these results and an *ad hoc* qualitative examination of the resulting topics, we found that 30 additional iterations of inference was acceptable; this is used in later experiments.

7 Getting Humans in the Loop

To move beyond using simulated users adding the same words regardless of what topics were discovered by the model, we needed to expose the model to human users. We solicited approximately 200 judgments from Mechanical Turk, a popular crowdsourcing platform that has been used to gather linguistic annotations (Snow et al., 2008), measure topic quality (Chang et al., 2009), and supplement traditional inference techniques for topic models (Chang, 2010). After presenting our interface for collecting judgments, we examine the results from these ITM sessions both quantitatively and qualitatively.

7.1 Interface for soliciting refinements

Figure 5 shows the interface used in the Mechanical Turk tests. The left side of the screen shows the current topics in a scrollable list, with the top 30 words displayed for each topic.

Users create constraints by clicking on words from the topic word lists. The word lists use a color-coding scheme to help the users keep track of which words they are currently grouping into constraints. The right side of the screen displays the existing constraints. Users can click on icons to edit or delete each one. The constraint currently being built is also shown.

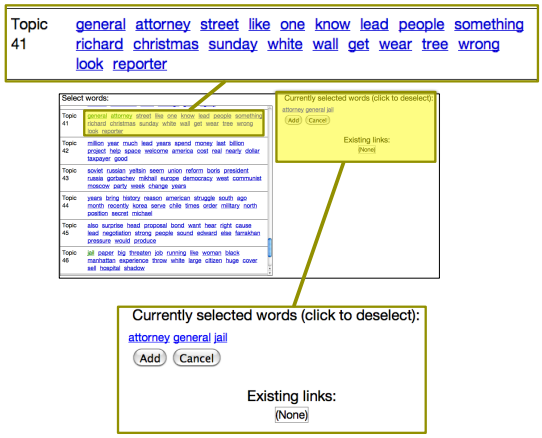


Figure 5: Interface for Mechanical Turk experiments. Users see the topics discovered by the model and select words (by clicking on them) to build constraints to be added to the model.

Clicking on a word will remove that word from the current constraint.

As in Section 6, we can compute the classification error for these users as they add words to constraints. The best users, who seemed to understand the task well, were able to decrease classification error. (Figure 6). The median user, however, had an error reduction indistinguishable from zero. Despite this, we can examine the users' behavior to better understand their goals and how they interact with the system.

7.2 Untrained users and ITM

Most of the large (10+ word) user-created constraints corresponded to the themes of the individual newsgroups, which users were able to infer from the discovered topics. Common constraint themes that

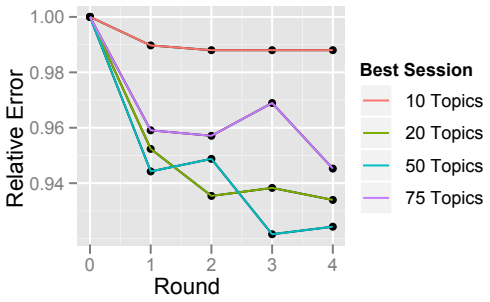


Figure 6: The relative error rate (using round 0 as a baseline) of the best Mechanical Turk user session for each of the four numbers of topics. While the 10-topic model does not provide enough flexibility to create good constraints, the best users could clearly improve classification with more topics.

matched specific newsgroups included religion, space exploration, graphics, and encryption. Other common themes were broader than individual newsgroups (e.g. sports, government and computers). Others matched sub-topics of a single newsgroup, such as homosexuality, Israel or computer programming.

Some users created inscrutable constraints, like ("better, people, right, take, things") and ("fbi, let, says"). They may have just clicked random words to finish the task quickly. While subsequent users could delete poor constraints, most chose not to. Because we wanted to understand broader behavior we made no effort to squelch such responses.

The two-word constraints illustrate an interesting contrast. Some pairs are linked together in the corpus, like ("jesus, christ") and ("solar, sun"). With others, like ("even, number") and ("book, list"), the users seem to be encouraging collocations to be in the same topic. However, the collocations may not be in any document in this corpus. Another user created a constraint consisting of male first names. A topic did emerge with these words, but the rest of the words in that topic seemed random, as male first names are not likely to co-occur in the same document.

Not all sensible constraints led to successful topic changes. Many users grouped "mac" and "windows" together, but they were almost never placed in the same topic. The corpus includes separate newsgroups for Macintosh and Windows hardware, and divergent contexts of "mac" and "windows" overpowered the prior distribution.

The constraint size ranged from one word to over 40. In general, the more words in the constraint, the more likely it was to noticeably affect the topic distribution. This observation makes sense given our ablation method. A constraint with more words will cause the topic assignments to be reset for more documents.

8 Discussion

In this work, we introduced a means for end-users to refine and improve the topics discovered by topic models. ITM offers a paradigm for non-specialist consumers of machine learning algorithms to refine models to better reflect their interests and needs. We demonstrated that even novice users are able to understand and build constraints using a simple interface and that their constraints can improve the model's ability to capture the latent structure of a corpus.

As presented here, the technique for incorporating constraints is closely tied to inference with Gibbs sampling. However, most inference techniques are essentially optimization problems. As long as it is possible to define a transition on the state space that moves from one less-constrained model to another more-constrained model, other inference procedures can also be used.

We hope to engage these algorithms with more sophisticated users than those on Mechanical Turk to measure how these models can help them better explore and understand large, uncurated data sets. As we learn their needs, we can add more avenues for interacting with topic models.

Acknowledgements

We would like to thank the anonymous reviewers, Edmund Talley, Jonathan Chang, and Philip Resnik for their helpful comments on drafts of this paper. This work was supported by NSF grant #0705832. Jordan Boyd-Graber is also supported by the Army Research Laboratory through ARL Cooperative Agreement W911NF-09-2-0072 and by NSF grant #1018625. Any opinions, findings, conclusions, or recommendations expressed are the authors' and do not necessarily reflect those of the sponsors.

References

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via Dirichlet forest priors. In *Proceedings of International Conference of Machine Learning*.

David M. Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jordan Boyd-Graber and David M. Blei. 2008. Syntactic topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

Jordan Boyd-Graber and Philip Resnik. 2010. Holistic sentiment analysis across languages: Multilingual supervised latent Dirichlet allocation. In *Proceedings of Emperical Methods in Natural Language Processing*.

Jordan Boyd-Graber, David M. Blei, and Xiaojin Zhu. 2007. A topic model for word sense disambiguation. In *Proceedings of Emperical Methods in Natural Language Processing*.

Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Neural Information Processing Systems*.

Jonathan Chang. 2010. Not-so-latent Dirichlet allocation: Collapsed Gibbs sampling using human judgments. In *NAACL Workshop: Creating Speech and Language Data With Amazon'ss Mechanical Turk*.

Hal Daumé III. 2009. Markov random topic fields. In *Proceedings of Artificial Intelligence and Statistics*.

Laura Dietz, Steffen Bickel, and Tobias Scheffer. 2007. Unsupervised prediction of citation influences. In *Proceedings of International Conference of Machine Learning*.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl 1):5228–5235.

Amit Gruber, Michael Rosen-Zvi, and Yair Weiss. 2007. Hidden topic Markov models. In *Artificial Intelligence and Statistics*.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18.

Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *Proceedings of Uncertainty in Artificial Intelligence*.

Thomas K. Landauer, Danielle S. McNamara, Dennis S. Marynick, and Walter Kintsch, editors. 2006. *Probabilistic Topic Models*. Laurence Erlbaum.

Wei-Hao Lin, Theresa Wilson, Janyce Wiebe, and Alexander Hauptmann. 2006. Which side are you on? identifying perspectives at the document and sentence levels. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.

Edward Loper and Steven Bird. 2002. NLTK: the natural language toolkit. In *Tools and methodologies for teaching*.

Radford M. Neal. 1993. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Michael Paul and Roxana Girju. 2010. A two-dimensional topic-aspect model for discovering multi-faceted topics. In *Association for the Advancement of Artificial Intelligence*.

James Petterson, Smola Alex, Tiberio Caetano, Wray Buntine, and Narayanamurthy Shravan. 2010. Word features for latent Dirichlet allocation. In *Neural Information Processing Systems*.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of Emperical Methods in Natural Language Processing*.

Fergus Rob, Li Fei-Fei, Perona Pietro, and Zisserman Andrew. 2005. Learning object categories from Google's image search. In *International Conference on Computer Vision*.

Michal Rosen-Zvi, Thomas L. Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *Proceedings of Uncertainty in Artificial Intelligence*.

Suyash Shringarpure and Eric P. Xing. 2008. mStruct: a new admixture model for inference of population structure in light of both genetic admixing and allele mutations. In *Proceedings of International Conference of Machine Learning*.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and fast—but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of Emperical Methods in Natural Language Processing*.

Hanna Wallach, David Mimno, and Andrew McCallum. 2009. Rethinking LDA: Why priors matter. In *Proceedings of Advances in Neural Information Processing Systems*.

Hanna M. Wallach. 2006. Topic modeling: Beyond bag-of-words. In *Proceedings of International Conference of Machine Learning*.

Limin Yao, David Mimno, and Andrew McCallum. 2009. Efficient methods for topic model inference on streaming document collections. In *Knowledge Discovery and Data Mining*.

Faster and Smaller N-Gram Language Models

Adam Pauls Dan Klein
Computer Science Division
University of California, Berkeley
{adpauls,klein}@cs.berkeley.edu

Abstract

N-gram language models are a major resource bottleneck in machine translation. In this paper, we present several language model implementations that are both highly compact and fast to query. Our fastest implementation is as fast as the widely used SRILM while requiring only 25% of the storage. Our most compact representation can store all 4 billion n-grams and associated counts for the Google n-gram corpus in 23 bits per n-gram, the most compact lossless representation to date, and even more compact than recent *lossy* compression techniques. We also discuss techniques for improving query speed during decoding, including a simple but novel language model caching technique that improves the query speed of our language models (and SRILM) by up to 300%.

1 Introduction

For modern statistical machine translation systems, language models must be both fast and compact. The largest language models (LMs) can contain as many as several hundred *billion* n-grams (Brants et al., 2007), so storage is a challenge. At the same time, decoding a single sentence can trigger hundreds of thousands of queries to the language model, so speed is also critical. As always, trade-offs exist between time, space, and accuracy, with many recent papers considering small-but-approximate noisy LMs (Chazelle et al., 2004; Guthrie and Hepple, 2010) or small-but-slow compressed LMs (Germann et al., 2009).

In this paper, we present several *lossless* methods for compactly but efficiently storing large LMs in memory. As in much previous work (Whittaker and Raj, 2001; Hsu and Glass, 2008), our methods are conceptually based on tabular trie encodings wherein each n-gram key is stored as the concatenation of one word (here, the last) and an offset encoding the remaining words (here, the context). After presenting a bit-conscious basic system that typifies such approaches, we improve on it in several ways. First, we show how the last word of each entry can be implicitly encoded, almost entirely eliminating its storage requirements. Second, we show that the deltas between adjacent entries can be efficiently encoded with simple variable-length encodings. Third, we investigate block-based schemes that minimize the amount of compressed-stream scanning during lookup.

To speed up our language models, we present two approaches. The first is a front-end cache. Caching itself is certainly not new to language modeling, but because well-tuned LMs are essentially lookup tables to begin with, naive cache designs only speed up slower systems. We present a direct-addressing cache with a fast key identity check that speeds up our systems (or existing fast systems like the widely-used, speed-focused SRILM) by up to 300%.

Our second speed-up comes from a more fundamental change to the language modeling interface. Where classic LMs take word tuples and produce counts or probabilities, we propose an LM that takes a word-and-context encoding (so the context need not be re-looked up) and returns both the probability and also the context encoding for the *suffix* of the original query. This setup substantially accelerates the scrolling queries issued by decoders, and also exploits language model state equivalence (Li and Khudanpur, 2008).

Overall, we are able to store the 4 billion n-grams of the Google Web1T (Brants and Franz, 2006) cor-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 258–267,
Portland, Oregon, June 19-24, 2011.

pus, with associated counts, in 10 GB of memory, which is smaller than state-of-the-art *lossy* language model implementations (Guthrie and Hepple, 2010), and significantly smaller than the best published lossless implementation (Germann et al., 2009). We are also able to simultaneously outperform SRILM in both total size and speed. Our LM toolkit, which is implemented in Java and compatible with the standard ARPA file formats, is available on the web.[1]

2 Preliminaries

Our goal in this paper is to provide data structures that map n-gram keys to values, i.e. probabilities or counts. Maps are fundamental data structures and generic implementations of mapping data structures are readily available. However, because of the sheer number of keys and values needed for n-gram language modeling, generic implementations do not work efficiently "out of the box." In this section, we will review existing techniques for encoding the keys and values of an n-gram language model, taking care to account for every bit of memory required by each implementation.

To provide absolute numbers for the storage requirements of different implementations, we will use the Google Web1T corpus as a benchmark. This corpus, which is on the large end of corpora typically employed in language modeling, is a collection of nearly 4 billion n-grams extracted from over a trillion tokens of English text, and has a vocabulary of about 13.5 million words.

2.1 Encoding Values

In the Web1T corpus, the most frequent n-gram occurs about 95 billion times. Storing this count explicitly would require 37 bits, but, as noted by Guthrie and Hepple (2010), the corpus contains only about 770 000 *unique* counts, so we can enumerate all counts using only 20 bits, and separately store an array called the *value rank array* which converts the rank encoding of a count back to its raw count. The additional array is small, requiring only about 3MB, but we save 17 bits per n-gram, reducing value storage from around 16GB to about 9GB for Web1T.

We can rank encode probabilities and back-offs in the same way, allowing us to be agnostic to whether we encode counts, probabilities and/or back-off weights in our model. In general, the number of bits per value required to encode all value ranks for a given language model will vary – we will refer to this variable as v .

2.2 Trie-Based Language Models

The data structure of choice for the majority of modern language model implementations is a *trie* (Fredkin, 1960). Tries or variants thereof are implemented in many LM tool kits, including SRILM (Stolcke, 2002), IRSTLM (Federico and Cettolo, 2007), CMU SLM (Whittaker and Raj, 2001), and MIT LM (Hsu and Glass, 2008). Tries represent collections of n-grams using a tree. Each node in the tree encodes a word, and paths in the tree correspond to n-grams in the collection. Tries ensure that each n-gram prefix is represented only once, and are very efficient when n-grams share common prefixes. Values can also be stored in a trie by placing them in the appropriate nodes.

Conceptually, trie nodes can be implemented as records that contain two entries: one for the word in the node, and one for either a pointer to the parent of the node or a list of pointers to children. At a low level, however, naive implementations of tries can waste significant amounts of space. For example, the implementation used in SRILM represents a trie node as a C `struct` containing a 32-bit integer representing the word, a 64-bit memory[2] pointer to the list of children, and a 32-bit floating point number representing the value stored at a node. The total storage for a node alone is 16 bytes, with additional overhead required to store the list of children. In total, the most compact implementation in SRILM uses 33 bytes per n-gram of storage, which would require around 116 GB of memory to store Web1T.

While it is simple to implement a trie node in this (already wasteful) way in programming languages that offer low-level access to memory allocation like C/C++, the situation is even worse in higher level programming languages. In Java, for example, C-style `structs` are not available, and records are most naturally implemented as objects that carry an additional 64 bits of overhead.

[1] `http://code.google.com/p/berkeleylm/`

[2] While 32-bit architectures are still in use today, their limited address space is insufficient for modern language models and we will assume all machines use a 64-bit architecture.

Despite its relatively large storage requirements, the implementation employed by SRILM is still widely in use today, largely because of its speed – to our knowledge, SRILM is the fastest freely available language model implementation. We will show that we can achieve access speeds comparable to SRILM but using only 25% of the storage.

2.3 Implicit Tries

A more compact implementation of a trie is described in Whittaker and Raj (2001). In their implementation, nodes in a trie are represented implicitly as entries in an array. Each entry encodes a word with enough bits to index all words in the language model (24 bits for Web1T), a quantized value, and a 32-bit[3] offset that encodes the contiguous block of the array containing the children of the node. Note that 32 bits is sufficient to index all n-grams in Web1T; for larger corpora, we can always increase the size of the offset.

Effectively, this representation replaces system-level memory pointers with offsets that act as logical pointers that can reference other entries in the array, rather than arbitrary bytes in RAM. This representation saves space because offsets require fewer bits than memory pointers, but more importantly, it permits straightforward implementation in any higher-level language that provides access to arrays of integers.[4]

2.4 Encoding n-grams

Hsu and Glass (2008) describe a variant of the implicit tries of Whittaker and Raj (2001) in which each node in the trie stores the prefix (i.e. parent). This representation has the property that we can refer to each n-gram $\mathbf{w}_1^n$ by its last word w_n and the offset $c(\mathbf{w}_1^{n-1})$ of its prefix $\mathbf{w}_1^{n-1}$, often called the *context*. At a low-level, we can efficiently encode this pair $(w_n, c(\mathbf{w}_1^{n-1}))$ as a single 64-bit integer, where the first 24 bits refer to w_n and the last 40 bits encode $c(\mathbf{w}_1^{n-1})$. We will refer to this encoding as a *context encoding*.

Note that typically, n-grams are encoded in tries in the reverse direction (first-rest instead of last-rest), which enables a more efficient computation of back-offs. In our implementations, we found that the speed improvement from switching to a first-rest encoding and implementing more efficient queries was modest. However, as we will see in Section 4.2, the last-rest encoding allows us to exploit the scrolling nature of queries issued by decoders, which results in speedups that far outweigh those achieved by reversing the trie.

3 Language Model Implementations

In the previous section, we reviewed well-known techniques in language model implementation. In this section, we combine these techniques to build simple data structures in ways that are to our knowledge novel, producing language models with state-of-the-art memory requirements and speed. We will also show that our data structures can be very effectively compressed by implicitly encoding the word w_n, and further compressed by applying a variable-length encoding on context deltas.

3.1 Sorted Array

A standard way to implement a map is to store an array of key/value pairs, sorted according to the key. Lookup is carried out by performing binary search on a key. For an n-gram language model, we can apply this implementation with a slight modification: we need n sorted arrays, one for each n-gram order. We construct keys $(w_n, c(\mathbf{w}_1^{n-1}))$ using the context encoding described in the previous section, where the context offsets c refer to entries in the sorted array of $(n-1)$-grams. This data structure is shown graphically in Figure 1.

Because our keys are sorted according to their context-encoded representation, we cannot straightforwardly answer queries about an n-gram $\mathbf{w}$ without first determining its context encoding. We can do this efficiently by building up the encoding incrementally: we start with the context offset of the unigram w_1, which is simply its integer representation, and use that to form the context encoding of the bigram $\mathbf{w}_1^2 = (w_2, c(w_1))$. We can find the offset of

[3] The implementation described in the paper represents each 32-bit integer compactly using only 16 bits, but this representation is quite inefficient, because determining the full 32-bit offset requires a binary search in a look up table.

[4] Typically, programming languages only provide support for arrays of *bytes*, not bits, but it is of course possible to simulate arrays with arbitrary numbers of bits using byte arrays and bit manipulation.

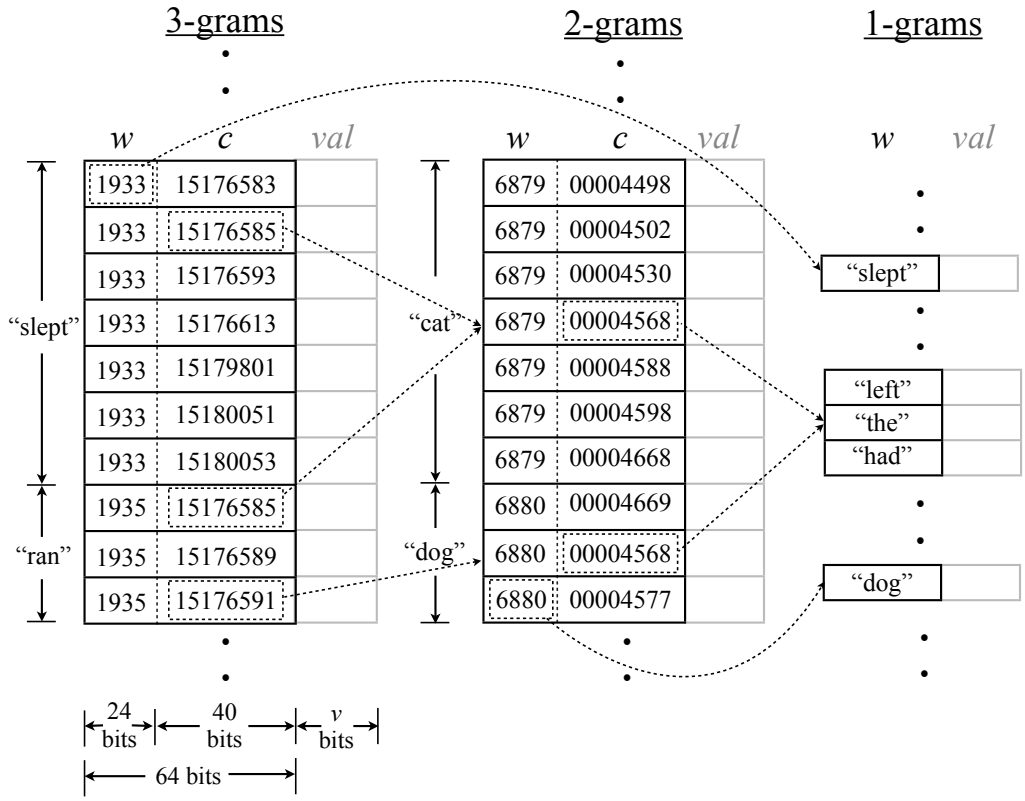


Figure 1: Our SORTED implementation of a trie. The dotted paths correspond to "the cat slept", "the cat ran", and "the dog ran". Each node in the trie is an entry in an array with 3 parts: w represents the word at the node; val represents the (rank encoded) value; and c is an offset in the array of $n-1$ grams that represents the parent (prefix) of a node. Words are represented as offsets in the unigram array.

the bigram using binary search, and form the context encoding of the trigram, and so on. Note, however, that if our queries arrive in context-encoded form, queries are faster since they involve only one binary search in the appropriate array. We will return to this later in Section 4.2

This implementation, SORTED, uses 64 bits for the integer-encoded keys and v bits for the values. Lookup is linear in the length of the key and logarithmic in the number of n-grams. For Web1T ($v = 20$), the total storage is 10.5 bytes/n-gram or about 37GB.

3.2 Hash Table

Hash tables are another standard way to implement associative arrays. To enable the use of our context encoding, we require an implementation in which we can refer to entries in the hash table via array offsets. For this reason, we use an *open address* hash map that uses linear probing for collision resolution.

As in the sorted array implementation, in order to insert an n-gram $\mathbf{w}_1^n$ into the hash table, we must form its context encoding incrementally from the offset of w_1. However, unlike the sorted array implementation, at query time, we only need to be able to check equality between the query key $\mathbf{w}_1^n = (w_n, c(\mathbf{w}_1^{n-1}))$ and a key $\mathbf{w}_1'^n = (w'_n, c(\mathbf{w}_1'^{n-1}))$ in the table. Equality can easily be checked by first checking if $w_n = w'_n$, then recursively checking equality between $\mathbf{w}_1^{n-1}$ and $\mathbf{w}_1'^{n-1}$, though again, equality is even faster if the query is already context-encoded.

This HASH data structure also uses 64 bits for integer-encoded keys and v bits for values. However, to avoid excessive hash collisions, we also allocate additional empty space according to a user-defined parameter that trades off speed and time – we used about 40% extra space in our experiments. For Web1T, the total storage for this implementation is 15 bytes/n-gram or about 53 GB total.

Look up in a hash map is linear in the length of an n-gram and constant with respect to the number

of n-grams. Unlike the sorted array implementation, the hash table implementation also permits efficient insertion and deletion, making it suitable for stream-based language models (Levenberg and Osborne, 2009).

3.3 Implicitly Encoding w_n

The context encoding we have used thus far still wastes space. This is perhaps most evident in the sorted array representation (see Figure 1): all n-grams ending with a particular word w_i are stored contiguously. We can exploit this redundancy by storing only the context offsets in the main array, using as many bits as needed to encode all context offsets (32 bits for Web1T). In auxiliary arrays, one for each n-gram order, we store the beginning and end of the range of the trie array in which all (w_i, c) keys are stored for each w_i. These auxiliary arrays are negligibly small – we only need to store $2n$ offsets for each word.

The same trick can be applied in the hash table implementation. We allocate contiguous blocks of the main array for n-grams which all share the same last word w_i, and distribute keys within those ranges using the hashing function.

This representation reduces memory usage for keys from 64 bits to 32 bits, reducing overall storage for Web1T to 6.5 bytes/n-gram for the sorted implementation and 9.1 bytes for the hashed implementation, or about 23GB and 32GB in total. It also increases query speed in the sorted array case, since to find (w_i, c), we only need to search the range of the array over which w_i applies. Because this implicit encoding reduces memory usage without a performance cost, we will assume its use for the rest of this paper.

3.4 A Compressed Implementation

3.4.1 Variable-Length Coding

The distribution of value ranks in language modeling is Zipfian, with far more n-grams having low counts than high counts. If we ensure that the value rank array sorts raw values by descending order of frequency, then we expect that small ranks will occur much more frequently than large ones, which we can exploit with a variable-length encoding.

To compress n-grams, we can exploit the context encoding of our keys. In Figure 2, we show a portion

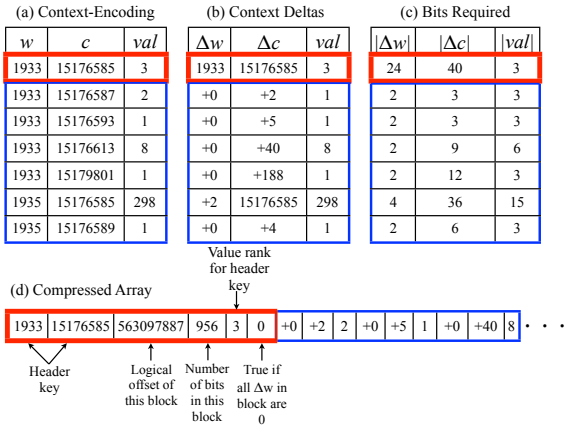


Figure 2: Compression using variable-length encoding. (a) A snippet of an (uncompressed) context-encoded array. (b) The context and word deltas. (c) The number of bits required to encode the context and word deltas as well as the value ranks. Word deltas use variable-length block coding with $k = 1$, while context deltas and value ranks use $k = 2$. (d) A snippet of the compressed encoding array. The header is outlined in bold.

of the key array used in our sorted array implementation. While we have already exploited the fact that the 24 word bits repeat in the previous section, we note here that consecutive context offsets tend to be quite close together. We found that for 5-grams, the median difference between consecutive offsets was about 50, and 90% of offset deltas were smaller than 10000. By using a variable-length encoding to represent these deltas, we should require far fewer than 32 bits to encode context offsets.

We used a very simple variable-length coding to encode offset deltas, word deltas, and value ranks. Our encoding, which is referred to as "variable-length block coding" in Boldi and Vigna (2005), works as follows: we pick a (configurable) radix $r = 2^k$. To encode a number m, we determine the number of digits d required to express m in base r. We write d in unary, i.e. $d - 1$ zeroes followed by a one. We then write the d digits of m in base r, each of which requires k bits. For example, using $k = 2$, we would encode the decimal number 7 as 010111. We can choose k separately for deltas and value indices, and also tune these parameters to a given language model.

We found this encoding outperformed other standard prefix codes, including Golomb codes (Golomb, 1966; Church et al., 2007)

and Elias γ and δ codes. We also experimented with the ζ codes of Boldi and Vigna (2005), which modify variable-length block codes so that they are optimal for certain power law distributions. We found that ζ codes performed no better than variable-length block codes and were slightly more complex. Finally, we found that Huffman codes outperformed our encoding slightly, but came at a much higher computational cost.

3.4.2 Block Compression

We could in principle compress the entire array of key/value pairs with the encoding described above, but this would render binary search in the array impossible: we cannot jump to the mid-point of the array since in order to determine what key lies at a particular point in the compressed bit stream, we would need to know the entire history of offset deltas.

Instead, we employ block compression, a technique also used by Harb et al. (2009) for smaller language models. In particular, we compress the key/value array in blocks of 128 bytes. At the beginning of the block, we write out a header consisting of: an explicit 64-bit key that begins the block; a 32-bit integer representing the offset of the header key in the uncompressed array;[5] the number of bits of compressed data in the block; and the variable-length encoding of the value rank of the header key. The remainder of the block is filled with as many compressed key/value pairs as possible. Once the block is full, we start a new block. See Figure 2 for a depiction.

When we encode an offset delta, we store the delta of the word portion of the key separately from the delta of the context offset. When an entire block shares the same word portion of the key, we set a single bit in the header that indicates that we do not encode any word deltas.

To find a key in this compressed array, we first perform binary search over the header blocks (which are predictably located every 128 bytes), followed by a linear search within a compressed block.

Using $k = 6$ for encoding offset deltas and $k = 5$ for encoding value ranks, this COMPRESSED implementation stores Web1T in less than 3 bytes per n-gram, or about 10.2GB in total. This is about 6GB less than the storage required by Germann et al. (2009), which is the best published lossless compression to date.

[5] We need this because n-grams refer to their contexts using array offsets.

4 Speeding up Decoding

In the previous section, we provided compact and efficient implementations of associative arrays that allow us to query a value for an arbitrary n-gram. However, decoders do not issue language model requests at random. In this section, we show that language model requests issued by a standard decoder exhibit two patterns we can exploit: they are highly *repetitive*, and also exhibit a *scrolling* effect.

4.1 Exploiting Repetitive Queries

In a simple experiment, we recorded all of the language model queries issued by the Joshua decoder (Li et al., 2009) on a 100 sentence test set. Of the 31 million queries, only about 1 million were unique. Therefore, we expect that keeping the results of language model queries in a cache should be effective at reducing overall language model latency.

To this end, we added a very simple cache to our language model. Our cache uses an array of key/value pairs with size fixed to $2^b - 1$ for some integer b (we used 24). We use a b-bit hash function to compute the address in an array where we will always place a given n-gram and its fully computed language model score. Querying the cache is straightforward: we check the address of a key given by its b-bit hash. If the key located in the cache array matches the query key, then we return the value stored in the cache. Otherwise, we fetch the language model probability from the language model and place the new key and value in the cache, evicting the old key in the process. This scheme is often called a *direct-mapped* cache because each key has exactly one possible address.

Caching n-grams in this way reduces overall latency for two reasons: first, lookup in the cache is extremely fast, requiring only a single evaluation of the hash function, one memory lookup to find the cache key, and one equality check on the key. In contrast, even our fastest (HASH) implementation may have to perform multiple memory lookups and equality checks in order to resolve collisions. Second, when calculating the probability for an n-gram

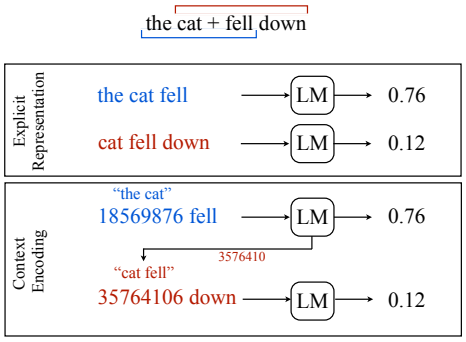


Figure 3: Queries issued when scoring trigrams that are created when a state with LM context "the cat" combines with "fell down". In the standard explicit representation of an n-gram as list of words, queries are issued atomically to the language model. When using a context-encoding, a query from the n-gram "the cat fell" returns the context offset of "cat fell", which speeds up the query of "cat fell down".

not in the language model, language models with back-off schemes must in general perform multiple queries to fetch the necessary back-off information. Our cache retains the full result of these calculations and thus saves additional computation.

Federico and Cettolo (2007) also employ a cache in their language model implementation, though based on traditional hash table cache with linear probing. Unlike our cache, which is of fixed size, their cache must be cleared after decoding a sentence. We would not expect a large performance increase from such a cache for our faster models since our HASH implementation is *already* a hash table with linear probing. We found in our experiments that a cache using linear probing provided marginal performance increases of about 40%, largely because of cached back-off computation, while our simpler cache increases performance by about 300% even over our HASH LM implementation. More timing results are presented in Section 5.

4.2 Exploiting Scrolling Queries

Decoders with integrated language models (Och and Ney, 2004; Chiang, 2005) score partial translation hypotheses in an incremental way. Each partial hypothesis maintains a language model context consisting of at most $n-1$ target-side words. When we combine two language model contexts, we create several new n-grams of length of n, each of which generate a query to the language model. These new

WMT2010	
Order	#n-grams
1gm	4,366,395
2gm	61,865,588
3gm	123,158,761
4gm	217,869,981
5gm	269,614,330
Total	676,875,055

WEB1T	
Order	#n-grams
1gm	13,588,391
2gm	314,843,401
3gm	977,069,902
4gm	1,313,818,354
5gm	1,176,470,663
Total	3,795,790,711

Table 1: Sizes of the two language models used in our experiments.

n-grams exhibit a *scrolling* effect, shown in Figure 3: the $n-1$ suffix words of one n-gram form the $n-1$ prefix words of the next.

As discussed in Section 3, our LM implementations can answer queries about context-encoded n-grams faster than explicitly encoded n-grams. With this in mind, we augment the values stored in our language model so that for a key $(w_n, c(\mathbf{w}_1^{n-1}))$, we store the offset of the *suffix* $c(\mathbf{w}_2^n)$ as well as the normal counts/probabilities. Then, rather than represent the LM context in the decoder as an explicit list of words, we can simply store context offsets. When we query the language model, we get back both a language model score and context offset $c(\hat{\mathbf{w}}_{\mathbf{1}}^{\mathbf{n-1}})$, where $\hat{\mathbf{w}}_{\mathbf{1}}^{\mathbf{n-1}}$ is the the longest suffix of $\mathbf{w}_1^{n-1}$ contained in the language model. We can then quickly form the context encoding of the next query by simply concatenating the new word with the offset $c(\hat{\mathbf{w}}_1^{n-1})$ returned from the previous query.

In addition to speeding up language model queries, this approach also automatically supports an equivalence of LM states (Li and Khudanpur, 2008): in standard back-off schemes, whenever we compute the probability for an n-gram $(w_n, c(\mathbf{w}_1^{n-1}))$ when $\mathbf{w}_1^{n-1}$ is not in the language model, the result will be the same as the result of the query $(w_n, c(\hat{\mathbf{w}}_1^{n-1}))$. It is therefore only necessary to store as much of the context as the language model contains instead of all $n-1$ words in the context. If a decoder maintains LM states using the context offsets returned by our language model, then the decoder will automatically exploit this equivalence and the size of the search space will be reduced. This same effect is exploited explicitly by some decoders (Li and Khudanpur, 2008).

WMT2010

LM Type	bytes/ key	bytes/ value	bytes/ n-gram	Total Size
SRILM-H	–	–	42.2	26.6G
SRILM-S	–	–	33.5	21.1G
HASH	5.6	6.0	11.6	7.5G
SORTED	4.0	4.5	8.5	5.5G
TPT	–	–	7.5**	4.7G**
COMPRESSED	**2.1**	**3.8**	**5.9**	**3.7**G

Table 2: Memory usages of several language model implementations on the WMT2010 language model. A ** indicates that the storage in bytes per n-gram is reported for a different language model of comparable size, and the total size is thus a rough projection.

5 Experiments

5.1 Data

To test our LM implementations, we performed experiments with two different language models. Our first language model, WMT2010, was a 5-gram Kneser-Ney language model which stores probability/back-off pairs as values. We trained this language model on the English side of all French-English corpora provided[6] for use in the WMT 2010 workshop, about 2 billion tokens in total. This data was tokenized using the `tokenizer.perl` script provided with the data. We trained the language model using SRILM. We also extracted a count-based language model, WEB1T, from the Web1T corpus (Brants and Franz, 2006). Since this data is provided as a collection of 1- to 5-grams and associated counts, we used this data without further preprocessing. The make up of these language models is shown in Table 1.

5.2 Compression Experiments

We tested our three implementations (HASH, SORTED, and COMPRESSED) on the WMT2010 language model. For this language model, there are about 80 million unique probability/back-off pairs, so $v \approx 36$. Note that here v includes both the cost per key of storing the value rank as well as the (amortized) cost of storing two 32 bit floating point numbers (probability and back-off) for each unique value. The results are shown in Table 2.

[6] `www.statmt.org/wmt10/translation-task.html`

WEB1T

LM Type	bytes/ key	bytes/ value	bytes/ n-gram	Total Size
Gzip	–	–	7.0	24.7G
T-MPHR†	–	–	3.0	10.5G
COMPRESSED	1.3	1.6	2.9	10.2G

Table 3: Memory usages of several language model implementations on the WEB1T. A † indicates lossy compression.

We compare against three baselines. The first two, SRILM-H and SRILM-S, refer to the hash table- and sorted array-based trie implementations provided by SRILM. The third baseline is the Tightly-Packed Trie (TPT) implementation of Germann et al. (2009). Because this implementation is not freely available, we use their published memory usage in bytes per n-gram on a language model of similar size and project total usage.

The memory usage of all of our models is considerably smaller than SRILM – our HASH implementation is about 25% the size of SRILM-H, and our SORTED implementation is about 25% the size of SRILM-S. Our COMPRESSED implementation is also smaller than the state-of-the-art compressed TPT implementation.

In Table 3, we show the results of our COMPRESSED implementation on WEB1T and against two baselines. The first is compression of the ASCII text count files using `gzip`, and the second is the Tiered Minimal Perfect Hash (T-MPHR) of Guthrie and Hepple (2010). The latter is a *lossy* compression technique based on Bloomier filters (Chazelle et al., 2004) and additional variable-length encoding that achieves the best published compression of WEB1T to date. Our COMPRESSED implementation is even smaller than T-MPHR, despite using a *lossless* compression technique. Note that since T-MPHR uses a lossy encoding, it is possible to reduce the storage requirements arbitrarily at the cost of additional errors in the model. We quote here the storage required when keys[7] are encoded using 12-bit hash codes, which gives a false positive rate of about 2^{-12} =0.02%.

[7] Guthrie and Hepple (2010) also report additional savings by quantizing values, though we could perform the same quantization in our storage scheme.

LM Type	No Cache	Cache	Size
COMPRESSED	9264±73ns	565±7ns	3.7G
SORTED	1405±50ns	243±4ns	5.5G
HASH	495±10ns	179±6ns	7.5G
SRILM-H	428±5ns	159±4ns	26.6G
HASH+SCROLL	323±5ns	139±6ns	10.5G

Table 4: Raw query speeds of various language model implementations. Times were averaged over 3 runs on the same machine. For HASH+SCROLL, all queries were issued to the decoder in context-encoded form, which speeds up queries that exhibit scrolling behaviour. Note that memory usage is higher than for HASH because we store suffix offsets along with the values for an n-gram.

LM Type	No Cache	Cache	Size
COMPRESSED	9880±82s	1547±7s	3.7G
SRILM-H	1120±26s	938±11s	26.6G
HASH	1146±8s	943±16s	7.5G

Table 5: Full decoding times for various language model implementations. Our HASH LM is as fast as SRILM while using 25% of the memory. Our caching also reduces total decoding time by about 20% for our fastest models and speeds up COMPRESSED by a factor of 6. Times were averaged over 3 runs on the same machine.

5.3 Timing Experiments

We first measured pure query speed by logging all LM queries issued by a decoder and measuring the time required to query those n-grams in isolation. We used the the Joshua decoder[8] with the WMT2010 model to generate queries for the first 100 sentences of the French 2008 News test set. This produced about 30 million queries. We measured the time[9] required to perform each query in order with and without our direct-mapped caching, not including any time spent on file I/O.

The results are shown in Table 4. As expected, HASH is the fastest of our implementations, and comparable[10] in speed to SRILM-H, but using significantly less space. SORTED is slower but of course more memory efficient, and COMPRESSED is the slowest but also the most compact representation. In HASH+SCROLL, we issued queries to the language model using the context encoding, which speeds up queries substantially. Finally, we note that our direct-mapped cache is very effective. The query speed of all models is boosted substantially. In particular, our COMPRESSED implementation with caching is nearly as fast as SRILM-H without caching, and even the already fast HASH implementation is 300% faster in raw query speed with caching enabled.

We also measured the effect of LM performance on overall decoder performance. We modified Joshua to optionally use our LM implementations during decoding, and measured the time required to decode all 2051 sentences of the 2008 News test set. The results are shown in Table 5. Without caching, SRILM-H and HASH were comparable in speed, while COMPRESSED introduces a performance penalty. With caching enabled, overall decoder speed is improved for both HASH and SRILM-H, while the COMPRESSED implementation is only about 50% slower that the others.

6 Conclusion

We have presented several language model implementations which are state-of-the-art in both size and speed. Our experiments have demonstrated improvements in query speed over SRILM and compression rates against state-of-the-art lossy compression. We have also described a simple caching technique which leads to performance increases in overall decoding time.

Acknowledgements

This work was supported by a Google Fellowship for the first author and by BBN under DARPA contract HR0011-06-C-0022. We would like to thank David Chiang, Zhifei Li, and the anonymous reviewers for their helpful comments.

[8] We used a grammar trained on all French-English data provided for WMT 2010 using the `make` scripts provided at `http://sourceforge.net/projects/joshua/files/joshua/1.3/wmt2010-experiment.tgz/download`

[9] All experiments were performed on an Amazon EC2 High-Memory Quadruple Extra Large instance, with an Intel Xeon X5550 CPU running at 2.67GHz and 8 MB of cache.

[10] Because we implemented our LMs in Java, we issued queries to SRILM via Java Native Interface (JNI) calls, which introduces a performance overhead. When called natively, we found that SRILM was about 200 ns/query faster. Unfortunately, it is not completely fair to compare our LMs against either of these numbers: although the JNI overhead slows down SRILM, implementing our LMs in Java instead of C++ slows down our LMs. In the tables, we quote times which include the JNI overhead, since this reflects the true cost to a decoder written in Java (e.g. Joshua).

References

Paolo Boldi and Sebastiano Vigna. 2005. Codes for the world wide web. *Internet Mathematics*, 2.

Thorsten Brants and Alex Franz. 2006. Google web1t 5-gram corpus, version 1. In *Linguistic Data Consortium, Philadelphia, Catalog Number LDC2006T13*.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Bernard Chazelle, Joe Kilian, Ronitt Rubinfeld, and Ayellet Tal. 2004. The Bloomier filter: an efficient data structure for static support lookup tables. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *The Annual Conference of the Association for Computational Linguistics*.

Kenneth Church, Ted Hart, and Jianfeng Gao. 2007. Compressing trigram language models with golomb coding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Marcello Federico and Mauro Cettolo. 2007. Efficient handling of n-gram language models for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*.

Edward Fredkin. 1960. Trie memory. *Communications of the ACM*, 3:490–499, September.

Ulrich Germann, Eric Joanis, and Samuel Larkin. 2009. Tightly packed tries: how to fit large models into memory, and make them load fast, too. In *Proceedings of the Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing*.

S. W. Golomb. 1966. Run-length encodings. *IEEE Transactions on Information Theory*, 12.

David Guthrie and Mark Hepple. 2010. Storing the web in memory: space efficient language models with constant time retrieval. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Boulos Harb, Ciprian Chelba, Jeffrey Dean, and Sanjay Ghemawat. 2009. Back-off language model compression. In *Proceedings of Interspeech*.

Bo-June Hsu and James Glass. 2008. Iterative language model estimation: Efficient data structure and algorithms. In *Proceedings of Interspeech*.

Abby Levenberg and Miles Osborne. 2009. Stream-based randomised language models for smt. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Zhifei Li and Sanjeev Khudanpur. 2008. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance. In *Proceedings of the Second Workshop on Syntax and Structure in Statistical Translation*.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren N. G. Thornton, Jonathan Weese, and Omar F. Zaidan. 2009. Joshua: an open source toolkit for parsing-based machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computationl Linguistics*, 30:417–449, December.

Andreas Stolcke. 2002. SRILM: An extensible language modeling toolkit. In *Proceedings of Interspeech*.

E. W. D. Whittaker and B. Raj. 2001. Quantization-based language model compression. In *Proceedings of Eurospeech*.

Learning to Win by Reading Manuals in a Monte-Carlo Framework

S.R.K. Branavan **David Silver *** **Regina Barzilay**

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{branavan, regina}@csail.mit.edu

* Department of Computer Science
University College London
d.silver@cs.ucl.ac.uk

Abstract

This paper presents a novel approach for leveraging automatically extracted textual knowledge to improve the performance of control applications such as games. Our ultimate goal is to enrich a stochastic player with high-level guidance expressed in text. Our model jointly learns to identify text that is relevant to a given game state in addition to learning game strategies guided by the selected text. Our method operates in the Monte-Carlo search framework, and learns both text analysis and game strategies based only on environment feedback. We apply our approach to the complex strategy game Civilization II using the official game manual as the text guide. Our results show that a linguistically-informed game-playing agent significantly outperforms its language-unaware counterpart, yielding a 27% absolute improvement and winning over 78% of games when playing against the built-in AI of Civilization II. [1]

1 Introduction

In this paper, we study the task of grounding linguistic analysis in control applications such as computer games. In these applications, an agent attempts to optimize a utility function (e.g., game score) by learning to select situation-appropriate actions. In complex domains, finding a winning strategy is challenging even for humans. Therefore, human players typically rely on manuals and guides that describe promising tactics and provide general advice about the underlying task. Surprisingly, such textual information has never been utilized in control algorithms despite its potential to greatly improve performance.

[1]The code, data and complete experimental setup for this work are available at http://groups.csail.mit.edu/rbg/code/civ.

> *The natural resources available where a population settles affects its ability to produce food and goods. Build your city on a plains or grassland square with a river running through it if possible.*

Figure 1: An excerpt from the user manual of the game Civilization II.

Consider for instance the text shown in Figure 1. This is an excerpt from the user manual of the game Civilization II.[2] This text describes game locations where the action "build-city" can be effectively applied. A stochastic player that does not have access to this text would have to gain this knowledge the hard way: it would repeatedly attempt this action in a myriad of states, thereby learning the characterization of promising state-action pairs based on the observed game outcomes. In games with large state spaces, long planning horizons, and high-branching factors, this approach can be prohibitively slow and ineffective. An algorithm with access to the text, however, could learn correlations between words in the text and game attributes – e.g., the word "river" and places with rivers in the game – thus leveraging strategies described in text to better select actions.

The key technical challenge in leveraging textual knowledge is to automatically extract relevant information from text and incorporate it effectively into a control algorithm. Approaching this task in a supervised framework, as is common in traditional information extraction, is inherently difficult. Since the game's state space is extremely large, and the states that will be encountered during game play cannot be known a priori, it is impractical to manually annotate the information that would be relevant to those states. Instead, we propose to learn text analysis based on a feedback signal inherent to the control application, such as game score.

[2]http://en.wikipedia.org/wiki/Civilization_II

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 268–277,
Portland, Oregon, June 19-24, 2011.

Our general setup consists of a game in a stochastic environment, where the goal of the player is to maximize a given utility function $R(s)$ at state s. We follow a common formulation that has been the basis of several successful applications of machine learning to games. The player's behavior is determined by an action-value function $Q(s, a)$ that assesses the goodness of an action a in a given state s based on the features of s and a. This function is learned based solely on the utility $R(s)$ collected via simulated game-play in a Monte-Carlo framework.

An obvious way to enrich the model with textual information is to augment the action-value function with word features in addition to state and action features. However, adding all the words in the document is unlikely to help since only a small fraction of the text is relevant for a given state. Moreover, even when the relevant sentence is known, the mapping between raw text and the action-state representation may not be apparent. This representation gap can be bridged by inducing a predicate structure on the sentence—e.g., by identifying words that describe actions, and those that describe state attributes.

In this paper, we propose a method for learning an action-value function augmented with linguistic features, while simultaneously modeling sentence relevance and predicate structure. We employ a multilayer neural network where the hidden layers represent sentence relevance and predicate parsing decisions. Despite the added complexity, all the parameters of this non-linear model can be effectively learned via Monte-Carlo simulations.

We test our method on the strategy game Civilization II, a notoriously challenging game with an immense action space.[3] As a source of knowledge for guiding our model, we use the official game manual. As a baseline, we employ a similar Monte-Carlo search based player which does not have access to textual information. We demonstrate that the linguistically-informed player significantly outperforms the baseline in terms of number of games won. Moreover, we show that modeling the deeper linguistic structure of sentences further improves performance. In full-length games, our algorithm yields a 27% improvement over a language unaware baseline, and wins over 78% of games against the built-in, hand-crafted AI of Civilization II.[4]

2 Related Work

Our work fits into the broad area of grounded language acquisition where the goal is to learn linguistic analysis from a situated context (Oates, 2001; Siskind, 2001; Yu and Ballard, 2004; Fleischman and Roy, 2005; Mooney, 2008a; Mooney, 2008b; Branavan et al., 2009; Vogel and Jurafsky, 2010). Within this line of work, we are most closely related to reinforcement learning approaches that learn language by proactively interacting with an external environment (Branavan et al., 2009; Branavan et al., 2010; Vogel and Jurafsky, 2010). Like the above models, we use environment feedback (in the form of a utility function) as the main source of supervision. The key difference, however, is in the language interpretation task itself. Previous work has focused on the interpretation of instruction text where input documents specify a set of actions to be executed in the environment. In contrast, game manuals provide high-level advice but do not directly describe the correct actions for every potential game state. Moreover, these documents are long, and use rich vocabularies with complex grammatical constructions. We do not aim to perform a comprehensive interpretation of such documents. Rather, our focus is on language analysis that is sufficiently detailed to help the underlying control task.

The area of language analysis situated in a game domain has been studied in the past (Eisenstein et al., 2009). Their method, however, is different both in terms of the target interpretation task, and the supervision signal it learns from. They aim to learn the rules of a given game, such as which moves are valid, given documents describing the rules. Our goal is more open ended, in that we aim to learn winning game strategies. Furthermore, Eisenstein et al. (2009) rely on a different source of supervision – game traces collected a priori. For complex games, like the one considered in this paper, collecting such game traces is prohibitively expensive. Therefore our approach learns by actively playing the game.

[3] Civilization II was #3 in IGN's 2007 list of top video games of all time (http://top100.ign.com/2007/ign_top_game_3.html)

[4] In this paper, we focus primarily on the linguistic aspects of our task and algorithm. For a discussion and evaluation of the non-linguistic aspects please see Branavan et al. (2011).

3 Monte-Carlo Framework for Computer Games

Our method operates within the Monte-Carlo search framework (Tesauro and Galperin, 1996), which has been successfully applied to complex computer games such as Go, Poker, Scrabble, multi-player card games, and real-time strategy games, among others (Gelly et al., 2006; Tesauro and Galperin, 1996; Billings et al., 1999; Sheppard, 2002; Schäfer, 2008; Sturtevant, 2008; Balla and Fern, 2009). Since Monte-Carlo search forms the foundation of our approach, we briefly describe it in this section.

Game Representation The game is defined by a large Markov Decision Process $\langle S, A, T, R \rangle$. Here S is the set of possible states, A is the space of legal actions, and $T(s'|s,a)$ is a stochastic state transition function where $s, s' \in S$ and $a \in A$. Specifically, a state encodes attributes of the game world, such as available resources and city locations. At each step of the game, a player executes an action a which causes the current state s to change to a new state s' according to the transition function $T(s'|s,a)$. While this function is not known a priori, the program encoding the game can be viewed as a black box from which transitions can be sampled. Finally, a given utility function $R(s) \in \mathbb{R}$ captures the likelihood of winning the game from state s (e.g., an intermediate game score).

Monte-Carlo Search Algorithm The goal of the Monte-Carlo search algorithm is to dynamically select the best action for the current state s_t. This selection is based on the results of multiple *roll-outs* which measure the outcome of a sequence of actions in a simulated game – e.g., simulations played against the game's built-in AI. Specifically, starting at state s_t, the algorithm repeatedly selects and executes actions, sampling state transitions from T. On game completion at time τ, we measure the final utility $R(s_\tau)$.[5] The actual game action is then selected as the one corresponding to the roll-out with the best final utility. See Algorithm 1 for details.

The success of Monte-Carlo search is based on its ability to make a fast, local estimate of the action quality at each step of the roll-outs. States and actions are evaluated by an *action-value function* $Q(s,a)$, which is an estimate of the expected outcome of action a in state s. This action-value function is used to guide action selection during the roll-outs. While actions are usually selected to maximize the action-value function, sometimes other actions are also randomly explored in case they are more valuable than predicted by the current estimate of $Q(s,a)$. As the accuracy of $Q(s,a)$ improves, the quality of action selection improves and vice

procedure PlayGame ()

Initialize game state to fixed starting state
$s_1 \leftarrow s_0$

for $t = 1 \dots T$ **do**

  Run N simulated games
  for $i = 1 \dots N$ **do**
    $(a_i, r_i) \leftarrow \text{SimulateGame}(s)$
  end

  Compute average observed utility for each action
  $a_t \leftarrow \arg\max_a \frac{1}{N_a} \sum_{i:a_i=a} r_i$

  Execute selected action in game
  $s_{t+1} \leftarrow T(s'|s_t, a_t)$
end

procedure SimulateGame (s_t)

for $u = t \dots \tau$ **do**

  Compute Q function approximation
  $Q(s,a) = \vec{w} \cdot \vec{f}(s,a)$

  Sample action from action-value function in ϵ-greedy fashion:
  $$a_u \sim \begin{cases} \text{uniform}(a \in A) & \text{with probability } \epsilon \\ \arg\max_a Q(s,a) & \text{otherwise} \end{cases}$$

  Execute selected action in game:
  $s_{u+1} \leftarrow T(s'|s_u, a_u)$

  if game is won or lost **break**
end

Update parameters $\vec{w}$ of $Q(s,a)$

Return action and observed utility:
return $a_t, R(s_\tau)$

Algorithm 1: The general Monte-Carlo algorithm.

[5] In general, roll-outs are run till game completion. However, if simulations are expensive as is the case in our domain, roll-outs can be truncated after a fixed number of steps.

versa, in a cycle of continual improvement (Sutton and Barto, 1998).

In many games, it is sufficient to maintain a distinct action-value for each unique state and action in a large search tree. However, when the branching factor is large it is usually beneficial to approximate the action-value function, so that the value of many related states and actions can be learned from a reasonably small number of simulations (Silver, 2009). One successful approach is to model the action-value function as a linear combination of state and action attributes (Silver et al., 2008):

$$Q(s,a) = \vec{w} \cdot \vec{f}(s,a).$$

Here $\vec{f}(s,a) \in \mathbb{R}^n$ is a real-valued feature function, and $\vec{w}$ is a weight vector. We take a similar approach here, except that our feature function includes latent structure which models language.

The parameters $\vec{w}$ of $Q(s,a)$ are learned based on feedback from the roll-out simulations. Specifically, the parameters are updated by stochastic gradient descent by comparing the current predicted $Q(s,a)$ against the observed utility at the end of each roll-out. We provide details on parameter estimation in the context of our model in Section 4.2.

The roll-outs themselves are fully guided by the action-value function. At every step of the simulation, actions are selected by an ϵ-greedy strategy: with probability ϵ an action is selected uniformly at random; otherwise the action is selected greedily to maximize the current action-value function, $\arg\max_a Q(s,a)$.

4 Adding Linguistic Knowledge to the Monte-Carlo Framework

In this section we describe how we inform the simulation-based player with information automatically extracted from text – in terms of both model structure and parameter estimation.

4.1 Model Structure

To inform action selection with the advice provided in game manuals, we modify the action-value function $Q(s,a)$ to take into account words of the document in addition to state and action information. Conditioning $Q(s,a)$ on all the words in the document is unlikely to be effective since only a small fraction of the document provides guidance relevant to the current state, while the remainder of the text is likely to be irrelevant. Since this information is not known a priori, we model the decision about a sentence's relevance to the current state as a hidden variable. Moreover, to fully utilize the information presented in a sentence, the model identifies the words that describe *actions* and those that describe *state attributes*, discriminating them from the rest of the sentence. As with the relevance decision, we model this labeling using hidden variables.

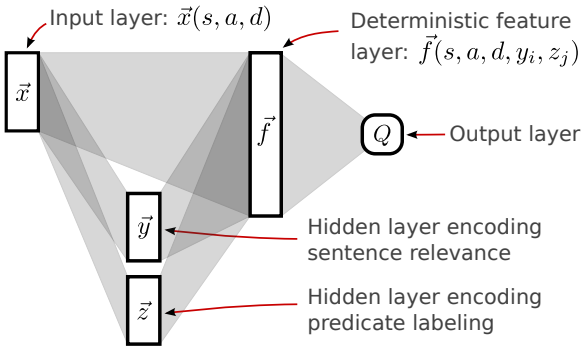


Figure 2: The structure of our model. Each rectangle represents a collection of units in a layer, and the shaded trapezoids show the connections between layers. A fixed, real-valued feature function $\vec{x}(s,a,d)$ transforms the game state s, action a, and strategy document d into the input vector $\vec{x}$. The first hidden layer contains two disjoint sets of units $\vec{y}$ and $\vec{z}$ corresponding to linguistic analyzes of the strategy document. These are softmax layers, where only one unit is active at any time. The units of the second hidden layer $\vec{f}(s,a,d,y_i,z_i)$ are a set of fixed real valued feature functions on s, a, d and the active units y_i and z_i of $\vec{y}$ and $\vec{z}$ respectively.

As shown in Figure 2, our model is a four layer neural network. The *input layer* $\vec{x}$ represents the current state s, candidate action a, and document d. The *second layer* consists of two disjoint sets of units $\vec{y}$ and $\vec{z}$ which encode the sentence-relevance and predicate-labeling decisions respectively. Each of these sets of units operates as a stochastic 1-of-n softmax selection layer (Bridle, 1990) where only a single unit is activated. The activation function for units in this layer is the standard softmax function:

$$p(y_i = 1|\vec{x}) = e^{\vec{u}_i \cdot \vec{x}} \Big/ \sum_k e^{\vec{u}_k \cdot \vec{x}},$$

where y_i is the i^{th} hidden unit of $\vec{y}$, and $\vec{u}_i$ is the weight vector corresponding to y_i. Given this acti-

vation function, the second layer effectively models sentence relevance and predicate labeling decisions via log-linear distributions, the details of which are described below.

The third *feature layer* $\vec{f}$ of the neural network is deterministically computed given the active units y_i and z_j of the softmax layers, and the values of the input layer. Each unit in this layer corresponds to a fixed feature function $f_k(s_t, a_t, d, y_i, z_j) \in \mathbb{R}$. Finally the *output layer* encodes the action-value function $Q(s, a, d)$, which now also depends on the document d, as a weighted linear combination of the units of the feature layer:

$$Q(s_t, a_t, d) = \vec{w} \cdot \vec{f},$$

where $\vec{w}$ is the weight vector.

Modeling Sentence Relevance Given a strategy document d, we wish to identify a sentence y_i that is most relevant to the current game state s_t and action a_t. This relevance decision is modeled as a log-linear distribution over sentences as follows:

$$p(y_i|s_t, a_t, d) \propto e^{\vec{u}\cdot\phi(y_i, s_t, a_t, d)}.$$

Here $\phi(y_i, s_t, a_t, d) \in \mathbb{R}^n$ is a feature function, and $\vec{u}$ are the parameters we need to estimate.

Modeling Predicate Structure Our goal here is to label the words of a sentence as either *action-description*, *state-description* or *background*. Since these word label assignments are likely to be mutually dependent, we model predicate labeling as a sequence prediction task. These dependencies do not necessarily follow the order of words in a sentence, and are best expressed in terms of a syntactic tree. For example, words corresponding to *state-description* tend to be descendants of *action-description* words. Therefore, we label words in dependency order — i.e., starting at the root of a given dependency tree, and proceeding to the leaves. This allows a word's label decision to condition on the label of the corresponding dependency tree parent.

Given sentence y_i and its dependency parse q_i, we model the distribution over predicate labels $\vec{e}_i$ as:

$$\begin{aligned} p(\vec{e}_i\,|y_i, q_i) &= \prod_j p(e_j|j, \vec{e}_{1:j-1}, y_i, q_i), \\ p(e_j|j, \vec{e}_{1:j-1}, y_i, q_i) &\propto e^{\vec{v}\cdot\psi(e_j, j, \vec{e}_{1:j-1}, y_i, q_i)}. \end{aligned}$$

Here e_j is the predicate label of the j^{th} word being labeled, and $\vec{e}_{1:j-1}$ is the partial predicate labeling constructed so far for sentence y_i.

In the second layer of the neural network, the units $\vec{z}$ represent a predicate labeling $\vec{e}_i$ of every sentence $y_i \in d$. However, our intention is to incorporate, into action-value function Q, information from only the most relevant sentence. Thus, in practice, we only perform a predicate labeling of the sentence selected by the relevance component of the model.

Given the sentence selected as relevant and its predicate labeling, the output layer of the network can now explicitly learn the correlations between textual information, and game states and actions – for example, between the word "grassland" in Figure 1, and the action of building a city. This allows our method to leverage the automatically extracted textual information to improve game play.

4.2 Parameter Estimation

Learning in our method is performed in an online fashion: at each game state s_t, the algorithm performs a simulated game roll-out, observes the outcome of the game, and updates the parameters $\vec{u}$, $\vec{v}$ and $\vec{w}$ of the action-value function $Q(s_t, a_t, d)$. These three steps are repeated a fixed number of times at each actual game state. The information from these roll-outs is used to select the actual game action. The algorithm re-learns $Q(s_t, a_t, d)$ for every new game state s_t. This specializes the action-value function to the subgame starting from s_t.

Since our model is a non-linear approximation of the underlying action-value function of the game, we learn model parameters by applying non-linear regression to the observed final utilities from the simulated roll-outs. Specifically, we adjust the parameters by stochastic gradient descent, to minimize the mean-squared error between the action-value $Q(s, a)$ and the final utility $R(s_\tau)$ for each observed game state s and action a. The resulting update to model parameters θ is of the form:

$$\begin{aligned} \Delta\theta &= -\frac{\alpha}{2}\nabla_\theta \left[R(s_\tau) - Q(s, a)\right]^2 \\ &= \alpha \left[R(s_\tau) - Q(s, a)\right] \nabla_\theta Q(s, a; \theta), \end{aligned}$$

where α is a learning rate parameter.

This minimization is performed via standard error backpropagation (Bryson and Ho, 1969; Rumelhart

et al., 1986), which results in the following online updates for the output layer parameters $\vec{w}$:

$$\vec{w} \leftarrow \vec{w} + \alpha_w \left[Q - R(s_\tau)\right] \vec{f}(s, a, d, y_i, z_j),$$

where α_w is the learning rate, and $Q = Q(s, a, d)$. The corresponding updates for the sentence relevance and predicate labeling parameters $\vec{u}$ and $\vec{v}$ are:

$$\vec{u}_i \leftarrow \vec{u}_i + \alpha_u \left[Q - R(s_\tau)\right] Q \vec{x} \left[1 - p(y_i|\cdot)\right],$$

$$\vec{v}_i \leftarrow \vec{v}_i + \alpha_v \left[Q - R(s_\tau)\right] Q \vec{x} \left[1 - p(z_i|\cdot)\right].$$

5 Applying the Model

We apply our model to playing the turn-based strategy game, Civilization II. We use the official manual [6] of the game as the source of textual strategy advice for the language aware algorithms.

Civilization II is a multi-player game set on a grid-based map of the world. Each grid location represents a tile of either land or sea, and has various resources and terrain attributes. For example, land tiles can have hills with rivers running through them. In addition to multiple cities, each player controls various units – e.g., settlers and explorers. Games are won by gaining control of the entire world map. In our experiments, we consider a two-player game of Civilization II on a grid of 1000 squares, where we play against the built-in AI player.

Game States and Actions We define the game state of Civilization II to be the map of the world, the attributes of each map tile, and the attributes of each player's cities and units. Some examples of the attributes of states and actions are shown in Figure 3. The space of possible actions for a given city or unit is known given the current game state. The actions of a player's cities and units combine to form the action space of that player. In our experiments, on average a player controls approximately 18 units, and each unit can take one of 15 actions. This results in a very large action space for the game – i.e., 10^{21}. To effectively deal with this large action space, we assume that given the state, the actions of a single unit are independent of the actions of all other units of the same player.

Utility Function The Monte-Carlo algorithm uses the utility function to evaluate the outcomes of simulated game roll-outs. In the typical application of the algorithm, the final game outcome is used as the utility function (Tesauro and Galperin, 1996). Given the complexity of Civilization II, running simulation roll-outs until game completion is impractical. The game, however, provides each player with a *game score*, which is a noisy indication of how well they are currently playing. Since we are playing a two-player game, we use the ratio of the game score of the two players as our utility function.

Map tile attributes:
- Terrain type (e.g. grassland, mountain, etc)
- Tile resources (e.g. wheat, coal, wildlife, etc)

City attributes:
- City population
- Amount of food produced

Unit attributes:
- Unit type (e.g., worker, explorer, archer, etc)
- Is unit in a city ?

$$f_1(s, a, v, \vec{e}) = \begin{cases} 1 \text{ if action=build-city} \\ \quad \& \text{ tile-has-river=true} \\ \quad \& \text{ action-words=\{build,city\}} \\ \quad \& \text{ state-words=\{river,hill\}} \\ 0 \text{ otherwise} \end{cases}$$

$$\phi_1(v, s, a) = \begin{cases} 1 \text{ if action=build-city} \\ \quad \& \text{ tile-has-river=true} \\ \quad \& \text{ words=\{build,city,river\}} \\ 0 \text{ otherwise} \end{cases}$$

$$\psi_1(e_i, i, \vec{e}_{1:i-1}, v, z_v) = \begin{cases} 1 \text{ if label=action} \\ \quad \& \text{ word-type='build'} \\ \quad \& \text{ parent-label=action} \\ 0 \text{ otherwise} \end{cases}$$

Figure 3: Example attributes of the game (box above), and features computed using the game manual and these attributes (box below).

Features The sentence relevance features $\vec{\phi}$ and the action-value function features $\vec{f}$ consider the attributes of the game state and action, and the words of the sentence. Some of these features compute text overlap between the words of the sentence, and text labels present in the game. The feature function $\vec{\psi}$ used for predicate labeling on the other hand operates only on a given sentence and its dependency parse. It computes features which are the Cartesian product of the candidate predicate label with word attributes such as type, part-of-speech tag, and dependency parse information. Overall, $\vec{f}$, $\vec{\phi}$ and $\vec{\psi}$ compute approximately 306,800, 158,500, and 7,900 features respectively. Figure 3 shows some examples of these features.

[6] www.civfanatics.com/content/civ2/reference/Civ2manual.zip

6 Experimental Setup

Datasets We use the official game manual for Civilization II as our strategy guide. This manual uses a large vocabulary of 3638 words, and is composed of 2083 sentences, each on average 16.9 words long.

Experimental Framework To apply our method to the Civilization II game, we use the game's open source implementation *Freeciv*.[7] We instrument the game to allow our method to programmatically measure the current state of the game and to execute game actions. The Stanford parser (de Marneffe et al., 2006) was used to generate the dependency parse information for sentences in the game manual.

Across all experiments, we start the game at the same initial state and run it for 100 steps. At each step, we perform 500 Monte-Carlo roll-outs. Each roll-out is run for 20 simulated game steps before halting the simulation and evaluating the outcome. For our method, and for each of the baselines, we run 200 independent games in the above manner, with evaluations averaged across the 200 runs. We use the same experimental settings across all methods, and all model parameters are initialized to zero.

The test environment consisted of typical PCs with single Intel Core i7 CPUs (4 hyper-threaded cores each), with the algorithms executing 8 simulation roll-outs in parallel. In this setup, a single game of 100 steps runs in approximately 1.5 hours.

Evaluation Metrics We wish to evaluate two aspects of our method: how well it leverages textual information to improve game play, and the accuracy of the linguistic analysis it produces. We evaluate the first aspect by comparing our method against various baselines in terms of the percentage of games won against the built-in AI of Freeciv. This AI is a fixed algorithm designed using extensive knowledge of the game, with the intention of challenging human players. As such, it provides a good open-reference baseline. Since full games can last for multiple days, we compute the percentage of games won within the first 100 game steps as our primary evaluation. To confirm that performance under this evaluation is meaningful, we also compute the percentage of full games won over 50 independent runs, where each game is run to completion.

[7] http://freeciv.wikia.com. Game version 2.2

Method	% Win	% Loss	Std. Err.
Random	0	100	—
Built-in AI	0	0	—
Game only	17.3	5.3	± 2.7
Sentence relevance	46.7	2.8	± 3.5
Full model	**53.7**	5.9	± 3.5
Random text	40.3	4.3	± 3.4
Latent variable	26.1	3.7	± 3.1

Table 1: Win rate of our method and several baselines within the first 100 game steps, while playing against the built-in game AI. Games that are neither won nor lost are still ongoing. Our model's win rate is statistically significant against all baselines except *sentence relevance*. All results are averaged across 200 independent game runs. The standard errors shown are for percentage wins.

Method	% Wins	Standard Error
Game only	45.7	± 7.0
Latent variable	62.2	± 6.9
Full model	**78.8**	± 5.8

Table 2: Win rate of our method and two baselines on 50 full length games played against the built-in AI.

7 Results

Game performance As shown in Table 1, our language aware Monte-Carlo algorithm substantially outperforms several baselines – on average winning 53.7% of all games within the first 100 steps. The dismal performance, on the other hand, of both the *random* baseline and the game's own *built-in AI* (playing against itself) is an indicator of the difficulty of the task. This evaluation is an underestimate since it assumes that any game not won within the first 100 steps is a loss. As shown in Table 2, our method wins over 78% of full length games.

To characterize the contribution of the language components to our model's performance, we compare our method against two ablative baselines. The first of these, *game-only*, does not take advantage of any textual information. It attempts to model the action value function $Q(s, a)$ only in terms of the attributes of the game state and action. The performance of this baseline – a win rate of 17.3% – effectively confirms the benefit of automatically extracted textual information in the context of our task. The second ablative baseline, *sentence-relevance*, is

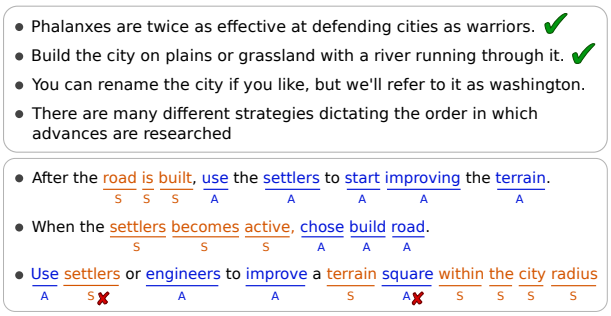


Figure 4: Examples of our method's sentence relevance and predicate labeling decisions. The box above shows two sentences (identified by check marks) which were predicted as relevant, and two which were not. The box below shows the predicted predicate structure of three sentences, with "S" indicating *state* description,"A" *action* description and *background* words unmarked. Mistakes are identified with crosses.

identical to our model, but lacks the predicate labeling component. This method wins 46.7% of games, showing that while identifying the text relevant to the current game state is essential, a deeper structural analysis of the extracted text provides substantial benefits.

One possible explanation for the improved performance of our method is that the non-linear approximation simply models game characteristics better, rather than modeling textual information. We directly test this possibility with two additional baselines. The first, *random-text*, is identical to our full model, but is given a document containing random text. We generate this text by randomly permuting the word locations of the actual game manual, thereby maintaining the document's overall statistical properties. The second baseline, *latent variable*, extends the linear action-value function $Q(s, a)$ of the *game only* baseline with a set of latent variables – i.e., it is a four layer neural network, where the second layer's units are activated only based on game information. As shown in Table 1 both of these baselines significantly underperform with respect to our model, confirming the benefit of automatically extracted textual information in the context of this task.

Sentence Relevance Figure 4 shows examples of the sentence relevance decisions produced by our method. To evaluate the accuracy of these decisions, we ideally require a ground-truth relevance annotation of the game's user manual. This however, is

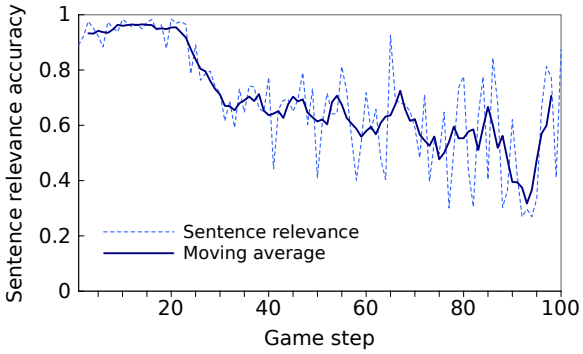


Figure 5: Accuracy of our method's sentence relevance predictions, averaged over 100 independent runs.

impractical since the relevance decision is dependent on the game context, and is hence specific to each time step of each game instance. Therefore, for the purposes of this evaluation, we modify the game manual by adding to it sentences randomly selected from the Wall Street Journal corpus (Marcus et al., 1993) – sentences that are highly unlikely to be relevant to game play. We then evaluate the accuracy with which sentences from the original manual are picked as relevant.

In this evaluation, our method achieves an average accuracy of 71.8%. Given that our model only has to differentiate between the game manual text and the Wall Street Journal, this number may seem disappointing. Furthermore, as can be seen from Figure 5, the sentence relevance accuracy varies widely as the game progresses, with a high average of 94.2% during the initial 25 game steps.

In reality, this pattern of high initial accuracy followed by a lower average is not entirely surprising: the official game manual for Civilization II is written for first time players. As such, it focuses on the initial portion of the game, providing little strategy advice relevant to subsequence game play.[8] If this is the reason for the observed sentence relevance trend, we would also expect the final layer of the neural network to emphasize game features over text features after the first 25 steps of the game. This is indeed the case, as can be seen from Figure 6.

To further test this hypothesis, we perform an experiment where the first 50 steps of the game are played using our full model, and the subsequent 50 steps are played without using any textual informa-

[8]This is reminiscent of *opening books* for games like Chess or Go, which aim to guide the player to a playable middle game.

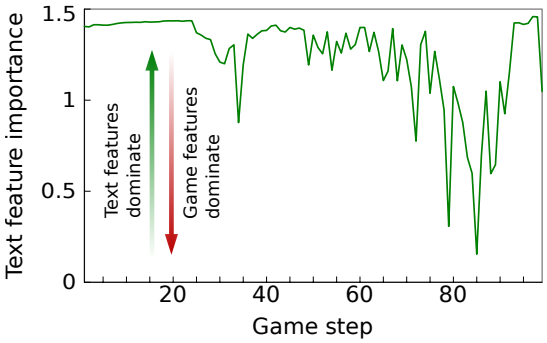


Figure 6: Difference between the norms of the text features and game features of the output layer of the neural network. Beyond the initial 25 steps of the game, our method relies increasingly on game features.

tion. This hybrid method performs as well as our full model, achieving a 53.3% win rate, confirming that textual information is most useful during the initial phase of the game. This shows that our method is able to accurately identify relevant sentences when the information they contain is most pertinent to game play.

Predicate Labeling Figure 4 shows examples of the predicate structure output of our model. We evaluate the accuracy of this labeling by comparing it against a gold-standard annotation of the game manual. Table 3 shows the performance of our method in terms of how accurately it labels words as *state*, *action* or *background*, and also how accurately it differentiates between *state* and *action* words. In addition to showing a performance improvement over the random baseline, these results display two clear trends: first, under both evaluations, labeling accuracy is higher during the initial stages of the game. This is to be expected since the model relies heavily on textual features only during the beginning of the game (see Figure 6). Second, the model clearly performs better in differentiating between state and action words, rather than in the three-way labeling.

To verify the usefulness of our method's predicate labeling, we perform a final set of experiments where predicate labels are selected uniformly at random within our full model. This random labeling results in a win rate of 44% – a performance similar to the *sentence relevance* model which uses no predicate information. This confirms that our method is able identify a predicate structure which, while noisy, provides information relevant to game play.

Method	S/A/B	S/A
Random labeling	33.3%	50.0%
Model, first 100 steps	45.1%	78.9%
Model, first 25 steps	48.0%	92.7%

Table 3: Predicate labeling accuracy of our method and a random baseline. Column "S/A/B" shows performance on the three-way labeling of words as *state*, *action* or *background*, while column "S/A" shows accuracy on the task of differentiating between state and action words.

game attribute	word
state: grassland	"city"
state: grassland	"build"
action: settlers_build_city	"city"
action: set_research	"discovery"

Figure 7: Examples of word to game attribute associations that are learned via the feature weights of our model.

Figure 7 shows examples of how this textual information is grounded in the game, by way of the associations learned between words and game attributes in the final layer of the full model.

8 Conclusions

In this paper we presented a novel approach for improving the performance of control applications by automatically leveraging high-level guidance expressed in text documents. Our model, which operates in the Monte-Carlo framework, jointly learns to identify text relevant to a given game state in addition to learning game strategies guided by the selected text. We show that this approach substantially outperforms language-unaware alternatives while learning only from environment feedback.

Acknowledgments

The authors acknowledge the support of the NSF (CAREER grant IIS-0448168, grant IIS-0835652), DARPA Machine Reading Program (FA8750-09-C-0172) and the Microsoft Research New Faculty Fellowship. Thanks to Michael Collins, Tommi Jaakkola, Leslie Kaelbling, Nate Kushman, Sasha Rush, Luke Zettlemoyer, the MIT NLP group, and the ACL reviewers for their suggestions and comments. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

R. Balla and A. Fern. 2009. UCT for tactical assault planning in real-time strategy games. In *21st International Joint Conference on Artificial Intelligence*.

Darse Billings, Lourdes Peña Castillo, Jonathan Schaeffer, and Duane Szafron. 1999. Using probabilistic knowledge and simulation to play poker. In *16th National Conference on Artificial Intelligence*, pages 697–703.

S.R.K Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of ACL*, pages 82–90.

S.R.K Branavan, Luke Zettlemoyer, and Regina Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of ACL*, pages 1268–1277.

S.R.K. Branavan, David Silver, and Regina Barzilay. 2011. Non-linear monte-carlo search in civilization ii. In *Proceedings of IJCAI*.

John S. Bridle. 1990. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *Advances in NIPS*, pages 211–217.

Arthur E. Bryson and Yu-Chi Ho. 1969. *Applied optimal control: optimization, estimation, and control*. Blaisdell Publishing Company.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC 2006*.

Jacob Eisenstein, James Clarke, Dan Goldwasser, and Dan Roth. 2009. Reading to learn: Constructing features from semantic abstracts. In *Proceedings of EMNLP*, pages 958–967.

Michael Fleischman and Deb Roy. 2005. Intentional context in situated natural language learning. In *Proceedings of CoNLL*, pages 104–111.

S. Gelly, Y. Wang, R. Munos, and O. Teytaud. 2006. Modification of UCT with patterns in Monte-Carlo Go. Technical Report 6062, INRIA.

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

Raymond J. Mooney. 2008a. Learning language from its perceptual context. In *Proceedings of ECML/PKDD*.

Raymond J. Mooney. 2008b. Learning to connect language and perception. In *Proceedings of AAAI*, pages 1598–1601.

James Timothy Oates. 2001. *Grounding knowledge in sensors: Unsupervised learning for language and planning*. Ph.D. thesis, University of Massachusetts Amherst.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323:533–536.

J. Schäfer. 2008. The UCT algorithm applied to games with imperfect information. Diploma Thesis. Otto-von-Guericke-Universität Magdeburg.

B. Sheppard. 2002. World-championship-caliber Scrabble. *Artificial Intelligence*, 134(1-2):241–275.

D. Silver, R. Sutton, and M. Müller. 2008. Sample-based learning and search with permanent and transient memories. In *25th International Conference on Machine Learning*, pages 968–975.

D. Silver. 2009. *Reinforcement Learning and Simulation-Based Search in the Game of Go*. Ph.D. thesis, University of Alberta.

Jeffrey Mark Siskind. 2001. Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *Journal of Artificial Intelligence Research*, 15:31–90.

N. Sturtevant. 2008. An analysis of UCT in multi-player games. In *6th International Conference on Computers and Games*, pages 37–49.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. The MIT Press.

G. Tesauro and G. Galperin. 1996. On-line policy improvement using Monte-Carlo search. In *Advances in Neural Information Processing 9*, pages 1068–1074.

Adam Vogel and Daniel Jurafsky. 2010. Learning to follow navigational directions. In *Proceedings of the ACL*, pages 806–814.

Chen Yu and Dana H. Ballard. 2004. On the integration of grounding language and learning objects. In *Proceedings of AAAI*, pages 488–493.

Creative Language Retrieval:
A Robust Hybrid of Information Retrieval and Linguistic Creativity

Tony Veale
School of Computer Science and Informatics,
University College Dublin,
Belfield, Dublin D4, Ireland.
Tony.Veale@UCD.ie

Abstract

Information retrieval (IR) and figurative language processing (FLP) could scarcely be more different in their treatment of language and meaning. IR views language as an open-ended set of mostly stable signs with which texts can be indexed and retrieved, focusing more on a text's potential relevance than its potential meaning. In contrast, FLP views language as a system of *unstable* signs that can be used to talk about the world in creative new ways. There is another key difference: IR is practical, scalable and robust, and in daily use by millions of casual users. FLP is neither scalable nor robust, and not yet practical enough to migrate beyond the lab. This paper thus presents a mutually beneficial hybrid of IR and FLP, one that enriches IR with new operators to enable the non-literal retrieval of creative expressions, and which also transplants FLP into a robust, scalable framework in which practical applications of linguistic creativity can be implemented.

1 Introduction

Words should not always be taken at face value. Figurative devices like metaphor can communicate far richer meanings than are evident from a superficial – and perhaps *literally* nonsensical – reading. Figurative Language Processing (FLP) thus uses a variety of special mechanisms and representations, to assign non-literal meanings not just to metaphors, but to similes, analogies, epithets, puns and other creative uses of language (see Martin, 1990; Fass, 1991; Way, 1991; Indurkhya, 1992; Fass, 1997; Barnden, 2006; Veale and Butnariu, 2010).

Computationalists have explored heterodox solutions to the procedural and representational challenges of metaphor, and FLP more generally, ranging from flexible representations (e.g. the *preference semantics* of Wilks (1978) and the *collative semantics* of Fass (1991, 1997)) to processes of cross-domain structure alignment (e.g. *structure mapping theory*; see Gentner (1983) and Falkenhainer *et al.* 1989) and even structural inversion (Veale, 2006). Though thematically related, each approach to FLP is broadly distinct, giving computational form to different cognitive demands of creative language: thus, some focus on inter-domain mappings (e.g. Gentner, 1983) while others focus more on intra-domain inference (e.g. Barnden, 2006). However, while computationally interesting, none has yet achieved the scalability or robustness needed to make a significant practical impact outside the laboratory. Moreover, such systems tend to be developed in isolation, and are rarely designed to cohere as part of a larger framework of creative reasoning (e.g. Boden, 1994).

In contrast, Information Retrieval (IR) is both scalable and robust, and its results translate easily from the laboratory into practical applications (e.g. see Salton, 1968; Van Rijsbergen, 1979). Whereas FLP derives its utility *and* its fragility from its attempts to identify deeper meanings beneath the surface, the widespread applicability of IR stems directly from its superficial treatment of language

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 278–287,
Portland, Oregon, June 19-24, 2011.

and meaning. IR does not distinguish between creative and conventional uses of language, or between literal and non-literal meanings. IR is also remarkably modular: its components are designed to work together interchangeably, from stemmers and indexers to heuristics for query expansion and document ranking. Yet, because IR treats all language as literal language, it relies on literal matching between queries and the texts that they retrieve. Documents are retrieved precisely because they contain stretches of text that literally resemble the query. This works well in the main, but it means that IR falls flat when the goal of retrieval is not to identify relevant documents but to retrieve new and creative ways of expressing a given idea. To retrieve creative language, and to be potentially surprised or inspired by the results, one needs to facilitate a non-literal relationship between queries and the texts that they match.

The complementarity of FLP and IR suggests a productive hybrid of both paradigms. If the most robust elements of FLP are used to provide new non-literal query operators for IR, then IR can be used to retrieve potentially new and creative ways of speaking about a topic from a large text collection. In return, IR can provide a stable, robust and extensible platform on which to use these operators to build FLP systems that exhibit linguistic creativity. In the next section we consider the related work on which the current realization of these ideas is founded, before presenting a specific trio of new semantic query operators in section 3. We describe three simple but practical applications of this creative IR paradigm in section 4. Empirical support for the FLP intuitions that underpin our new operators is provided in section 5. The paper concludes with some closing observations about future goals and developments in section 6.

2 Related Work and Ideas

IR works on the premise that a user can turn an information need into an effective query by anticipating the language that is used to talk about a given topic in a target collection. If the collection uses creative language in speaking about a topic, then a query must also contain the seeds of this creative language. Veale (2004) introduces the idea of *creative information retrieval* to explore how an IR system can itself provide a degree of creative anticipation, acting as a mediator between the literal specification of a meaning and the retrieval of creative articulations of this meaning. This anticipation ranges from simple re-articulation (e.g. a text may implicitly evoke "*Qur'an*" even if it only contains "*Muslim bible*") to playful allusions and epithets (e.g. the CEO of a rubber company may be punningly described as a "*rubber baron*"). A creative IR system may even anticipate out-of-dictionary words, like *chocoholic* and *sexoholic*.

Conventional IR systems use a range of query expansion techniques to automatically bolster a user's query with additional keywords or weights, to permit the retrieval of relevant texts it might not otherwise match (e.g. Vernimb, 1977; Voorhees, 1994). Techniques vary, from the use of stemmers and morphological analysis to the use of thesauri (such as WordNet; see Fellbaum, 1998; Voorhees, 1998) to pad a query with synonyms, to the use of statistical analysis to identify more appropriate context-sensitive associations and near-synonyms (e.g. Xu and Croft, 1996). While some techniques may suggest conventional metaphors that have become lexicalized in a language, they are unlikely to identify relatively novel expressions. Crucially, expansion improves recall at the expense of overall precision, making automatic techniques even more dangerous when the goal is to retrieve results that are creative *and* relevant. Creative IR must balance a need for fine user control with the statistical breadth and convenience of automatic expansion.

Fortunately, statistical corpus analysis is an obvious area of overlap for IR and FLP. Distributional analyses of large corpora have been shown to produce nuanced models of lexical similarity (e.g. Weeds and Weir, 2005) as well as context-sensitive thesauri for a given domain (Lin, 1998). Hearst (1992) shows how a pattern like "*Xs and other Ys*" can be used to construct more fluid, context-specific taxonomies than those provided by WordNet (e.g. "*athletes and other celebrities*" suggests a context in which athletes are viewed as stars). Mason (2004) shows how statistical analysis can automatically detect and extract conventional metaphors from corpora, though creative metaphors still remain a tantalizing challenge. Hanks (2005) shows how the "*Xs like A, B and C*" construction allows us to derive flexible ad-hoc categories from corpora, while Hanks (2006) argues for a gradable conception of metaphoricity based on word-sense distributions in corpora.

Veale and Hao (2007) exploit the simile frame "*as X as Y*" to harvest a great many common similes and their underlying stereotypes from the web (e.g. "as *hot* as an oven"), while Veale and Hao (2010) show that the pattern "*about as X as Y*" retrieves an equally large collection of creative (if mostly ironic) comparisons. These authors demonstrate that a large vocabulary of stereotypical ideas (over 4000 nouns) and their salient properties (over 2000 adjectives) can be harvested from the web.

We now build on these results to develop a set of new semantic operators, that use corpus-derived knowledge to support finely controlled non-literal matching and automatic query expansion.

3 Creative Text Retrieval

In language, creativity is always a matter of construal. While conventional IR queries articulate a need for information, creative IR queries articulate a need for expressions to convey the same meaning in a fresh or unusual way. A query and a matching phrase can be figuratively construed to have the same meaning if there is a non-literal mapping between the elements of the query and the elements of the phrase. In creative IR, this non-literal mapping is facilitated by the query's explicit use of *semantic wildcards* (e.g. see Mihalcea, 2002).

The wildcard * is a boon for power-users of the Google search engine, precisely because it allows users to focus on the retrieval of matching phrases rather than relevant documents. For instance, * can be used to find alternate ways of instantiating a culturally-established linguistic pattern, or "snowclone": thus, the Google queries "*In * no one can hear you scream*" (from *Alien*), "*Reader, I * him*" (from *Jane Eyre*) and "*This is your brain on **" (from a famous TV advert) find new ways in which old patterns have been instantiated for humorous effect on the Web. On a larger scale, Veale and Hao (2007) used the * wildcard to harvest web similes, but reported that harvesting cultural data with wildcards is not a straightforward process. Google and other engines are designed to maximize document relevance and to rank results accordingly. They are not designed to maximize the diversity of results, or to find the largest set of wildcard bindings. Nor are they designed to find the most commonplace bindings for wildcards.

Following Guilford's (1950) pioneering work, diversity is widely considered a key component in the psychology of creativity. By focusing on the phrase level rather than the document level, and by returning phrase sets rather than document sets, creative IR maximizes diversity by finding as many bindings for its wildcards as a text collection will support. But we need more flexible and precise wildcards than *. We now consider three varieties of semantic wildcards that build on insights from corpus-linguistic approaches to FLP.

3.1 The *Neighborhood* Wildcard ?X

Semantic query expansion replaces a query term **X** with a set $\{X, X_1, X_2, \ldots, X_n\}$ where each X_i is related to **X** by a prescribed lexico-semantic relationship, such as synonymy, hyponymy or meronymy. A generic, lightweight resource like WordNet can provide these relations, or a richer ontology can be used if one is available (e.g. see Navigli and Velardi, 2003). Intuitively, each query term suggests other terms from its semantic neighborhood, yet there are practical limits to this intuition. X_i may not be an obvious or natural substitute for X. A neighborhood can be drawn too small, impacting recall, or too large, impacting precision.

Corpus analysis suggests an approach that is both semantic *and* pragmatic. As noted in Hanks (2005), languages provide constructions for building ad-hoc sets of items that can be considered comparable in a given context. For instance, a coordination of bare plurals suggests that two ideas are related at a generic level, as in "*priests and imams*" or "*mosques and synagogues*". More generally, consider the pattern "*X and Y*", where X and Y are proper-names (e.g., "*Zeus and Hera*"), or X and Y are inflected nouns or verbs with the same inflection (e.g., the plurals "*cats and dogs*" or the verb forms "*kicking and screaming*"). Millions of matches for this pattern can be found in the Google 3-grams (Brants and Franz, 2006), allowing us to build a map of comparable terms by linking the root-forms of X and Y with a similarity score obtained via a WordNet-based measure (e.g. see Budanitsky and Hirst (2006) for a good selection).

The pragmatic neighborhood of a term X can be defined as $\{X, X_1, X_2, \ldots, X_n\}$, so that for each X_i, the Google 3-grams contain "*X+inf and X_i+inf*" or "*X+inf and X_i+inf*". The boundaries of neighborhoods are thus set by usage patterns: if **?X** denotes the neighborhood of X, then **?artist**

matches not just *artist*, *composer* and *poet*, but *studio*, *portfolio* and *gallery,* and many other terms that are semantically dissimilar but pragmatically linked to *artist*. Since each $X_i \in$ **?X** is ranked by similarity to **X**, query matches can also be ranked by similarity.

When X is an adjective, then **?X** matches any element of {X, X_i, X_2, …, X_n}, where each X_i pragmatically reinforces X, and X pragmatically reinforces each X_i. To ensure X and X_i really are mutually reinforcing adjectives, we use the double-ground simile pattern "*as* X *and* X_i *as*" to harvest {X_1, …, X_n} for each X. Moreover, to maximize recall, we use the Google API (rather than Google ngrams) to harvest suitable bindings for X and X_i from the web. For example, **@witty** = {*charming, clever, intelligent, entertaining, ..., edgy, fun*}.

3.2 The *Cultural Stereotype* Wildcard @X

Dickens claims in *A Christmas Carol* that "the wisdom of a people is in the simile". Similes exploit familiar stereotypes to describe a less familiar concept, so one can learn a great deal about a culture and its language from the similes that have the most currency (Taylor, 1954). The wildcard **@X** builds on the results of Veale and Hao (2007) to allow creative IR queries to retrieve matches on the basis of cultural expectations. This foundation provides a large set of adjectival features (over 2000) for a larger set of nouns (over 4000) denoting stereotypes for which these features are salient.

If N is a noun, then **@N** matches any element of the set {A_1, A_2, …, A_n}, where each A_i is an adjective denoting a stereotypical property of N. For example, **@diamond** matches any element of {*transparent, immutable, beautiful, tough, expensive, valuable, shiny, bright, lasting, desirable, strong, ..., hard*} . If A is an adjective, then **@A** matches any element of the set {N_1, N_2, …, N_n}, where each N_i is a noun denoting a stereotype for which A is a culturally established property. For example, **@tall** matches any element of {*giraffe, skyscraper, tree, redwood, tower, sunflower, lighthouse, beanstalk, rocket, ..., supermodel*}.

Stereotypes crystallize in a language as clichés, so one can argue that stereotypes and clichés are little or no use to a creative IR system. Yet, as demonstrated in Fishlov (1992), creative language is replete with stereotypes, not in their clichéd guises, but in novel and often incongruous combinations. The creative value of a stereotype lies in how it is used, as we'll show later in section 4.

3.3 The *Ad-Hoc Category* Wildcard ^X

Barsalou (1983) introduced the notion of an *ad-hoc category*, a cross-cutting collection of often disparate elements that cohere in the context of a specific task or goal. The ad-hoc nature of these categories is reflected in the difficulty we have in naming them concisely: the cumbersome "things to take on a camping trip" is Barsalou's most cited example. But ad-hoc categories do not replace natural kinds; rather, they supplement an existing system of more-or-less rigid categories, such as the categories found in WordNet.

The semantic wildcard **^C** matches C and any element of {C_1, C_2, …, C_n}, where each C_i is a member of the category named by C. **^C** can denote a fixed category in a resource like WordNet or even Wikipedia; thus, **^fruit** matches any member of {*apple, orange, pear, ..., lemon*} and **^animal** any member of *{dog, cat, mouse, ..., deer, fox}*.

Ad-hoc categories arise in creative IR when the results of a query – or more specifically, the bindings for a query wildcard – are funneled into a new user-defined category. For instance, the query *"^**fruit** juice*" matches any phrase in a text collection that denotes a named fruit juice, from "*lemon juice*" to "*pawpaw juice*". A user can now funnel the bindings for **^fruit** in this query into an ad-hoc category **juicefruit**, to gather together those fruits that are used for their juice. Elements of *^juicefruit* are ranked by the corpus frequencies discovered by the original query; low-frequency **juicefruit** members in the Google ngrams include *coffee*, *raisin*, *almond*, *carob* and *soybean*. Ad-hoc categories allow users of IR to remake a category system in their own image, and create a new vocabulary of categories to serve their own goals and interests, as when "***^food** pizza*" is used to suggest disparate members for the ad-hoc category **pizzatopping**.

The more subtle a query, the more disparate the elements it can funnel into an ad-hoc category. We now consider how basic semantic wildcards can be combined to generate even more diverse results.

3.4 Compound Operators

Each wildcard maps a query term onto a set of ex-

pansion terms. The compositional semantics of a wildcard combination can thus be understood in set-theoretic terms. The most obvious and useful combinations of **?**, **@** and ^ are described below:

?? *Neighbor-of-a-neighbor*: if **?X** matches any element of {X, X_1, X_2, …, X_n} then **??X** matches any of **?X** ∪ **?X_1** ∪ … ∪ **?X_n**, where the ranking of X_{ij} in **??X** is a function of the ranking of X_i in **?X** and the ranking of X_{ij} in **?X_i**. Thus, **??artist** matches far more terms than **?artist**, yielding more diversity, more noise, and more creative potential.

@@ *Stereotype-of-a-stereotype*: if **@X** matches any element of {X_1, X_2, …, X_n} then **@@X** matches any of **@X_1** ∪ **@X_2** ∪ … ∪ **@X_n**. For instance, **@@diamond** matches any stereotype that shares a salient property with *diamond*, and **@@sharp** matches any salient property of any noun for which *sharp* is a stereotypical property.

?@ *Neighborhood-of-a-stereotype*: if **@X** matches any element of {X_1, X_2, …, X_n} then **?@X** matches any of **?X_1** ∪ **?X_2** ∪ … ∪ **?X_n**. Thus, **?@cunning** matches any term in the pragmatic neighborhood of a stereotype for *cunning*, while **?@knife** matches any property that mutually reinforces any stereotypical property of *knife*

@? *Stereotypes-in-a-neighborhood*: if **?X** matches any of {X, X_1, X_2, …, X_n} then **@?X** matches any of **@X** ∪ **@X_1** ∪ … ∪ **@X_n**. Thus, **@?corpse** matches any salient property of any stereotype in the neighborhood of *corpse*, while **@?fast** matches any stereotype noun with a salient property that is similar to, and reinforced by, *fast*.

?^ *Neighborhood-of-a-category*: if **^C** matches any of {C, C_1, C_2, …, C_n} then **?^C** matches any of **?C** ∪ **?C_1** ∪ … ∪ **?C_n**.

^? *Categories-in-a-neighborhood*: if **?X** matches any of {X, X_1, X_2, …, X_n} then **^?X** matches any of **^X** ∪ **^X_1** ∪ … ∪ **^X_n**.

@^ *Stereotypes-in-a-category*: if **^C** matches any of {C, C_1, C_2, …, C_n} then **@^C** matches any of **@C** ∪ **@C_1** ∪ … ∪ **@C_n**.

^@ *Members-of-a-stereotype-category*: if **@X** matches any element of {X_1, X_2, …, X_n} then **^@X** matches any of **^X_1** ∪ **^X_2** ∪ … ∪ **^X_n**. So **^@strong** matches any member of a category (such as *warrior*) that is stereotypically *strong*.

4 Applications of Creative Retrieval

The Google ngrams comprise a vast array of extracts from English web texts, of 1 to 5 words in length (Brants and Franz, 2006). Many extracts are well-formed phrases that give lexical form to many different ideas. But an even greater number of ngrams are not linguistically well-formed. The Google ngrams can be seen as a *lexicalized idea space*, embedded within a larger sea of noise. Creative IR can be used to explore this idea space.

Each creative query is a jumping off point in a space of lexicalized ideas that is implied by a large corpus, with each successive match leading the user deeper into the space. By turning matches into queries, a user can perform a creative exploration of the space of phrases and ideas (see Boden, 1994) while purposefully sidestepping the noise of the Google ngrams. Consider the pleonastic query "*Catholic* ***?pope***". Retrieved phrases include, in descending order of lexical similarity, "*Catholic president*", "*Catholic politician*", "*Catholic king*", "*Catholic emperor*" and "*Catholic patriarch*". Suppose a user selects "*Catholic king*": the new query "*Catholic* ***?king***" now retrieves "*Catholic queen*", "*Catholic court*", "*Catholic knight*", "*Catholic kingdom*" and "*Catholic throne*". The subsequent query "*Catholic* ***?kingdom***" in turn retrieves "*Catholic dynasty*" and "*Catholic army*", among others. In this way, creative IR allows a user to explore the text-supported ramifications of a metaphor like *Popes are Kings* (e.g., if popes are kings, they too might have queens, command armies, found dynasties, or sit on thrones).

Creative IR gives users the tools to conduct their own explorations of language. The more wildcards a query contains, the more degrees of freedom it offers to the explorer. Thus, the query "***?scientist*** *'s* ***?laboratory***" uncovers a plethora of analogies for the relationship between scientists and their labs: matches in the Google 3-grams include "*technician's workshop*", "*artist's studio*", "*chef's kitchen*" and "*gardener's greenhouse*".

4.1 Metaphors with *Aristotle*

For a term X, the wildcard **?X** suggests those other terms that writers have considered to be comparable to X, while **??X** extrapolates beyond the corpus evidence to suggest an even larger space of potential comparisons. A meaningful metaphor can be constructed for X by framing X with any stereotype to which it is pragmatically comparable, that is, any stereotype in **?X**. Collectively, these stereotypes can impart the properties **@?X** to X.

Suppose one wants to metaphorically ascribe the property P to X. The set **@P** contains those stereotypes for which P is culturally salient. Thus, close metaphors for X (what MacCormac (1985) dubs *epiphors*) in the context of P are suggested by **?X ∩ @P**. More distant metaphors (MacCormac dubs these *diaphors*) are suggested by **??X ∩ @P**. For instance, to describe a *scholar* as *wise*, one can use *poet*, *yogi*, *philosopher* or *rabbi* as comparisons. Yet even a simple metaphor will impart other features to a topic. If $\mathbf{\hat{}P_S}$ denotes the ad-hoc set of additional properties that may be inferred for X when a stereotype S is used to convey property P, then $\mathbf{\hat{}P_S}$ = **?P ∩ @@P**. The query "$\mathbf{\hat{}P_S}$ **X**" now finds corpus-attested elements of $\mathbf{\hat{}P_S}$ that can meaningfully be used to modify X.

These IR formulations are used by *Aristotle*, an online metaphor generator, to generate targeted metaphors that highlight a property P in a topic X. *Aristotle* uses the Google ngrams to supply values for **?X**, **??X**, **?P** and $\mathbf{\hat{}P_S}$. The system can be accessed at: *www.educatedinsolence.com/aristotle*

4.2 Expressing Attitude with *Idiom Savant*

Our retrieval goals in IR are often affective in nature: we want to find a way of speaking about a topic that expresses a particular sentiment and carries a certain tone. However, affective categories are amongst the most cross-cutting structures in language. Words for disparate ideas are grouped according to the sentiments in which they are generally held. We respect *judges* but dislike *critics*; we respect *heroes* but dislike *killers*; we respect *sharpshooters* but dislike *snipers*; and respect *rebels* but dislike *insurgents*. It seems therefore that the particulars of sentiment are best captured by a set of culture-specific ad-hoc categories.

We thus construct two ad-hoc categories, **^posword** and **^negword**, to hold the most obviously positive or negative words in Whissell's (1989) *Dictionary of Affect*. We then grow these categories to include additional reinforcing elements from their pragmatic neighborhoods, **?^posword** and **?^negword**. As these categories grow, so too do their neighborhoods, allowing a simple semi-automated bootstrapping process to significantly grow the categories over several iterations. We construct two phrasal equivalents of these categories, **^posphrase** and **^negphrase**, using the queries "***^posword - ^pastpart***" (e.g., matching "*high-minded*" and "*sharp-eyed*") and "***^negword - ^pastpart***" (e.g., matching "*flat-footed*" and "*dead-eyed*") to mine affective phrases from the Google 3-grams. The resulting ad-hoc categories (of ~600 elements each) are manually edited to fix any obvious mis-categorizations.

Idiom Savant is a web application that uses **^posphrase** and **^negphrase** to suggest flattering and insulting epithets for a given topic. The query "**^posphrase ?X**" retrieves phrases for a topic X that put a positive spin on a related topic to which X is sometimes compared, while "**^negphrase ?X**" conversely imparts a negative spin. Thus, for *politician*, the Google 4-grams provide the flattering epithets "*much-needed leader*", "*awe-inspiring leader*", "*hands-on boss*" and "*far-sighted statesman*", as well as insults like "*power-mad leader*", "*back-stabbing boss*", "*ice-cold technocrat*" and "*self-promoting hack*". Riskier diaphors can be retrieved via "**^posphrase ??X**" and "**^negphrase ??X**". *Idiom Savant* is accessible online at: *www.educatedinsolence.com/idiom-savant/*

4.3 Poetic Similes with *The Jigsaw Bard*

The well-formed phrases of a large corpus can be viewed as the linguistic equivalent of *objets trouvés* in art: readymade or "found" objects that might take on fresh meanings in a creative context. The phrase "*robot fish*", for instance, denotes a more-or-less literal object in the context of autonomous robotic submersibles, but can also be used to convey a figurative meaning as part of a creative comparison (e.g., "*he was as cold as a robot fish*").

Fishlov (1992) argues that poetic comparisons are most resonant when they combine mutually-reinforcing (if distant) ideas, to create memorable images and evoke nuanced feelings. Building on Fishlov's argument, creative IR can be used to turn

the readymade phrases of the Google ngrams into vehicles for creative comparison. For a topic X and a property P, simple similes of the form "***X is as P as S***" are easily generated, where **S** ∈ **@P** ∩ **??X**.

Fishlov would dub these *non-poetic similes* (NPS). However, the query "***?P @P***" will retrieve corpus-attested elaborations of stereotypes in **@P** to suggest similes of the form "***X is as P as*** $\mathbf{P_1}$ ***S***", where $\mathbf{P_1}$ ∈ **?P**. These similes exhibit elements of what Fishlov dubs *poetic similes* (PS). Why say "*as cold as a fish*" when you can say "*as cold as a wet fish*", "*a dead haddock*", "*a wet January*", "*a frozen corpse*", or "*a heartless robot*"? Complex queries can retrieve more creative combinations, so "***@P @P***" (e.g. "*robot fish*" or "*snow storm*" for *cold*), "**?P @P @P**" (e.g. "*creamy chocolate mousse*" for *rich*) and "***@P - ^pastpart @P***" (e.g. "*snow-covered graveyard*" and "*bullet-riddled corpse*" for *cold*) each retrieve ngrams that blend two different but overlapping stereotypes.

Blended properties also make for nuanced similes of the form "*as **P** and **?P** as **S***", where **S** ∈ **@P** ∩ **@?P**. While one can be "*as rich as a fat king*", something can be "*as rich and enticing as a chocolate truffle*", "*a chocolate brownie*", "*a chocolate fruitcake*", and even "*a chocolate king*".

The *Jigsaw Bard* is a web application that harnesses the readymades of the Google ngrams to formulate novel similes from existing phrases. By mapping blended properties to ngram phrases that combine multiple stereotypes, the *Bard* expands its generative scope considerably, allowing this application to generate hundreds of thousands of evocative comparisons. The *Bard* can be accessed online at: *www.educatedinsolence.com/jigsaw/*

5 Empirical Evaluation

Though ^ is the most overtly categorical of our wildcards, all three wildcards – **?**, **@** and ^ – are categorical in nature. Each has a semantic or pragmatic membership function that maps a term onto an expansion set of related members. The membership functions for specific uses of ^ are created in an ad-hoc fashion by the users that exploit it; in contrast, the membership functions for uses of **@** and **?** are derived automatically, via pattern-matching and corpus analysis. Nonetheless, ad-hoc categories in creative IR are often populated with the bindings produced by uses of **@** and **?** and combinations thereof. In a sense, **?X** and **@X** and their variations are themselves ad-hoc categories. But how well do they serve as categories? Are they large, but noisy? Or too small, with limited coverage? We can evaluate the effectiveness of **?** and **@**, and indirectly that of ^ too, by comparing the use of **?** and **@** as category builders to a hand-crafted gold standard like WordNet.

Other researchers have likewise used WordNet as a gold standard for categorization experiments, and we replicate here the experimental set-up of Almuhareb and Poesio (2004, 2005), which is designed to measure the effectiveness of web-acquired conceptual descriptions. Almuhareb and Poesio choose 214 English nouns from 13 of WordNet's upper-level semantic categories, and proceed to harvest property values for these concepts from the web using the Hearst-like pattern "*a|an|the * C is|was*". This pattern yields a combined total of 51,045 values for all 214 nouns; these values are primarily adjectives, such as *hot* and *black* for *coffee*, but noun-modifiers of C are also allowed, such as *fruit* for *cake*. They also harvest 8934 attribute nouns, such as *temperature* and *color*, using the query "*the * of the C is|was*". These values and attributes are then used as the basis of a clustering algorithm to partition the 214 nouns back into their original 13 categories. Comparing these clusters with the original WordNet-based groupings, Almuhareb and Poesio report a cluster accuracy of **71.96%** using just values like *hot* and *black* (51,045 values), an accuracy of **64.02%** using just attributes like *temperature* and *color* (8,934 attributes), and an accuracy of **85.5%** using both together (a combined 59,979 features).

How concisely and accurately does **@X** describe a noun X for purposes of categorization? Let **^AP** denote the set of 214 WordNet nouns used by Almuhareb and Poesio. Then **@^AP** denotes a set of 2,209 adjectival properties; this should be contrasted with the space of 51,045 adjectival values used by Almuhareb and Poesio. Using the same clustering algorithm over this feature set, **@X** achieves a clustering accuracy (as measured via cluster purity) of **70.2%**, compared to **71.96%** for Almuhareb and Poesio. However, when **@X** is used to harvest a further set of attribute nouns for X, via web queries of the form "***the P * of X***" (where P ∈ **@X**), then **@X** augmented with this additional set of attributes (like *hands* for surgeon)

produces a larger space of 7,183 features. This in turn yields a cluster accuracy of **90.2%** which contrasts with Almuhareb and Poesio's **85.5%** for 59,979 features. In either case, **@X** produces comparable clustering quality to Almuhareb and Poesio, with just a small fraction of the features.

So how concisely and accurately does **?X** describe a noun X for purposes of categorization? While **@X** denotes a set of salient adjectives, **?X** denotes a set of comparable nouns. So this time, **?^AP** denotes a set of 8,300 nouns in total, to act as a feature space for the 214 nouns of Almuhareb and Poesio. Remember, the contents of each **?X**, and of **?^AP** overall, are determined entirely by the contents of the Google 3-grams; the elements of **?X** are not ranked in any way, and all are treated as equals. When the 8,300 features in **?^AP** are clustered into 13 categories, the resulting clusters have a purity of **93.4%** relative to WordNet. The pragmatic neighborhood of X, **?X**, appears to be an accurate and concise proxy for the meaning of X.

What about adjectives? Almuhareb and Poesio's set of 214 words does not contain adjectives, and besides, WordNet does not impose a category structure on its adjectives. In any case, the role of adjectives in the applications of section 4 is largely an affective one: if X is a noun, then one must have confidence that the adjectives in **@X** are consonant with our understanding of X, and if P is a property, that the adjectives in **?P** evoke much the same mood and sentiment as P. Our evaluation of **@X** and **?P** should thus be an affective one.

So how well do the properties in **@X** capture our sentiments about a noun X? Well enough to estimate the pleasantness of X from the adjectives in **@X**, perhaps? Whissell's (1989) dictionary of affect provides pleasantness ratings for a sizeable number of adjectives and nouns (over 8,000 words in total), allowing us to estimate the pleasantness of X as a weighted average of the pleasantness of each X_i in **@X** (the weights here are web frequencies for the similes that underpin @ in section 3.2). We thus estimate the affect of all stereotype nouns for which Whissell also records a score. A two-tailed Pearson test ($p < 0.05$) shows a positive correlation of **0.5** between these estimates and the pleasantness scores assigned by Whissell. In contrast, estimates based on the pleasantness of adjectives found in corresponding WordNet glosses show a positive correlation of just **0.278**.

How well do the elements of **?P** capture our sentiments toward an adjective P? After all, we hypothesize that the adjectives in **?P** are highly suggestive of P, and vice versa. *Aristotle* and the *Jigsaw Bard* each rely on **?P** to suggest adjectives that evoke an unstated property in a metaphor or simile, or to suggest coherent blends of properties. When we estimate the pleasantness of each adjective P in Whissell's dictionary via the weighted mean of the pleasantness of adjectives in **?P** (again using web frequencies as weights), a two-tailed Pearson test ($p < 0.05$) shows a correlation of **0.7** between estimates and actual scores. It seems **?P** does a rather good job of capturing the feel of P.

6 Concluding Remarks

Creative information retrieval is not a single application, but a paradigm that allows us to conceive of many different kinds of application for creatively manipulating text. It is also a tool-kit for implementing such an application, as shown here in the cases of *Aristotle*, *Idiom Savant* and *Jigsaw Bard*.

The wildcards **@**, **?** and ^ allow users to formulate their own task-specific ontologies of ad-hoc categories. In a fully automated application, they provide developers with a simple but powerful vocabulary for describing the range and relationships of the words, phrases and ideas to be manipulated.

The **@**, **?** and ^ wildcards are just a start. We expect other aspects of figurative language to be incorporated into the framework whenever they prove robust enough for use in an IR context. In this respect, we aim to position Creative IR as an open, modular platform in which diverse results in FLP, from diverse researchers, can be meaningfully integrated. One can imagine wildcards for matching potential puns, portmanteau words and other novel forms, as well as wildcards for figurative processes like metonymy, synecdoche, hyperbolae and even irony. Ultimately, it is hoped that creative IR can serve as a textual bridge between high-level creativity and the low-level creative potentials that are implicit in a large corpus.

Acknowledgments

This work was funded in part by Science Foundation Ireland (SFI), via the Centre for Next Generation Localization. (CNGL).

References

Almuhareb, A. and Poesio, M. (2004). Attribute-Based and Value-Based Clustering: An Evaluation. In *Proc. of EMNLP 2004*. Barcelona.

Almuhareb, A. and Poesio, M. (2005). Concept Learning and Categorization from the Web. In *Proc. of the 27th Annual meeting of the Cognitive Science Society*.

Barnden, J. A. (2006). Artificial Intelligence, figurative language and cognitive linguistics. In: G. Kristiansen, M. Achard, R. Dirven, and F. J. Ruiz de Mendoza Ibanez (Eds.), *Cognitive Linguistics: Current Application and Future Perspectives*, 431-459. Berlin: Mouton de Gruyter.

Barsalou, L. W. (1983). Ad hoc categories. *Memory and Cognition*, 11:211–227.

Boden, M. (1994). Creativity: A Framework for Research, *Behavioural & Brain Sciences* 17(3):558-568.

Brants, T. and Franz, A. (2006). *Web 1T 5-gram Ver. 1*. Linguistic Data Consortium.

Budanitsky, A. and Hirst, G. (2006). Evaluating WordNet-based Measures of Lexical Semantic Relatedness. *Computational Linguistics,* 32(1):13-47.

Falkenhainer, B., Forbus, K. and Gentner, D. (1989). Structure-Mapping Engine: Algorithm and Examples. *Artificial Intelligence*, 41:1-63.

Fass, D. (1991). Met*: a method for discriminating metonymy and metaphor by computer. *Computational Linguistics*, 17(1):49-90.

Fass, D. (1997). Processing Metonymy and Metaphor. *Contemporary Studies in Cognitive Science & Technology*. New York: Ablex.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge.

Fishlov, D. (1992). Poetic and Non-Poetic Simile: Structure, Semantics, Rhetoric. *Poetics Today*, 14(1), 1-23.

Gentner, D. (1983), Structure-mapping: A Theoretical Framework. *Cognitive Science* 7:155–170.

Guilford, J.P. (1950) Creativity, *American Psychologist* 5(9):444–454.

Hanks, P. (2005). Similes and Sets: The English Preposition 'like'. In: Blatná, R. and Petkevic, V. (Eds.), *Languages and Linguistics: Festschrift for Fr. Cermak*. Charles University, Prague.

Hanks, P. (2006). Metaphoricity is gradable. In: Anatol Stefanowitsch and Stefan Th. Gries (Eds.), *Corpus-Based Approaches to Metaphor and Metonymy*,. 17-35. Berlin: Mouton de Gruyter.

Hearst, M. (1992). Automatic acquisition of hyponyms from large text corpora. *In Proc. of the 14th Int. Conf. on Computational Linguistics*, pp 539–545.

Indurkhya, B. (1992). *Metaphor and Cognition: Studies in Cognitive Systems*. Kluwer Academic Publishers, Dordrecht: The Netherlands.

Lin, D. (1998). Automatic retrieval and clustering of similar words. In Proc. of the 17th international conference on Computational linguistics, 768-774.

MacCormac, E. R. (1985). *A Cognitive Theory of Metaphor*. MIT Press.

Martin, J. H. (1990). A Computational Model of Metaphor Interpretation. New York: Academic Press.

Mason, Z. J. (2004). CorMet: A Computational, Corpus-Based Conventional Metaphor Extraction System, *Computational Linguistics*, 30(1):23-44.

Mihalcea, R. (2002). The Semantic Wildcard. In *Proc. of the LREC Workshop on Creating and Using Semantics for Information Retrieval and Filtering*. Canary Islands, Spain, May 2002.

Navigli, R. and Velardi, P. (2003). An Analysis of Ontology-based Query Expansion Strategies. In Proc. of the workshop on Adaptive Text Extraction and Mining (ATEM 2003), at ECML 2003, the 14th European Conf. on Machine Learning, 42–49

Salton, G. (1968). *Automatic Information Organization and Retrieval*. New York: McGraw-Hill.

Taylor, A. (1954). Proverbial Comparisons and Similes from California. *Folklore Studies* 3. Berkeley: University of California Press.

Van Rijsbergen, C. J. (1979). *Information Retrieval*. Oxford: Butterworth-Heinemann.

Veale, T. (2004). The Challenge of Creative Information Retrieval. *Computational Linguistics and Intelligent Text Processing: Lecture Notes in Computer Science*, Volume 2945/2004, 457-467.

Veale, T. (2006). Re-Representation and Creative Analogy: A Lexico-Semantic Perspective. New Generation Computing 24, pp 223-240.

Veale, T. and Hao, Y. (2007). Making Lexical Ontologies Functional and Context-Sensitive. *In Proc. of the 46th Annual Meeting of the Assoc. of Computational Linguistics.*

Veale, T. and Hao, Y. (2010). Detecting Ironic Intent in Creative Comparisons. *In Proc. of ECAI'2010, the 19th European Conference on Artificial Intelligence*.

Veale, T. and Butnariu, C. (2010). Harvesting and Understanding On-line Neologisms. In: Onysko, A. and Michel, S. (Eds.), *Cognitive Perspectives on Word Formation*. 393-416. Mouton De Gruyter.

Vernimb, C. (1977). Automatic Query Adjustment in Document Retrieval. *Information Processing & Management*. 13(6):339-353.

Voorhees, E. M. (1994). Query Expansion Using Lexical-Semantic Relations. In *the proc. of SIGIR 94, the 17th International Conference on Research and Development in Information Retrieval*. Berlin: Springer-Verlag, 61-69.

Voorhees, E. M. (1998). Using WordNet for text retrieval. *WordNet, An Electronic Lexical Database*, 285–303. The MIT Press.

Way, E. C. (1991). Knowledge Representation and Metaphor. *Studies in Cognitive systems*. Holland: Kluwer.

Weeds, J. and Weir, D. (2005). Co-occurrence retrieval: A flexible framework for lexical distributional similarity. *Computational Linguistics*, 31(4):433–475.

Whissell, C. (1989). The dictionary of affect in language. R. Plutchnik & H. Kellerman (Eds.) *Emotion: Theory and research*. NY: Harcourt Brace, 113-131.

Wilks, Y. (1978). Making Preferences More Active, *Artificial Intelligence* 11.

Xu, J. and Croft, B. W. (1996). Query expansion using local and global document analysis. In *Proc. of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*.

Local Histograms of Character N-grams for Authorship Attribution

Hugo Jair Escalante
Graduate Program in Systems Eng.
Universidad Autónoma de Nuevo León,
San Nicolás de los Garza, NL, 66450, México
hugo.jair@gmail.com

Thamar Solorio
Dept. of Computer and Information Sciences
University of Alabama at Birmingham,
Birmingham, AL, 35294, USA
solorio@cis.uab.edu

Manuel Montes-y-Gómez
Computer Science Department, INAOE,
Tonantzintla, Puebla, 72840, México
Department of Computer and Information Sciences,
University of Alabama at Birmingham,
Birmingham, AL, 35294, USA
mmontesg@cis.uab.edu

Abstract

This paper proposes the use of local histograms (LH) over character n-grams for authorship attribution (AA). LHs are enriched histogram representations that preserve sequential information in documents; they have been successfully used for text categorization and document visualization using *word* histograms. In this work we explore the suitability of LHs over n-grams at the *character-level* for AA. We show that LHs are particularly helpful for AA, because they provide useful information for uncovering, to some extent, the writing style of authors. We report experimental results in AA data sets that confirm that LHs over character n-grams are more helpful for AA than the usual global histograms, yielding results far superior to state of the art approaches. We found that LHs are even more advantageous in challenging conditions, such as having imbalanced and small training sets. Our results motivate further research on the use of LHs for modeling the writing style of authors for related tasks, such as authorship verification and plagiarism detection.

1 Introduction

Authorship attribution (AA) is the task of deciding whom, from a set of candidates, is the author of a given document (Houvardas and Stamatatos, 2006; Luyckx and Daelemans, 2010; Stamatatos, 2009b). There is a broad field of application for AA methods, including spam filtering (de Vel et al., 2001), fraud detection, computer forensics (Lambers and Veenman, 2009), cyber bullying (Pillay and Solorio, 2010) and plagiarism detection (Stamatatos, 2009a). Therefore, the development of automated AA techniques has received much attention recently (Stamatatos, 2009b). The AA problem can be naturally posed as one of single-label multiclass classification, with as many classes as candidate authors. However, unlike usual text categorization tasks, where the core problem is modeling the thematic content of documents (Sebastiani, 2002), the goal in AA is modeling authors' writing style (Stamatatos, 2009b). Hence, document representations that reveal information about writing style are required to achieve good accuracy in AA.

Word and character based representations have been used in AA with some success so far (Houvardas and Stamatatos, 2006; Luyckx and Daelemans, 2010; Plakias and Stamatatos, 2008b). Such representations can capture style information through word or character usage, but they lack sequential information, which can reveal further stylistic information. In this paper, we study the use of richer document representations for the AA task. In particular, we consider local histograms over n-grams at the character-level obtained via the locally-weighted bag of words (LOWBOW) framework (Lebanon et al., 2007).

Under LOWBOW, a document is represented by a set of local histograms, computed across the whole document but smoothed by kernels centered on different document locations. In this way, document

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 288–298,
Portland, Oregon, June 19-24, 2011.

representations preserve both word/character usage and sequential information (i.e., information about the positions in which words or characters occur), which can be more helpful for modeling the writing style of authors. We report experimental results in an AA data set used in previous studies under several conditions (Houvardas and Stamatatos, 2006; Plakias and Stamatatos, 2008b; Plakias and Stamatatos, 2008a). Results confirm that local histograms of character n-grams are more helpful for AA than the usual global histograms of words or character n-grams (Luyckx and Daelemans, 2010); our results are superior to those reported in related works. We also show that local histograms over character n-grams are more helpful than local histograms over words, as originally proposed by (Lebanon et al., 2007). Further, we performed experiments with imbalanced and small training sets (i.e., under a realistic AA setting) using the aforementioned representations. We found that the LOWBOW-based representation resulted even more advantageous in these challenging conditions. The contributions of this work are as follows:

- We show that the LOWBOW framework can be helpful for AA, giving evidence that sequential information encoded in local histograms is useful for modeling the writing style of authors.
- We propose the use of local histograms over character-level n-grams for AA. We show that character-level representations, which have proved to be very effective for AA (Luyckx and Daelemans, 2010), can be further improved by adopting a local histogram formulation. Also, we empirically show that local histograms at the character-level are more helpful than local histograms at the word-level for AA.
- We study several kernels for a support vector machine AA classifier under the local histograms formulation. Our study confirms that the diffusion kernel (Lafferty and Lebanon, 2005) is the most effective among those we tried, although competitive performance can be obtained with simpler kernels.
- We report experimental results that are superior to state of the art approaches (Plakias and Stamatatos, 2008b; Plakias and Stamatatos, 2008a), with improvements ranging from $2\% - 6\%$ in balanced data sets and from $14\% - 30\%$ in imbalanced data sets.

2 Related Work

AA can be faced as a multiclass classification task with as many classes as candidate authors. Standard classification methods have been applied to this problem, including support vector machine (SVM) classifiers (Houvardas and Stamatatos, 2006) and variants thereon (Plakias and Stamatatos, 2008b; Plakias and Stamatatos, 2008a), neural networks (Tearle et al., 2008), Bayesian classifiers (Coyotl-Morales et al., 2006), decision tree methods (Koppel et al., 2009) and similarity based techniques (Keselj et al., 2003; Lambers and Veenman, 2009; Stamatatos, 2009b; Koppel et al., 2009). In this work, we chose an SVM classifier as it has reported acceptable performance in AA and because it will allow us to directly compare results with previous work that has used this same classifier.

A broad diversity of features has been used to represent documents in AA (Stamatatos, 2009b). However, as in text categorization (Sebastiani, 2002), word-based and character-based features are among the most widely used features (Stamatatos, 2009b; Luyckx and Daelemans, 2010). With respect to word-based features, word histograms (i.e., the bag-of-words paradigm) are the most frequently used representations in AA (Zhao and Zobel, 2005; Argamon and Levitan, 2005; Stamatatos, 2009b). Some researchers have gone a step further and have attempted to capture sequential information by using n-grams at the word-level (Peng et al., 2004) or by discovering maximal frequent word sequences (Coyotl-Morales et al., 2006). Unfortunately, because of computational limitations, the latter methods cannot discover *enough* sequential information from documents (e.g., word n-grams are often restricted to $n \in \{1, 2, 3\}$, while full sequential information would be obtained with $n \in \{1 \dots D\}$ where D is the maximum number of words in a document).

With respect to character-based features, n-grams at the character level have been widely used in AA as well (Plakias and Stamatatos, 2008b; Peng et al., 2003; Luyckx and Daelemans, 2010). Peng et al. (2003) propose the use of language models at the n-gram character-level for AA, whereas Keselj et al. (2003) build author profiles based on a selection of frequent n-grams for each author. Stamatatos and co-workers have studied the impact of feature selection, with character n-grams, in AA (Houvardas and Stamatatos, 2006; Stamatatos, 2006a), ensemble learning with character n-grams (Stamatatos, 2006b) and novel classification techniques based

on characters at the n-gram level (Plakias and Stamatatos, 2008a).

Acceptable performance in AA has been reported with character n-gram representations. However, as with word-based features, character n-grams are unable to incorporate sequential information from documents in their original form (in terms of the positions in which the terms appear across a document). We believe that sequential clues can be helpful for AA because different authors are expected to use different character n-grams or words in different parts of the document. Accordingly, in this work we adopt the popular character-based and word-based representations, but we enrich them in a way that they incorporate sequential information via the LOWBOW framework. Hence, the proposed features preserve sequential information besides capturing character and word usage information. Our hypothesis is that the combination of sequential and frequency information can be particularly helpful for AA.

The LOWBOW framework has been mainly used for document visualization (Lebanon et al., 2007; Mao et al., 2007), where researchers have used information derived from local histograms for displaying a 2D representation of document's content. More recently, Chasanis et al. (2009) used the LOWBOW framework for segmenting movies into chapters and scenes. LOWBOW representations have also been applied to discourse segmentation (AMIDA, 2007) and have been suggested for text summarization (Das and Martins, 2007). However, to the best of our knowledge the use of the LOWBOW framework for AA has not been studied elsewhere. Actually, the only two references using this framework for text categorization are (Lebanon et al., 2007; AMIDA, 2007). The latter can be due to the fact that local histograms provide little gain over usual global histograms for thematic classification tasks. In this paper we show that LOWBOW representations provide important improvements over global histograms for AA; in particular, local histograms at the character-level achieve the highest performance in our experiments.

3 Background

This section describes preliminary information on document representations and pattern classification with SVMs.

3.1 Bag of words representations

In the bag of words (BOW) representation, documents are represented by histograms over the vocabulary[1] that was used to generate a collection of documents; that is, a document i is represented as:

$$\mathbf{d}_i = [x_{i,1}, \ldots, x_{i,|V|}] \quad (1)$$

where V is the vocabulary and $|V|$ is the number of elements in V, $\mathbf{d}_{i,j} = x_{i,j}$ is a weight that denotes the contribution of term j to the representation of document i; usually $x_{i,j}$ is related to the occurrence (binary weighting) or the weighted frequency of occurrence (e.g., the *tf-idf* weighting scheme) of the term j in document i.

3.2 Locally-weighted bag-of-words representation

Instead of using the BOW framework directly, we adopted the LOWBOW framework for document representation (Lebanon et al., 2007). The underlying idea in LOWBOW is to compute several local histograms per document, where these histograms are smoothed by a kernel function, see Figure 1. The parameters of the kernel specify the position of the kernel in the document (i.e., where the local histogram is centered) and its scale (i.e., to what extent it is smoothed). In this way the sequential information in the document is preserved together with term usage statistics.

Let $W_i = \{w_{i,1}, \ldots, w_{i,N_i}\}$, denote the terms (in order of appearance) in document i where N_i is the number of terms that appear in document i and $w_{i,j} \in V$ is the term appearing at position j; let $\mathbf{v}_i = \{v_{i,1}, \ldots, v_{i,N_i}\}$ be the set of indexes in the vocabulary V of the terms appearing in W_i, such that $v_{i,j}$ is the index in V of the term $w_{i,j}$; let $\mathbf{t} = [t_1, \ldots, t_{N_i}]$ be a set of (equally spaced) scalars that determine intervals, with $0 \leq t_j \leq 1$ and $\sum_{j=1}^{N_i} t_j = 1$, such that each t_j can be associated to a position in W_i. Given a kernel smoothing function $K^s_{\mu,\sigma} : [0,1] \rightarrow \mathbb{R}$ with location parameter μ and scale parameter σ, where $\sum_{j=1}^{k} K^s_{\mu,\sigma}(t_j) = 1$ and

[1] In the following we will refer to arbitrary vocabularies, which can be formed with terms from either words or character n-grams.

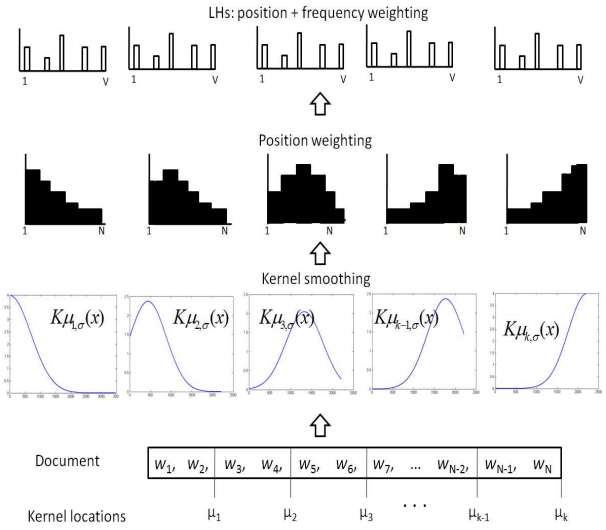


Figure 1: Diagram of the process for obtaining local histograms. Terms (w_i) appearing in different positions ($1, \ldots, N$) of the document are weighted according to the locations ($\mu_1, \ldots, \mu_k$) of the smoothing function $K_{\mu,\sigma}(x)$. Then, the term position weighting is combined with term frequency weighting for obtaining local histograms over the terms in the vocabulary ($1, \ldots, |V|$).

$\mu \in [0, 1]$. The LOWBOW framework computes a local histogram for each position $\mu_j \in \{\mu_1, \ldots, \mu_k\}$ as follows:

$$\mathbf{dl}^j_{i,\{v_{i,1},\ldots,v_{i,N_i}\}} = \mathbf{d}_{i,\{v_{i,1},\ldots,v_{i,N_i}\}} \times K^s_{\mu_j,\sigma}(\mathbf{t}) \quad (2)$$

where $\mathbf{dl}_{i,v_j:v_j \notin \mathbf{v}_i} = const$, a small constant value, and $\mathbf{d}_{i,j}$ is defined as above. Hence, a set $\mathbf{dl}_i^{\{1,\ldots,k\}}$ of k local histograms are computed for each document i. Each histogram $\mathbf{dl}_i^j$ carries information about the distribution of terms at a certain position μ_j of the document, where σ determines how the nearby terms to μ_j influence the local histogram j. Thus, sequential information of the document is considered throughout these local histograms. Note that when σ is small, most of the sequential information is preserved, as local histograms are calculated at very local scales; whereas when $\sigma \geq 1$, local histograms resemble the traditional BOW representation.

Under LOWBOW documents can be represented in two forms (Lebanon et al., 2007): as a single histogram $\mathbf{d}_i^L = const \times \sum_{j=1}^{k} \mathbf{dl}_i^j$ (hereafter LOWBOW histograms) or by the set of local histograms itself $\mathbf{dl}_i^{\{1,\ldots,k\}}$. We performed experiments with both forms of representation and considered words and n-grams at the character-level as terms (c.f. Section 5). Regarding the smoothing function, we considered the re-normalized Gaussian *pdf* restricted to [0, 1]:

$$K^s_{\mu,\sigma}(x) = \begin{cases} \frac{\mathcal{N}(x;\mu,\sigma)}{\phi(\frac{1-\mu}{\sigma}) - \phi(\frac{-\mu}{\sigma})} & \text{if } x \in [0,1] \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where $\phi(x)$ is the cumulative distribution function for a Gaussian with mean 0 and standard deviation 1, evaluated at x, see (Lebanon et al., 2007) for further details.

3.3 Support vector machines

Support vector machines (SVMs) are pattern classification methods that aim to find an optimal separating hyperplane between examples from two different classes (Shawe-Taylor and Cristianini, 2004). Let $\{\mathbf{x}_i, y_i\}_N$ be pairs of training patterns-outputs, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y \in \{-1, 1\}$, with d the dimensionality of the problem. SVMs aim at learning a mapping from training instances to outputs. This is done by considering a linear function of the form: $f(\mathbf{x}) = W\mathbf{x} + b$, where parameters W and b are learned from training data. The particular linear function considered by SVMs is as follows:

$$f(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) - b \quad (4)$$

that is, a linear function over (a subset of) training examples, where α_i is the weight associated with training example i (those for which $\alpha_i > 0$ are the so called support vectors) and y_i is the label associated with training example i, $K(\mathbf{x}_i, \mathbf{x}_j)$ is a kernel[2] function that aims at mapping the input vectors, $(\mathbf{x}_i, \mathbf{x}_j)$, into the so called feature space, and b is a bias term. Intuitively, $K(\mathbf{x}_i, \mathbf{x}_j)$ evaluates how similar instances $\mathbf{x}_i$ and $\mathbf{x}_j$ are, thus the particular choice of kernel is problem dependent. The parameters in expression (4), namely $\alpha_{\{1,\ldots,N\}}$ and b, are learned by using exact optimization techniques (Shawe-Taylor and Cristianini, 2004).

[2]One should not confuse the kernel smoothing function, $K^s_{\mu,\sigma}(x)$, defined in Equation (3) with the Mercer kernel in Equation (4), as the former acts as a smoothing function and the latter acts as a similarity function.

4 Authorship Attribution with LOWBOW Representations

For AA we represent the training documents of each author using the framework described in Section 3.2, thus each document of each candidate author is either a LOWBOW histogram or a bag of local histograms (BOLH). Recall that LOWBOW histograms are an un-weighted sum of local histograms and hence can be considered a summary of term usage and sequential information; whereas the BOLH can be seen as term occurrence frequencies across different locations of the document.

For both types of representations we consider an SVM classifier under the one-vs-all formulation for facing the AA problem. We consider SVM as base classifier because this method has proved to be very effective in a large number of applications, including AA (Houvardas and Stamatatos, 2006; Plakias and Stamatatos, 2008b; Plakias and Stamatatos, 2008a); further, since SVMs are kernel-based methods, they allow us to use local histograms for AA by considering kernels that work over sets of histograms.

We build a multiclass SVM classifier by considering the pairs of patterns-outputs associated to documents-authors. Where each pattern can be either a LOWBOW histogram or the set of local histograms associated with the corresponding document, and the output associated to each pattern is a categorical random variable (outputs) that associates the representation of each document to its corresponding author $y_{1,\ldots,N} \in \{1,\ldots,C\}$, with C the number of candidate authors. For building the multiclass classifier we adopted the one-vs-all formulation, where C binary classifiers are built and where each classifier f_i discriminates among examples from class i (positive examples) and the rest $j : j \in \{1,\ldots,C\}, j \neq i$; despite being one of the simplest formulations, this approach has shown to obtain comparable and even superior performance to that obtained by more complex formulations (Rifkin and Klautau, 2004).

For AA using LOWBOW histograms, we consider a linear kernel since it has been successfully applied to a wide variety of problems (Shawe-Taylor and Cristianini, 2004), including AA (Houvardas and Stamatatos, 2006; Plakias and Stamatatos, 2008b). However, *standard* kernels cannot work for input spaces where each instance is described by a set of vectors. Therefore, usual kernels are not applicable for AA using BOLH. Instead, we rely on particular kernels defined for sets of vectors rather than for a single vector. Specifically, we consider kernels of the form (Rubner et al., 2001; Grauman, 2006):

$$K(P,Q) = \exp\big(-\frac{D(P,Q)^2}{\gamma}\big) \qquad (5)$$

where $D(P,Q)$ is the sum of the distances between the elements of the bag of local histograms associated to author P and the elements of the bag of histograms associated with author Q; γ is the scale parameter of K. Let $P = \{\mathbf{p}_1,\ldots,\mathbf{p}_k\}$ and $Q = \{\mathbf{q}_1,\ldots,\mathbf{q}_k\}$ be the elements of the bags of local histograms for instances P and Q, respectively, Table 1 presents the distance measures we consider for AA using local histograms.

Kernel	Distance
Diffusion	$D(P,Q) = \sum_{l=1}^{k} \arccos\left(\langle\sqrt{\mathbf{p}}_l \cdot \sqrt{\mathbf{q}}_l\rangle\right)$
EMD	$D(P,Q) = EMD(P,Q)$
Eucidean	$D(P,Q) = \sqrt{\sum_{l=1}^{k}(\mathbf{p}_l - \mathbf{q}_l).^2}$
χ^2	$D(P,Q) = \sqrt{\sum_{l=1}^{k} \frac{(\mathbf{p}_l - \mathbf{q}_l)^2}{(\mathbf{p}_l + \mathbf{q}_l)}}$

Table 1: Distance functions used to calculate the kernel defined in Equation (5).

Diffusion, Euclidean, and χ^2 kernels compare local histograms one to one, which means that the local histograms calculated at the same locations are compared to each other. We believe that for AA this is advantageous as it is expected that an author uses similar terms at similar locations of the document. The Earth mover's distance (EMD), on the other hand, is an estimate of the optimal cost in taking local histograms from Q to local histograms in P (Rubner et al., 2001); that is, this measure computes the optimal matching distance between local histograms from different authors that are not necessarily computed at similar locations.

5 Experiments and Results

For our experiments we considered the data set used in (Plakias and Stamatatos, 2008b; Plakias and Stamatatos, 2008a). This corpus is a subset of the RCV1 collection (Lewis et al., 2004) and comprises

documents authored by 10 authors. All of the documents belong to the same topic. Since this data set has predefined training and testing partitions, our results are comparable to those obtained by other researchers. There are 50 documents per author for training and 50 documents per author for testing.

We performed experiments with LOWBOW[3] representations at word and character-level. For the experiments with words, we took the top 2,500 most common words used across the training documents and obtained LOWBOW representations. We used this setting in agreement with previous work on AA (Houvardas and Stamatatos, 2006). For our character n-gram experiments, we obtained LOWBOW representations for character 3-grams (only n-grams of size $n = 3$ were used) considering the $2,500$ most common n-grams. Again, this setting was adopted in agreement with previous work on AA with character n-grams (Houvardas and Stamatatos, 2006; Plakias and Stamatatos, 2008b; Plakias and Stamatatos, 2008a; Luyckx and Daelemans, 2010). All our experiments use the SVM implementation provided by Canu et al. (2005).

5.1 Experimental settings

In order to compare our methods to related works we adopted the following experimental setting. We perform experiments using all of the training documents per author, that is, a balanced corpus (we call this setting **BC**). Next we evaluate the performance of classifiers over reduced training sets. We tried balanced reduced data sets with: $1, 3, 5$ and 10 documents per author (we call this configuration **RBC**). Also, we experimented with reduced-imbalanced data sets using the same imbalance rates reported in (Plakias and Stamatatos, 2008b; Plakias and Stamatatos, 2008a): we tried settings $2-10$, $5-10$, and $10-20$, where, for example, setting 2-10 means that we use at least 2 and at most 10 documents per author (we call this setting **IRBC**). **BC** setting represents the AA problem under ideal conditions, whereas settings **RBC** and **IRBC** aim at emulating a more realistic scenario, where limited sample documents are available and the whole data set is highly imbalanced (Plakias and Stamatatos, 2008b).

[3]We used LOWBOW code of G. Lebanon and Y. Mao available from http://www.cc.gatech.edu/∼ymao8/lowbow.htm

5.2 Experimental results in balanced data

We first compare the performance of the LOWBOW histogram representation to that of the traditional BOW representation. Table 2 shows the accuracy (i.e., percentage of documents in the test set that were associated to its correct author) for the BOW and LOWBOW histogram representations when using words and character n-grams information. For LOWBOW histograms, we report results with three different configurations for μ. As in (Lebanon et al., 2007), we consider uniformly distributed locations and we varied the number of locations that were included in each setting. We denote with k the number of local histograms. In preliminary experiments we tried several other values for k, although we found that representative results can be obtained with the values we considered here.

Method	Parameters	Words	Characters
BOW	-	78.2%	75.0%
LOWBOW	$k = 2; \sigma = 0.2$	75.8%	72.0%
LOWBOW	$k = 5; \sigma = 0.2$	77.4%	75.2%
LOWBOW	$k = 20; \sigma = 0.2$	77.4%	75.0%

Table 2: Authorship attribution accuracy for the BOW representation and LOWBOW histograms. Column 2 shows the parameters we used for the LOWBOW histograms; columns 3 and 4 show results using words and character n-grams, respectively.

From Table 2 we can see that the BOW representation is very effective, outperforming most of the LOWBOW histogram configurations. Despite a small difference in performance, BOW is advantageous over LOWBOW histograms because it is simpler to compute and it does not rely on parameter selection. Recall that the LOWBOW histogram representations are obtained by the combination of several local histograms calculated at different locations of the document, hence, it seems that the raw sum of local histograms results in a loss of useful information for representing documents. The worse performance was obtained when $k = 2$ local histograms are considered (see row 3 in Table 2). This result is somewhat expected since the larger the number of local histograms, the more LOWBOW histograms approach the BOW formulation (Lebanon et al., 2007).

We now describe the AA performance obtained when using the BOLH formulation; these results

are shown in Table 3. Most of the results from this table are superior to those reported in Table 2, showing that bags of local histograms are a better way to exploit the LOWBOW framework for AA. As expected, different kernels yield different results. However, the diffusion kernel outperformed most of the results obtained with other kernels; confirming the results obtained by other researchers (Lebanon et al., 2007; Lafferty and Lebanon, 2005).

Kernel	Euc.	Diffusion	EMD	χ^2
		Words		
Setting-1	78.6%	81.0%	75.0%	75.4%
Setting-2	77.6%	82.0%	76.8%	77.2%
Setting-3	79.2%	80.8%	77.0%	79.0%
		Characters		
Setting-1	83.4%	82.8%	84.4%	83.8%
Setting-2	83.4%	84.2%	82.2%	84.6%
Setting-3	83.6%	**86.4%**	81.0%	85.2%

Table 3: Authorship attribution accuracy when using bags of local histograms and different kernels for word-based and character-based representations. The **BC** data set is used. Settings 1, 2 and 3 correspond to $k = 2, 5$ and 20, respectively.

On average, the worse kernel was that based on the earth mover's distance (EMD), suggesting that the comparison of local histograms at different locations is not a fruitful approach (recall that this is the only kernel that compares local histograms at different locations). This result evidences that authors use similar word/character distributions at similar locations when writing different documents.

The best performance across settings and kernels was obtained with the diffusion kernel (in bold, column 3, row 9) (86.4%); that result is 8% higher than that obtained with the BOW representation and 9% better than the best configuration of LOWBOW histograms, see Table 2. Furthermore, that result is more than 5% higher than the best reported result in related work (80.8% as reported in (Plakias and Stamatatos, 2008b)). Therefore, the considered local histogram representations over character n-grams have proved to be very effective for AA.

One should note that, in general, better performance was obtained when using character-level rather than word-level information. This confirms the results already reported by other researchers that have used character-level and word-level information for AA (Houvardas and Stamatatos, 2006; Plakias and Stamatatos, 2008b; Plakias and Stamatatos, 2008a; Peng et al., 2003). We believe this can be attributed to the fact that character n-grams provide a representation for the document at a finer granularity, which can be better exploited with local histogram representations. Note that by considering 3-grams, words of length up to three are incorporated, and usually these words are function words (e.g., the, it, as, etc.), which are known to be indicative of writing style. Also, n-gram information is more dense in documents than word-level information. Hence, the local histograms are less sparse when using character-level information, which results in better AA performance.

				True author					
AC	AS	BL	DL	JM	JG	MM	MD	RS	TN
88	2	0	0	0	0	0	0	0	0
10	98	0	0	0	0	0	0	0	0
0	0	68	0	40	0	0	0	0	0
0	0	0	80	0	0	0	0	0	4
0	0	12	2	42	0	0	2	0	0
0	0	0	0	0	100	0	0	0	2
2	0	2	0	0	0	100	0	0	0
0	0	18	0	18	0	0	98	0	0
0	0	0	2	0	0	0	0	100	4
0	0	0	16	0	0	0	0	0	90

Table 4: Confusion matrix (in terms of percentages) for the best result in the **BC** corpus (i.e., last row, column 3 in Table 3). Columns show the true author for test documents and rows show the authors predicted by the SVM.

Table 4 shows the confusion matrix for the setting that reached the best results (i.e., column 3, last row in Table 3). From this table we can see that 8 out of the 10 authors were recognized with an accuracy higher or equal to 80%. For these authors sequential information seems to be particularly helpful. However, low recognition performance was obtained for authors BL (B. K. Lim) and JM (J. MacArtney). The SVM with BOW representation of character n-grams achieved recognition rates of 40% and 50% for BL and JM respectively. Thus, we can state that sequential information was indeed helpful for modeling BL writing style (improvement of 28%), although it is an author that resulted very difficult to model. On the other hand, local histograms were not very useful for identifying documents written by JM (made it worse by -8%). The largest improvement (38%) of local histograms over the BOW formulation was obtained for author TN (T. Nissen). This

result gives evidence that TN uses a similar distribution of words in similar locations across the documents he writes. These results are interesting, although we would like to perform a careful analysis of results in order to determine for what type of authors it would be beneficial to use local histograms, and what type of authors are better modeled with a standard BOW approach.

5.3 Experimental results in imbalanced data

In this section we report results with **RBC** and **IRBC** data sets, which aim to evaluate the performance of our methods in a realistic setting. For these experiments we compare the performance of the BOW, LOWBOW histogram and BOLH representations; for the latter, we considered the best setting as reported in Table 3 (i.e., an SVM with diffusion kernel and $k = 20$). Tables 5 and 6 show the AA performances when using word and character information, respectively.

We first analyze the results in the **RBC** data set (recall that for this data set we consider 1, 3, 5, 10, and 50, randomly selected documents per author). From Tables 5 and 6 we can see that BOW and LOWBOW histogram representations obtained similar performance to each other across the different training set sizes, which agree with results in Table 2 for the **BC** data sets. The best performance across the different configurations of the **RBC** data set was obtained with the BOLH formulation (row 6 in Tables 5 and 6). The improvements of local histograms over the BOW formulation vary across different settings and when using information at word-level and character-level. When using words (columns 2-6 in Table 5) the differences in performance are of 15.6%, 6.2%, 6.8%, 2.9%, 3.8% when using 1, 3, 5, 10 and 50 documents per author, respectively. Thus, it is evident that local histograms are more beneficial when less documents are considered. Here, the lack of information is compensated by the availability of several histograms per author.

When using character n-grams (columns 2-6 in Table 6) the corresponding differences in performance are of 5.4%, 6.4%, 6.4%, 6% and 11.4%, when using 1, 3, 5, 10, and 50 documents per author, respectively. In this case, the larger improvement was obtained when 50 documents per author are available; nevertheless, one should note that results using character-level information are, in general, significantly better than those obtained with word-level information; hence, improvements are expected to be smaller.

When we compare the results of the BOLH formulation with the best reported results elsewhere (c.f. last row 6 in Tables 5 and 6) (Plakias and Stamatatos, 2008b), we found that the improvements range from 14% to 30.2% when using character n-grams and from 1.2% to 26% when using words. The differences in performance are larger when less information is used (e.g., when 5 documents are used for training) and we believe the differences would be even larger if results for 1 and 3 documents were available. These are very positive results; for example, we can obtain almost 71% of accuracy, using local histograms of character n-grams when a single document is available per author (recall that we have used all of the test samples for evaluating the performance of our methods).

We now analyze the performance of the different methods when using the **IRBC** data set (columns 7-9 in Tables 5 and 6). The same pattern as before can be observed in experimental results for these data sets as well: BOW and LOWBOW histograms obtained comparable performance to each other and the BOLH formulation performed the best. The BOLH formulation outperforms state of the art approaches by a considerable margin that ranges from 10% to 27%. Again, better results were obtained when using character n-grams for the local histograms. With respect to **RBC** data sets, the BOLH at the character-level resulted very robust to the reduction of training set size and the highly imbalanced data.

Summarizing, the results obtained in **RBC** and **IRBC** data sets show that the use of local histograms is advantageous under challenging conditions. An SVM under the BOLH representation is less sensitive to the number of training examples available and to the imbalance of data than an SVM using the BOW representation. Our hypothesis for this behavior is that local histograms can be thought of as expanding training instances, because for each training instance in the BOW formulation we have $k-$training instances under BOLH. The benefits of such expansion become more notorious as the number of available documents per author decreases.

WORDS								
Data set	**Balanced**					**Imbalanced**		
Setting	*1-doc*	*3-docs*	*5-docs*	*10-docs*	*50-docs*	*2-10*	*5-10*	*10-20*
BOW	36.8%	57.1%	62.4%	69.9%	78.2%	62.3%	67.2%	71.2%
LOWBOW	37.9%	55.6%	60.5%	69.3%	77.4%	61.1%	67.4%	71.5%
Diffusion kernel	52.4%	63.3%	69.2%	72.8%	82.0%	66.6%	70.7%	74.1%
Reference	-	-	53.4%	67.8%	80.8%	49.2%	59.8%	63.0%

Table 5: AA accuracy in **RBC** (columns 2-6) and **IRBC** (columns 7-9) data sets when using words as terms. We report results for the BOW, LOWBOW histogram and BOLH representations. For reference (last row), we also include the best result reported in (Plakias and Stamatatos, 2008b), when available, for each configuration.

CHARACTER N-GRAMS								
Data set	**Balanced**					**Imbalanced**		
Setting	*1-doc*	*3-docs*	*5-docs*	*10-docs*	*50-docs*	*2-10*	*5-10*	*10-20*
BOW	65.3%	71.9%	74.2%	76.2%	75.0%	70.1%	73.4%	73.1%
LOWBOW	61.9%	71.6%	74.5%	73.8%	75.0%	70.8%	72.8%	72.1%
Diffusion kernel	70.7%	78.3%	80.6%	82.2%	86.4%	77.8%	80.5%	82.2%
Reference	-	-	50.4%	67.8%	76.6%	49.2%	59.8%	63.0%

Table 6: AA accuracy in the **RBC** and **IRBC** data sets when using character n-grams as terms.

6 Conclusions

We have described the use of local histograms (LH) over character n-grams for AA. LHs are enriched histogram representations that preserve sequential information in documents (in terms of the positions of terms in documents); we explored the suitability of LHs over n-grams at the *character-level* for AA. We showed evidence supporting our hypothesis that LHs are very helpful for AA; we believe that this is due to the fact that LOWBOW representations can uncover, to some extent, the writing preferences of authors. Our experimental results showed that LHs outperform traditional bag-of-words formulations and state of the art techniques in balanced, imbalanced, and reduced data sets. The improvements were larger in reduced and imbalanced data sets, which is a very positive result as in real AA applications one often faces highly imbalanced and small sample issues. Our results are promising and motivate further research on the use and extension of the LOWBOW framework for related tasks (e.g. authorship verification and plagiarism detection).

As future work we would like to explore the use of LOWBOW representations for profile-based AA and related tasks. Also, we would like to develop model selection strategies for learning what combination of hyperparameters works better for modeling each author.

Acknowledgments

We thank E. Stamatatos for making his data set available. Also, we are grateful for the thoughtful comments of L. A. Barrón and those of the anonymous reviewers. This work was partially supported by CONACYT under project grants 61335, and CB-2009-134186, and by UAB faculty development grant 3110841.

References

AMIDA. 2007. Augmented multi-party interaction with distance access. Available from `http://www.amidaproject.org/`, AMIDA Report.

S. Argamon and S. Levitan. 2005. Measuring the usefulness of function words for authorship attribution. In *Proceedings of the Joint Conference of the Association for Computers and the Humanities and the Association for Literary and Linguistic Computing*, Victoria, BC, Canada.

S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy. 2005. SVM and kernel methods Matlab toolbox. Perception Systmes et Information, INSA de Rouen, Rouen, France.

V. Chasanis, A. Kalogeratos, and A. Likas. 2009. Movie segmentation into scenes and chapters using locally weighted bag of visual words. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, pages 35:1–35:7, Santorini, Fira, Greece. ACM Press.

R. M. Coyotl-Morales, L. Villaseñor-Pineda, M. Montes-y-Gómez, and P. Rosso. 2006. Authorship attribution using word sequences. In *Proceedings of 11th*

Iberoamerican Congress on Pattern Recognition, volume 4225 of *LNCS*, pages 844–852, Cancun, Mexico. Springer.

D. Das and A. Martins. 2007. A survey on automatic text summarization. Available from: `http://www.cs.cmu.edu/~nasmith/LS2/das-martins.07.pdf`, Literature Survey for the Language and Statistics II course at Carnegie Mellon University.

O. de Vel, A. Anderson, M. Corney, and G. Mohay. 2001. Multitopic email authorship attribution forensics. In *Proceedings of the ACM Conference on Computer Security - Workshop on Data Mining for Security Applications*, Philadelphia, PA, USA.

K. Grauman. 2006. *Matching Sets of Features for Efficient Retrieval and Recognition*. Ph.D. thesis, Massachusetts Institute of Technology.

J. Houvardas and E. Stamatatos. 2006. N-gram feature selection for author identification. In *Proceedings of the 12th International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, volume 4183 of *LNCS*, pages 77–86, Varna, Bulgaria. Springer.

V. Keselj, F. Peng, N. Cercone, and C. Thomas. 2003. N-gram-based author profiles for authorship attribution. In *Proceedings of the Pacific Association for Computational Linguistics*, pages 255–264, Halifax, Canada.

M. Koppel, J. Schler, and S. Argamon. 2009. Computational methods in authorship attribution. *Journal of the American Society for Information Science and Technology*, 60:9–26.

J. Lafferty and G. Lebanon. 2005. Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research*, 6:129–163.

M. Lambers and C. J. Veenman. 2009. Forensic authorship attribution using compression distances to prototypes. In *Computational Forensics, Lecture Notes in Computer Science, Volume 5718. ISBN 978-3-642-03520-3. Springer Berlin Heidelberg, 2009, p. 13*, volume 5718 of *LNCS*, pages 13–24. Springer.

G. Lebanon, Y. Mao, and J. Dillon. 2007. The locally weighted bag of words framework for document representation. *Journal of Machine Learning Research*, 8:2405–2441.

D. Lewis, T. Yang, and F. Rose. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

K. Luyckx and W. Daelemans. 2010. The effect of author set size and data size in authorship attribution. *Literary and Linguistic Computing*, pages 1–21, August.

Y. Mao, J. Dillon, and G. Lebanon. 2007. Sequential document visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1208–1215.

F. Peng, D. Shuurmans, V. Keselj, and S. Wang. 2003. Language independent authorship attribution using character level language models. In *Proceedings of the 10th conference of the European chapter of the Association for Computational Linguistics*, volume 1, pages 267–274, Budapest, Hungary.

F. Peng, D. Shuurmans, and S. Wang. 2004. Augmenting naive Bayes classifiers with statistical language models. *Information Retrieval Journal*, 7(1):317–345.

S. R. Pillay and T. Solorio. 2010. Authorship attribution of web forum posts. In *Proceedings of the eCrime Researchers Summit (eCrime), 2010*, pages 1–7, Dallas, TX, USA. IEEE.

S. Plakias and E. Stamatatos. 2008a. Author identification using a tensor space representation. In *Proceedings of the 18th European Conference on Artificial Intelligence*, volume 178, pages 833–834, Patras, Greece. IOS Press.

S. Plakias and E. Stamatatos. 2008b. Tensor space models for authorship attribution. In *Proceedings of the 5th Hellenic Conference on Artificial Intelligence: Theories, Models and Applications*, volume 5138 of *LNCS*, pages 239–249, Syros, Greece. Springer.

R. Rifkin and A. Klautau. 2004. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141.

Y. Rubner, C. Tomasi, J. Leonidas, and J. Guibas. 2001. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121.

F. Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

J. Shawe-Taylor and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

E. Stamatatos. 2006a. Authorship attribution based on feature set subspacing ensembles. *International Journal on Artificial Intelligence Tools*, 15(5):823–838.

E. Stamatatos. 2006b. Ensemble-based author identification using character n-grams. In *Proceedings of the 3rd International Workshop on Text-based Information Retrieval*, pages 41–46, Riva del Garda, Italy.

E. Stamatatos. 2009a. Intrinsic plagiarism detection using character n-gram profiles. In *Proceedings of the 3rd International Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse, PAN'09*, pages 38–46, Donostia-San Sebastian, Spain.

E. Stamatatos. 2009b. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556.

M. Tearle, K. Taylor, and H. Demuth. 2008. An algorithm for automated authorship attribution using neural networks. *Literary and Linguist Computing*, 23(4):425–442.

Y. Zhao and J. Zobel. 2005. Effective and scalable authorship attribution using function words. In *Proceedings of 2nd Asian Information Retrieval Symposium*, volume 3689 of *LNCS*, pages 174–189, Jeju Island, Korea. Springer.

Word Maturity: Computational Modeling of Word Knowledge

Kirill Kireyev **Thomas K Landauer**

Pearson Education, Knowledge Technologies
Boulder, CO

{kirill.kireyev, tom.landauer}@pearson.com

Abstract

While computational estimation of difficulty of words in the lexicon is useful in many educational and assessment applications, the concept of scalar word difficulty and current corpus-based methods for its estimation are inadequate. We propose a new paradigm called *word meaning maturity* which tracks the degree of knowledge of each word at different stages of language learning. We present a computational algorithm for estimating word maturity, based on modeling language acquisition with Latent Semantic Analysis. We demonstrate that the resulting metric not only correlates well with external indicators, but captures deeper semantic effects in language.

1 Motivation

It is no surprise that through stages of language learning, different words are learned at different times and are known to different extents. For example, a common word like "dog" is familiar to even a first-grader, whereas a more advanced word like "focal" does not usually enter learners' vocabulary until much later. Although individual rates of learning words may vary between high- and low-performing students, it has been observed that "children […] acquire word meanings in roughly the same sequence" (Biemiller, 2008).

The aim of this work is to model the degree of knowledge of words at different learning stages. Such a metric would have extremely useful applications in personalized educational technologies, for the purposes of accurate assessment and personalized vocabulary instruction.

2 Rethinking Word Difficulty

Previously, related work in education and psychometrics has been concerned with measuring *word difficulty* or classifying words into different difficulty categories.

Examples of such approaches include creation of word lists for targeted vocabulary instruction at various grade levels that were compiled by educational experts, such as Nation (1993) or Biemiller (2008). Such word difficulty assignments are also implicitly present in some readability formulas that estimate difficulty of texts, such as Lexiles (Stenner, 1996), which include a lexical difficulty component based on the frequency of occurrence of words in a representative corpus, on the assumption that word difficulty is inversely correlated to corpus frequency. Additionally, research in psycholinguistics has attempted to outline and measure psycholinguistic dimensions of words such as *age-of-acquisition* and *familiarity*, which aim to track when certain words become known and how familiar they appear to an average person.

Importantly, all such word *difficulty* measures can be thought of as functions that assign a single scalar value to each word *w*:

$$difficulty : f(w) \rightarrow \mathbb{R} \qquad (1)$$

There are several important limitations to such metrics, regardless of whether they are derived from corpus frequency, expert judgments or other measures.

First, learning each word is a continual process, one that is interdependent with the rest of the vocabulary. Wolter (2001) writes:

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 299–308,
Portland, Oregon, June 19-24, 2011.

> […] Knowing a word is quite often not an either-or situation; some words are known well, some not at all, and some are known to varying degrees. […] How well a particular word is known may condition the connections made between that particular word and the other words in the mental lexicon.

Thus, instead of modeling *when* a particular word will become fully known, it makes more sense to model the *degree* to which a word is known at different levels of language exposure.

Second, word difficulty is inherently perspectival: the degree of word understanding depends not only on the word itself, but also on the sophistication of a given learner. Consider again the difference between "dog" and "focal": a typical first-grader will have much more difficulty understanding the latter word compared to the former, whereas a well-educated adult will be able to use these words with equal ease. Therefore, the degree, or *maturity,* of word knowledge is inherently a function of two parameters -- word *w* and learner level *l*:

$$maturity : f(w, l) \rightarrow \mathbb{R} \qquad (2)$$

As the level *l* increases (i.e. for more advanced learners), we would expect the degree of understanding of word *w* to approach its full value corresponding to perfect knowledge; this will happen at different rates for different words.

Ideally, we would obtain maturity values by testing word knowledge of learners across different levels (ages or school grades) for all the words in the lexicon. Such a procedure, however, is prohibitively expensive; so instead we would like to estimate word maturity by using computational models.

To summarize: our aim is to model the development of *meaning* of words as a function of increasing exposure to language, and ultimately - the degree to which the meaning of words at each stage of exposure resemble their "adult" meaning. We therefore define *word meaning maturity* to be the degree to which the understanding of the word (expected for the average learner of a particular level) resembles that of an ideal mature learner.

3 Modeling Word Meaning Acquisition with Latent Semantic Analysis

3.1 Latent Semantic Analysis (LSA)

An appealing choice for quantitatively modeling word meanings and their growth over time is Latent Semantic Analysis (LSA), an unsupervised method for representing word and document meaning in a multi-dimensional vector space.

The LSA vector representation is derived in an unsupervised manner, based on occurrence patterns of words in a large corpus of natural language documents. A Singular Value Decomposition on the high-dimensional matrix of word/document occurrence counts (*A*) in the corpus, followed by zeroing all but the largest *r* elements[1] of the diagonal matrix *S*, yields a lower-rank word vector matrix (*U)*. The dimensionality reduction has the effect of smoothing out incidental co-occurrences and preserving significant semantic relationships between words. The resulting word vectors[2] in *U* are positioned in such a way that semantically related words vectors point in similar directions or, equivalently, have higher cosine values between them. For more details, please refer to Landauer et al. (2007) and others.

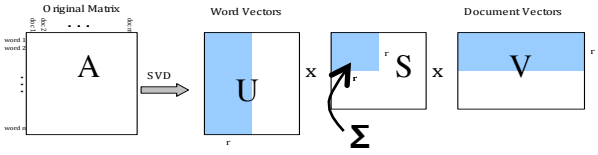


Figure 1. The SVD process in LSA illustrated. The original high-dimensional word-by-document matrix A is decomposed into word (U) and document (V) matrices of lower dimensionality.

In addition to merely measuring semantic relatedness, LSA has been shown to emulate the learning of word meanings from natural language (as can be evidenced by a broad range of applications from synonym tests to automated essay grading), at rates that resemble those of human learners (Laundauer et al, 1997). Landauer and Dumais (1997) have demonstrated empirically that LSA can emulate not only the rate of human language acquisition, but also more subtle phenomena, such as the effects of learning certain words on meaning of other words. LSA can model meaning with

[1] Typically the first approx. 300 dimensions are retained

[2] UΣ is used to project word vectors into V-space

high accuracy, as attested, for example, by 90% correlation with human judgments on assessing the quality of student essay content (Landauer, 2002).

3.2 Using LSA to Compute Word Maturity

In this work, the general procedure behind computationally estimating word maturity of a learner at a particular intermediate level (i.e. age or school grade level) is as follows:

1. Create an intermediate corpus for the given level. This corpus approximates the amount and sophistication of language encountered by a learner at the given level.
2. Build an LSA space on that corpus. The resulting LSA word vectors model the meaning of each word to the particular intermediate-level learner.
3. Compare the meaning representation of each word (its LSA vector) to the corresponding one in a reference model. The reference model is trained on a much larger corpus and approximates the word meanings by a mature adult learner.

We can repeat this process for each of a number of levels. These levels may directly correspond to school grades, learner ages or any other arbitrary gradations.

In summary, we estimate word maturity of a given word at a given learner level by comparing the word vector from an *intermediate* LSA model (trained on a corpus of size and sophistication comparable to that which a typical real student at the given level encounters) to the corresponding vector from a reference *adult* LSA model (trained on a larger corpus corresponding to a mature language learner). A high discrepancy between the vectors would suggest that an intermediate model's meaning of a particular word is quite different from the reference meaning, and thus the word maturity at the corresponding level is relatively low.

3.3 Procrustes Alignment (PA)

Comparing vectors across different LSA spaces is less straightforward, since the individual dimensions in LSA do not have a meaningful interpretation, and are an artifact of the content and ordering of the training corpus used. Therefore, direct comparisons across two different spaces, even of the same dimensionality, are meaningless, due to a mismatch in their coordinate systems.

Fortunately, we can employ a multivariate algebra technique known as Procrustes Alignment (or Procrustes Analysis) (PA) typically used to align two multivariate configurations of a corresponding set of points in two different geometric spaces. PA has been used in conjunction with LSA, for example, in cross-language information retrieval (Littman, 1998).

The basic idea behind PA is to derive a rotation matrix that allows one space to be rotated into the other. The rotation matrix is computed in such a way as to minimize the differences (namely: sum of squared distances) between corresponding points, which in the case of LSA can be common words or documents in the training set.

For more details, the reader is advised to consult chapter 5 of (Krzanowski, 2000) or similar literature on multivariate analysis. In summary, given two matrices containing coordinates of n corresponding points X and Y (and assuming mean-centering and equal number of dimensions, as is the case in this work), we would like to minimize the sum of squared distances between the points:

$$M^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} (x_{ij} - y_{ij})^2$$

We try to find an orthogonal rotation matrix Q, which minimizes M^2 by rotating Y relative to X. That matrix can be obtained by solving the equation:

$$M^2 = trace(XX' + YY' - 2XQ'Y')$$

It turns out that the solution to Q is given by VU', where $U\Sigma V'$ is the singular value decomposition of the matrix $X'Y$.

In our situation, where there are two spaces, *adult* and *intermediate*, the alignment points are the corresponding document vectors corresponding to the documents that the training corpora of the two models have in common (recall that the adult corpus is a superset of each of the intermediate corpora). The result of the Procrustes Alignment of the two spaces is effectively a joint LSA space containing two distinct word vectors for each word (e.g. "*dog1*", "*dog2*"), corresponding to the vectors from each of the original spaces. After

merging using Procrustes Alignment, the comparison of word meanings becomes a simple problem of comparing word vectors in the joint space using the standard cosine metric.

4 Implementation Details

In our experiments we used passages from the MetaMetrics Inc. 2002 corpus[3], largely consisting of educational and literary content representative of the reading material used in American schools at different grade levels. The average length of each passage is approximately 135 words.

The first-level intermediate corpus was composed of 6,000 text passages, intended for school grade 1 or below. The grade level is approximated using the Coleman-Liau readability formula (Coleman, 1975), which estimates the US grade level necessary to comprehend a given text, based on its average sentence and word length statistics:

$$CLI = 0.0588L - 0.296S - 15.8 \qquad (4)$$

where L is the average number of letters per 100 words and S is the average number of sentences per 100 words.

Each subsequent intermediate corpus contains additional 6,000 new passages of the next grade level, in addition to the previous corpus. In this way, we create 14 levels. The adult corpus is twice as large, and of same grade level range (0-14) as the largest intermediate corpus.

In summary, the following describes the size and makeup of the corpora used:

Corpus	Size (passages)	Approx. Grade Level (Coleman-Liau Index)
Intermediate 1	6,000	0.0 - 1.0
Intermediate 2	12,000	0.0 - 2.0
Intermediate 3	18,000	0.0 - 3.0
Intermediate 4	24,000	0.0 - 4.0
…		
Intermediate 14	84,000	0.0 - 14.0
Adult	168,000	0.0 - 14.0

Table 1. Size and makeup of corpora. used for LSA models.

The particular choice of the Coleman-Liau readability formula (CLI) is not essential; our experiments show that other well-known readability formulas (such as Lexiles) work equally well. All that is needed is some *approximate* ordering of passages by difficulty, in order to mimic the way typical human learners encounter progressively more difficult materials at successive school grades.

After creating the corpora, we:

1. Build LSA spaces on the adult and each of the intermediate corpora
2. Merge the intermediate space for level l with the adult space, using Procrustes Alignment. This results in a joint space with two sets of vectors: the versions from the intermediate space $\{vl_w\}$, and adult space $\{va_w\}$.
3. Compute the cosine in the joint space between the two word vectors for the given word w

$$wm(w, l) = cos\ (vl_w, va_w) \qquad (5)$$

In the cases where a word w has not been encountered in a given intermediate space, or in the rare cases where the cosine value falls below 0, the word maturity value is set to 0. Hence, the range for the word maturity function falls in the closed interval [0.0, 1.0]. A higher cosine value means greater similarity in meaning between the reference and intermediate spaces, which implies a more mature meaning of word w at the level l, i.e. higher word meaning maturity. The scores between discrete levels are interpolated, resulting in a continuous word maturity curve for each word.

Figure 1 below illustrates resulting word maturity curves for some of the words.

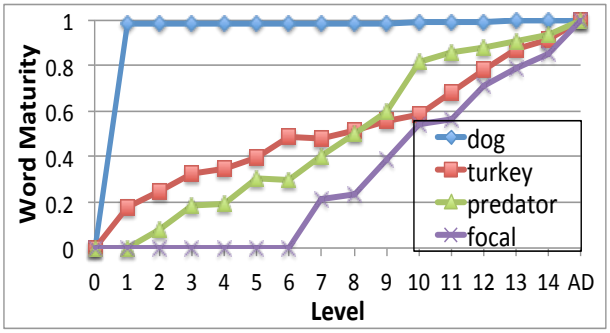


Figure 2. Word maturity curves for selected words.

Consistent with intuition, simple words like "dog" approach their adult meaning rather quickly, while "focal" takes much longer to become known to any degree.

An interesting example is "turkey", which has a noticeable plateau in the middle. This can be explained by the fact that this word has two distinct senses. Closer analysis of the corpus and the semantic near-neighbor word vectors at each in-

[3] We would like to acknowledge Jack Stenner and MetaMetrics for the use of their corpus.

termediate space, shows that earlier meaning deal almost exclusively with the first sense (bird), while later readings with the other (country). Therefore, even though the word "turkey" is quite prevalent in earlier readings, its full meaning is not learned until later levels. This demonstrates that our method takes into account the *meaning*, and not merely the frequency of occurrence.

5 Evaluation

5.1 Time-to-maturity

Evaluation of the word maturity metric against external data is not always straightforward because, to the best of our knowledge, data that contains word knowledge statistics *at different learner levels* does not exist. Instead, we often have to evaluate against external data consisting of scalar *difficulty* values (see Section 2 for discussion) for each word, such as age-of-acquisition norms described in the following subsection.

There are two ways to make such comparisons possible. One is to compute the word maturity at a particular level, obtaining a single number for each word. Another is by computing *time-to-maturity*: the minimum level (the value on the x-axis of the word maturity graph) at which the word maturity reaches[4] a particular threshold α:

$$ttm(w) = \min(l)\, s.t.\, wm(w,l) > \alpha \qquad (6)$$

Intuitively, this measure corresponds to the age in a learner's development when a given word becomes sufficiently understood. The parameter α can be estimated empirically (in practice α=0.45 gives good correlations with external measures). Since the values of word maturity are interpolated, the *ttm(w)* can take on fractional values.

It should be emphasized that such a collapsing of word *maturity* into a scalar value inherently results in loss of information; we only perform it in order to allow evaluation against external data sources.

As a baseline for these experiments we include word frequency, namely the document frequency of words in the adult corpus.

5.2 Age-of-Acquisition Norms

Age-of-Acquisition (AoA) is a psycholinguistic property of words originally reported by Carol & White (1973). Age of Acquisition approximates the age at which a word is first learned and has been proposed as a significant contributor to language and memory processes. With some exceptions, AoA norms are collected by subjective measures, typically by asking each of a large number of participants to estimate in years the age when they have learned the word. AoA estimates have been shown to be reliable and provide a valid estimate for the objective age at which a word is acquired; see (Davis, in press) for references and discussion.

In this experiment we compute Spearman correlations between time-to-maturity and two available collections of AoA norms: Gilhooly et al., (1980) norms[5], and Bristol norms[6] (Stadthagen-Gonzalez et al., 2010).

Measure	**Gilhooly** (n=1643)	**Bristol** (n=1402)
(-) Frequency	0.59	0.59
Time-to-Maturity (α=0.45)	*0.72*	*0.64*

Table 2. Correlations with Age of Acquisition norms.

5.3 Instruction Word Lists

In this experiment, we examine leveled lists of words, as created by Biemiller (2008) in the book entitled "Words Worth Teaching: Closing the Vocabulary Gap". Based on results of multiple-choice word comprehension tests administered to students of different grades as well as expert judgments, the author derives several word difficulty lists for vocabulary instruction in schools, including:

- Words known by most children in grade 2
- Words known by 40-80% of children in grade 2
- Words known by 40-80% of children in grade 6
- Words known by fewer than 40% of children in grade 6

One would expect the words in these four groups to increase in difficulty, in the order they are presented above.

[4] Values between discrete levels are obtained using piecewise linear interpolation

[5] http://www.psy.uwa.edu.au/mrcdatabase/uwa_mrc.htm

[6] http://language.psy.bris.ac.uk/bristol_norms.html

To verify how these word groups correspond to the word maturity metric, we assign each of the words in the four groups a difficulty rating 1-4 respectively, and measure the correlation with time-to-maturity.

Measure	Correlation
(-) Frequency	0.43
Time-to-maturity (α=0.45)	*0.49*

Table 3. Correlations with instruction word lists (n=4176).

The word maturity metric shows higher correlation with instruction word list norms than word frequency.

5.4 Text Complexity

Another way in which our metric can be evaluated is by examining the word maturity in texts that have been leveled, i.e. have been assigned ratings of difficulty. On average, we would expect more difficult texts to contain more difficult words. Thus, the correlation between text difficulty and our word maturity metric can serve as another validation of the metric.

For this purpose, we obtained a collection of readings that are used as reading comprehension tests by different state websites in the US[7]. The collection consists of 1,220 readings, each annotated with a US school grade level (in the range between 3-12) for which the reading is intended. The average length each passage was approximately 489 words.

In this experiment we computed the correlation of the grade level with time-to-maturity, and two other measures, namely:

- *Time-to-maturity*: average time-to-maturity of unique words in text (excluding stopwords) with α=0.45.
- *Coleman-Liau.* The Coleman-Liau readability index (Equation 4).
- *Frequency*. Average of corpus log-frequency for unique words in the text, excluding stopwords.

Measure	Correlation
Frequency (avg. of unique words)	0.60
Coleman-Liau	0.64
Time-to-maturity (α=0.45) (avg. of unique non-stopwords)	*0.70*

Table 4. Correlations of grade levels with different metrics.

6 Emphasis on Meaning

In this section, we would like to highlight certain properties of the LSA-based word maturity metric, particularly aiming to illustrate the fact that the metric tracks acquisition of meaning from exposure to language and not merely more shallow effects, such as word frequency in the training corpus.

6.1 Maturity based on Frequency

For a baseline that does not take meaning into account, let us construct a set of maturity-like curves based on frequency statistics alone. More specifically, we define the *frequency-maturity* for a particular word at a given level as the ratio of the number of occurrences at the intermediate corpus for that level (l) to the number of occurrences in the reference corpus (a):

$$fm(w,l) = \frac{num\_occur_l(w)}{num\_occur_a(w)}$$

Similarly to the original LSA-based word maturity metric, this ratio increases from 0 to 1 for each word as the amount of cumulative language exposure increases. The corpora used at each intermediate level are identical to the original word maturity model, but instead of creating LSA spaces we simply use the corpora to compute word frequency.

The following figure shows the Spearman correlations between the external measures used for experiments in Section 5, and time-to-maturity computed based on the two maturity metrics: the new frequency-based maturity and the original LSA-based word maturity.

[7] The collection was created as part of the "Aspects of Text Complexity" project funded by the Bill and Melinda Gates Foundation, 2010.

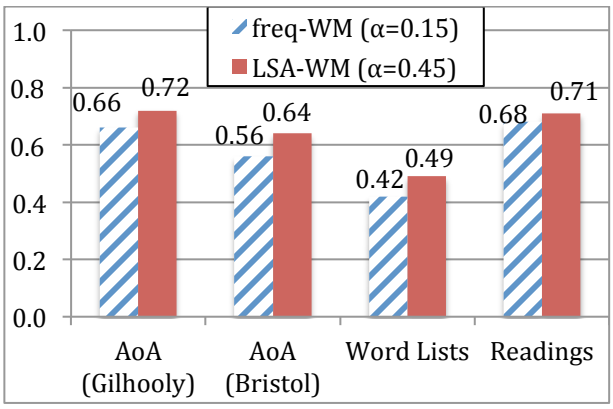


Figure 3. Correlations of word maturity computed using frequency (as well as the original) against external metrics described in Section 5.

The results indicate that the original LSA-based word maturity correlates better with real-world data than a maturity metric simply based on frequency.

6.2 Homographs

Another insight into the fact that the LSA-based word maturity metric tracks word meaning rather than mere frequency may be gained from analysis of words that are *homographs*: words that contain two or more unrelated meanings in the same written form, such as the word "turkey" illustrated in Section 4. (This is related to but distinct from the merely *polysemous* words that have several related meanings),

Because of the conflation of several unrelated meanings into the same orthographic form, homographs implicitly contain more semantic content in a single word. Therefore, one would expect the meaning of homographs to mature more slowly than would be predicted by frequency alone: all things being equal, a learner has to learn the meanings for all of the senses of a homograph word before the word can be considered fully known.

More specifically, one would expect the time-to-maturity of homographs to have greater values than words of similar frequency. To test this hypothesis, we obtained[8] a list 174 common English homographs. For each of them, we compared their time-to-maturity to the average time-to-maturity of words that have the same (+/- 1%) corpus frequency.

The results of a paired t-test confirms the hypothesis that the time-to-maturity of homographs is greater than other words of the same frequency, with the *p-value* = $5.9e^{-6}$. This is consistent with the observation that homographs will take longer to learn and serves as evidence that LSA-based word maturity approximates effects related to meaning.

6.3 Size of the Reference Corpus

Another area of investigation is the repercussions of the choice of the corpus for the reference (adult) model. The size (and content) of the corpus used to train the reference model is potentially important, since it affects the word maturity calculations, which are comparisons of the intermediate LSA spaces to the reference LSA space built on this corpus.

It is interesting to investigate how the word maturity model would be affected if the adult corpus were made significantly more sophisticated. If the word maturity metric were simply based on word frequency (including the frequency-based maturity baseline described in Section 6.1), one would expect the word maturity of the words at each level to *decrease* significantly if the reference model is made significantly larger, since each intermediate level will have encountered fewer words by comparison. Intuition about language learning, however, tells us that with enough language exposure a learner learns virtually all there is to know about any particular word; after the word reaches its adult maturity, subsequent encounters of natural readings do little to further change the knowledge of that word. Therefore, if word maturity were tracking something similar to real word knowledge, one would expect the word maturity for most words to plateau over time, and subsequently not change significantly, no matter how sophisticated the reference model becomes.

To evaluate this inquiry we created a reference corpus that is twice as large as before (four times as large and of the same difficulty range as the corpus for the last intermediate level), containing roughly 329,000 passages. We computed the word maturity model using this larger reference corpus, while keeping all the original intermediate corpora of the same size and content.

The results show that the average word maturity of words at the last intermediate level (14) de-

[8] http://en.wikipedia.org/wiki/List_of_English_homographs

creases by less than 14% as a result of doubling the adult corpus. Furthermore, this number is as low as 6%, if one only considers more common words that occur 50 times or more in the corpus. This relatively small difference, in spite of a two-fold increase of the adult corpus, is consistent with the idea that word knowledge should approach a plateau, after which further exposure to language does little to change most word meanings.

6.4 Integration into Lexicon

Another important consideration with respect to word learning mentioned in Wotler (2001), is the "connections made between [a] particular word and the other words in the mental lexicon." One implication of that is that measuring word maturity must take into account the way words in the language are integrated with other words.

One way to test this effect is to introduce readings where a large part of the important vocabulary is not well known to learners at a given level. One would expect learning to be impeded when the learning materials are inappropriate for the learner level.

This can be simulated in the word maturity model by rearranging the order of some of the training passages, by introducing certain advanced passages at a very early level. If the results of the word maturity metric were merely based on frequency, such a reordering would have no effect on the maturity of important words (measured after all the passages containing these words have been encountered), since the total number of relevant word encounters does not change as a result of this reshuffling. If, however, the metric reflected at least some degree of semantics, we would expect word maturities for important words in these readings to be lower as a result of such rearranging, due to the fact that they are being introduced in contexts consisting of words that are not well known at the early levels.

To test this effect, we first collected all passages in the training corpus of intermediate models containing some advanced words from different topics, namely: "chromosome", "neutron" and "filibuster" together with their plural variants. We changed the order of inclusion of these 89 passages into the intermediate models in each of the two following ways:

1. All the passages were introduced at the first level (*l=1*) intermediate corpus
2. All the passages were introduced at the last level (*l=14*) intermediate corpus.

This resulted in two new variants of word maturity models, which were computed in all the same ways as before, except that all of these 89 advanced passages were introduced either at the very first level or at the very last level. We then computed the word maturity at the levels they were introduced. The hypothesis consistent with a meaning-based maturity method would be that less learning (i.e. lower word maturity) of the relevant words will occur when passages are introduced prematurely (at level 1). Table 5 shows the word maturities measured for each of those cases, at the level (*1* or *14*) when all of the passages have been introduced.

Word	**Introduced at *l=1*** (WM at *l=1*)	**Introduced at *l=14*** (WM at *l=14*)
chromosome	0.51	0.73
neutron	0.51	0.72
filibuster	0.58	0.85

Table 5. Word maturity of words resulting when all the relevant passages are introduced early vs late.

Indeed, the results show lower word maturity values when advanced passages are introduced too early, and higher ones when the passages are introduced at a later stage, when the rest of the supporting vocabulary is known.

7 Conclusion

We have introduced a new metric for estimating the degree of knowledge of words by learners at different levels. We have also proposed and evaluated an implementation of this metric using Latent Semantic Analysis.

The implementation is based on unsupervised word meaning acquisition from natural text, from corpora that resemble in volume and complexity the reading materials a typical human learner might encounter.

The metric correlates better than word frequency to a range of external measures, including vocabulary word lists, psycholinguistic norms and leveled texts. Furthermore, we have shown that the metric is based on word meaning (to the extent that it can be approximated with LSA), and not merely on shallow measures like word frequency.

Many interesting research questions still remain pertaining to the best way to select and partition the training corpora, align adult and intermediate LSA models, correlate the results with real school grade levels, as well as other free parameters in the model. Nevertheless, we have shown that LSA can be employed to usefully mimic model word knowledge. The models are currently used (at Pearson Education) to create state-of-the-art personalized vocabulary instruction and assessment tools.

References

Andrew Biemiller (2008). *Words Worth Teaching.* Columbus, OH: SRA/McGraw-Hill.

John B. Carrol and M. N. White (1973). Age of acquisition norms for 220 picturable nouns. *Journal of Verbal Learning & Verbal Behavior*, 12, 563-576.

Meri Coleman and T.L. Liau (1975). *A computer readability formula designed for machine scoring*, Journal of Applied Psychology, Vol. 60, pp. 283-284.

Ken J. Gilhooly and R. H. Logie (1980). Age of acquisition, imagery, concreteness, familiarity and ambiguity measures for 1944 words. *Behaviour Research Methods & Instrumentation*, 12, 395-427.

Wojtek J. Krzanowski (2000) *Principles of Multivariate Analysis: A User's Perspective* (Oxford Statistical Science Series). Oxford University Press, USA.

Thomas K Landauer and Susan Dumais (1997). A solution to Plato's problem: The Latent Semantic Analysis Theory of the Acquisition, Induction, and Representation of Knowledge. *Psychological Review*, 104, pp 211-240.

Thomas K Landauer (2002). On the Computation Basis of Learning and Cognition: Arguments from LSA. In N. Ross (Ed.), *The Psychology of Learning and Motivation*, 41, 43-84.

Thomas K Landauer, Danielle S. McNamara, Simon Dennis, and Walter Kintsch (2007). *Handbook of Latent Semantic Analysis.* Lawrence Erlbaum.

Paul Nation (1993). Measuring readiness for simplified material: a test of the first 1,000 words of English. In *Simplification: Theory and Application* M. L. Tickoo (ed.), RELC Anthology Series 31: 193-203.

Hans Stadthagen-Gonzalez and C. J. Davis (2006). The Bristol Norms for Age of Acquisition, Imageability and Familiarity. *Behavior Research Methods*, 38, 598-605.

A. Jackson Stenner (1996). Measuring Reading Comprehension with the Lexile Framework. *Forth North American Conference on Adolescent/Adult Literacy.*

Brent Wolter (2001). Comparing the L1 and L2 Mental Lexicon. *Studies in Second Language Acquisition.* Cambridge University Press.

Finding Deceptive Opinion Spam by Any Stretch of the Imagination

Myle Ott **Yejin Choi** **Claire Cardie** **Jeffrey T. Hancock**
Department of Computer Science, Cornell University, Ithaca, NY 14853
Department of Communication, Cornell University, Ithaca, NY 14853
{myleott,ychoi,cardie}@cs.cornell.edu jth34@cornell.edu

Abstract

Consumers increasingly rate, review and research products online (Jansen, 2010; Litvin et al., 2008). Consequently, websites containing consumer reviews are becoming targets of *opinion spam*. While recent work has focused primarily on manually identifiable instances of opinion spam, in this work we study *deceptive opinion spam*—fictitious opinions that have been deliberately written to sound authentic. Integrating work from psychology and computational linguistics, we develop and compare three approaches to detecting deceptive opinion spam, and ultimately develop a classifier that is nearly 90% accurate on our *gold-standard* opinion spam dataset. Based on feature analysis of our learned models, we additionally make several theoretical contributions, including revealing a relationship between deceptive opinions and imaginative writing.

1 Introduction

With the ever-increasing popularity of review websites that feature user-generated opinions (e.g., TripAdvisor[1] and Yelp[2]), there comes an increasing potential for monetary gain through *opinion spam*—inappropriate or fraudulent reviews. Opinion spam can range from annoying self-promotion of an unrelated website or blog to deliberate review fraud, as in the recent case[3] of a Belkin employee who hired people to write positive reviews for an otherwise poorly reviewed product.[4]

While other kinds of spam have received considerable computational attention, regrettably there has been little work to date (see Section 2) on opinion spam detection. Furthermore, most previous work in the area has focused on the detection of DISRUPTIVE OPINION SPAM—uncontroversial instances of spam that are easily identified by a human reader, e.g., advertisements, questions, and other irrelevant or non-opinion text (Jindal and Liu, 2008). And while the presence of disruptive opinion spam is certainly a nuisance, the risk it poses to the user is minimal, since the user can always choose to ignore it.

We focus here on a potentially more insidious type of opinion spam: DECEPTIVE OPINION SPAM—fictitious opinions that have been deliberately written to sound authentic, in order to deceive the reader. For example, one of the following two hotel reviews is truthful and the other is *deceptive opinion spam*:

1. I have stayed at many hotels traveling for both business and pleasure and I can honestly stay that The James is tops. The service at the hotel is first class. The rooms are modern and very comfortable. The location is perfect within walking distance to all of the great sights and restaurants. Highly recommend to both business travellers and couples.
2. My husband and I stayed at the James Chicago Hotel for our anniversary. This place is fantastic! We knew as soon as we arrived we made the right choice! The rooms are BEAUTIFUL and the staff very attentive and wonderful!! The area of the hotel is great, since I love to shop I couldn't ask for more!! We will definatly be

[1] http://tripadvisor.com
[2] http://yelp.com
[3] http://news.cnet.com/8301-1001_3-10145399-92.html
[4] It is also possible for opinion spam to be negative, potentially in order to sully the reputation of a competitor.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 309–319,
Portland, Oregon, June 19-24, 2011.

back to Chicago and we will for sure be back to the James Chicago.

Typically, these deceptive opinions are neither easily ignored nor even identifiable by a human reader;[5] consequently, there are few good sources of labeled data for this research. Indeed, in the absence of gold-standard data, related studies (see Section 2) have been forced to utilize ad hoc procedures for evaluation. In contrast, one contribution of the work presented here is the creation of the first large-scale, publicly available[6] dataset for deceptive opinion spam research, containing 400 truthful and 400 *gold-standard* deceptive reviews.

To obtain a deeper understanding of the nature of deceptive opinion spam, we explore the relative utility of three potentially complementary framings of our problem. Specifically, we view the task as: (a) a standard *text categorization* task, in which we use n-gram–based classifiers to label opinions as either deceptive or truthful (Joachims, 1998; Sebastiani, 2002); (b) an instance of *psycholinguistic deception detection*, in which we expect deceptive statements to exemplify the psychological effects of lying, such as increased negative emotion and psychological distancing (Hancock et al., 2008; Newman et al., 2003); and, (c) a problem of *genre identification*, in which we view deceptive and truthful writing as sub-genres of imaginative and informative writing, respectively (Biber et al., 1999; Rayson et al., 2001).

We compare the performance of each approach on our novel dataset. Particularly, we find that machine learning classifiers trained on features traditionally employed in (a) psychological studies of deception and (b) genre identification are both outperformed at statistically significant levels by n-gram–based text categorization techniques. Notably, a combined classifier with both n-gram and psychological deception features achieves nearly 90% cross-validated accuracy on this task. In contrast, we find deceptive opinion spam detection to be well beyond the capabilities of most human judges, who perform roughly at-chance—a finding that is consistent with decades of traditional deception detection research (Bond and DePaulo, 2006).

[5]The second example review is deceptive opinion spam.

[6]Available by request at: `http://www.cs.cornell.edu/~myleott/op_spam`

Additionally, we make several theoretical contributions based on an examination of the feature weights learned by our machine learning classifiers. Specifically, we shed light on an ongoing debate in the deception literature regarding the importance of considering the context and motivation of a deception, rather than simply identifying a universal set of deception cues. We also present findings that are consistent with recent work highlighting the difficulties that liars have encoding spatial information (Vrij et al., 2009). Lastly, our study of deceptive opinion spam detection as a genre identification problem reveals relationships between deceptive opinions and imaginative writing, and between truthful opinions and informative writing.

The rest of this paper is organized as follows: in Section 2, we summarize related work; in Section 3, we explain our methodology for gathering data and evaluate human performance; in Section 4, we describe the features and classifiers employed by our three automated detection approaches; in Section 5, we present and discuss experimental results; finally, conclusions and directions for future work are given in Section 6.

2 Related Work

Spam has historically been studied in the contexts of e-mail (Drucker et al., 2002), and the Web (Gyöngyi et al., 2004; Ntoulas et al., 2006). Recently, researchers have began to look at *opinion spam* as well (Jindal and Liu, 2008; Wu et al., 2010; Yoo and Gretzel, 2009).

Jindal and Liu (2008) find that opinion spam is both widespread and different in nature from either e-mail or Web spam. Using product review data, and in the absence of gold-standard deceptive opinions, they train models using features based on the review text, reviewer, and product, to distinguish between *duplicate* opinions[7] (considered deceptive spam) and *non-duplicate* opinions (considered truthful). Wu et al. (2010) propose an alternative strategy for detecting deceptive opinion spam in the absence

[7]Duplicate (or near-duplicate) opinions are opinions that appear more than once in the corpus with the same (or similar) text. While these opinions are likely to be deceptive, they are unlikely to be representative of deceptive opinion spam in general. Moreover, they are potentially detectable via off-the-shelf plagiarism detection software.

of gold-standard data, based on the distortion of popularity rankings. Both of these heuristic evaluation approaches are unnecessary in our work, since we compare *gold-standard* deceptive and truthful opinions.

Yoo and Gretzel (2009) gather 40 truthful and 42 deceptive hotel reviews and, using a standard statistical test, manually compare the psychologically relevant linguistic differences between them. In contrast, we create a much larger dataset of 800 opinions that we use to develop and evaluate *automated* deception classifiers.

Research has also been conducted on the related task of *psycholinguistic deception detection*. Newman et al. (2003), and later Mihalcea and Strapparava (2009), ask participants to give both their true and untrue views on personal issues (e.g., their stance on the death penalty). Zhou et al. (2004; 2008) consider computer-mediated deception in role-playing games designed to be played over instant messaging and e-mail. However, while these studies compare n-gram–based deception classifiers to a random guess baseline of 50%, we additionally evaluate and compare two other computational approaches (described in Section 4), as well as the performance of human judges (described in Section 3.3).

Lastly, automatic approaches to determining *review quality* have been studied—directly (Weimer et al., 2007), and in the contexts of helpfulness (Danescu-Niculescu-Mizil et al., 2009; Kim et al., 2006; O'Mahony and Smyth, 2009) and credibility (Weerkamp and De Rijke, 2008). Unfortunately, most measures of quality employed in those works are based exclusively on human judgments, which we find in Section 3 to be poorly calibrated to detecting deceptive opinion spam.

3 Dataset Construction and Human Performance

While truthful opinions are ubiquitous online, deceptive opinions are difficult to obtain without resorting to heuristic methods (Jindal and Liu, 2008; Wu et al., 2010). In this section, we report our efforts to gather (and validate with human judgments) the first publicly available opinion spam dataset with *gold-standard* deceptive opinions.

Following the work of Yoo and Gretzel (2009), we compare truthful and deceptive **positive** reviews for hotels found on TripAdvisor. Specifically, we mine all 5-star truthful reviews from the 20 most popular hotels on TripAdvisor[8] in the Chicago area.[9] Deceptive opinions are gathered for those same 20 hotels using Amazon Mechanical Turk[10] (AMT). Below, we provide details of the collection methodologies for deceptive (Section 3.1) and truthful opinions (Section 3.2). Ultimately, we collect 20 truthful and 20 deceptive opinions for each of the 20 chosen hotels (800 opinions total).

3.1 Deceptive opinions via Mechanical Turk

Crowdsourcing services such as AMT have made large-scale data annotation and collection efforts financially affordable by granting anyone with basic programming skills access to a marketplace of anonymous online workers (known as *Turkers*) willing to complete small tasks.

To solicit gold-standard **deceptive** opinion spam using AMT, we create a pool of 400 *Human Intelligence Tasks* (HITs) and allocate them evenly across our 20 chosen hotels. To ensure that opinions are written by unique authors, we allow only a single submission per Turker. We also restrict our task to Turkers who are located in the United States, and who maintain an approval rating of at least 90%. Turkers are allowed a maximum of 30 minutes to work on the HIT, and are paid one US dollar for an accepted submission.

Each HIT presents the Turker with the name and website of a hotel. The HIT instructions ask the Turker to assume that they work for the hotel's marketing department, and to pretend that their boss wants them to write a fake review (as if they were a customer) to be posted on a travel review website; additionally, the review needs to sound realistic and portray the hotel in a positive light. A disclaimer

[8] TripAdvisor utilizes a proprietary ranking system to assess hotel popularity. We chose the 20 hotels with the greatest number of reviews, irrespective of the TripAdvisor ranking.

[9] It has been hypothesized that popular offerings are less likely to become targets of deceptive opinion spam, since the relative impact of the spam in such cases is small (Jindal and Liu, 2008; Lim et al., 2010). By considering only the most popular hotels, we hope to minimize the risk of mining opinion spam and labeling it as truthful.

[10] `http://mturk.com`

Time spent t (minutes)	
All submissions	$count$: 400 t_{min}: 0.08, t_{max}: 29.78 $\bar{t}$: 8.06, s: 6.32
Length ℓ (words)	
All submissions	ℓ_{min}: 25, ℓ_{max}: 425 $\bar{\ell}$: 115.75, s: 61.30
Time spent $t < 1$	$count$: 47 ℓ_{min}: 39, ℓ_{max}: 407 $\bar{\ell}$: 113.94, s: 66.24
Time spent $t \geq 1$	$count$: 353 ℓ_{min}: 25, ℓ_{max}: 425 $\bar{\ell}$: 115.99, s: 60.71

Table 1: Descriptive statistics for 400 deceptive opinion spam submissions gathered using AMT. s corresponds to the sample standard deviation.

indicates that any submission found to be of insufficient quality (e.g., written for the wrong hotel, unintelligible, unreasonably short,[11] plagiarized,[12] etc.) will be rejected.

It took approximately 14 days to collect 400 satisfactory deceptive opinions. Descriptive statistics appear in Table 1. Submissions vary quite dramatically both in length, and time spent on the task. Particularly, nearly 12% of the submissions were completed in *under one minute*. Surprisingly, an independent two-tailed t-test between the mean length of these submissions ($\bar{\ell}_{t<1}$) and the other submissions ($\bar{\ell}_{t\geq 1}$) reveals no significant difference ($p = 0.83$). We suspect that these "*quick*" users may have started working prior to having formally accepted the HIT, presumably to circumvent the imposed time limit. Indeed, the quickest submission took just 5 seconds and contained 114 words.

3.2 Truthful opinions from TripAdvisor

For truthful opinions, we mine all 6,977 reviews from the 20 most popular Chicago hotels on TripAdvisor. From these we eliminate:

- 3,130 non-5-star reviews;
- 41 non-English reviews;[13]
- 75 reviews with fewer than 150 characters since, by construction, deceptive opinions are at least 150 characters long (see footnote 11 in Section 3.1);
- 1,607 reviews written by *first-time authors*—new users who have not previously posted an opinion on TripAdvisor—since these opinions are more likely to contain opinion spam, which would reduce the integrity of our truthful review data (Wu et al., 2010).

Finally, we balance the number of truthful and deceptive opinions by selecting 400 of the remaining 2,124 truthful reviews, such that the document lengths of the selected truthful reviews are similarly distributed to those of the deceptive reviews. Work by Serrano et al. (2009) suggests that a *log-normal* distribution is appropriate for modeling document lengths. Thus, for each of the 20 chosen hotels, we select 20 truthful reviews from a log-normal (left-truncated at 150 characters) distribution fit to the lengths of the deceptive reviews.[14] Combined with the 400 deceptive reviews gathered in Section 3.1 this yields our final dataset of 800 reviews.

3.3 Human performance

Assessing human deception detection performance is important for several reasons. First, there are few other baselines for our classification task; indeed, related studies (Jindal and Liu, 2008; Mihalcea and Strapparava, 2009) have only considered a random guess baseline. Second, assessing human performance is necessary to validate the deceptive opinions gathered in Section 3.1. If human performance is low, then our deceptive opinions are convincing, and therefore, deserving of further attention.

Our initial approach to assessing human performance on this task was with Mechanical Turk. Unfortunately, we found that some Turkers selected among the choices seemingly at random, presumably to maximize their hourly earnings by obviating the need to read the review. While a similar effect has been observed previously (Akkaya et al., 2010), there remains no universal solution.

Instead, we solicit the help of three volunteer undergraduate university students to make judgments on a subset of our data. This balanced subset, corresponding to the first fold of our cross-validation

[11] A submission is considered unreasonably short if it contains fewer than 150 characters.

[12] Submissions are individually checked for plagiarism at `http://plagiarisma.net`.

[13] Language is determined using `http://tagthe.net`.

[14] We use the R package GAMLSS (Rigby and Stasinopoulos, 2005) to fit the left-truncated log-normal distribution.

			TRUTHFUL			DECEPTIVE		
		Accuracy	**P**	**R**	**F**	**P**	**R**	**F**
HUMAN	JUDGE 1	**61.9%**	57.9	87.5	**69.7**	74.4	36.3	48.7
	JUDGE 2	56.9%	53.9	**95.0**	68.8	**78.9**	18.8	30.3
	JUDGE 3	53.1%	52.3	70.0	59.9	54.7	36.3	43.6
META	MAJORITY	58.1%	54.8	92.5	68.8	76.0	23.8	36.2
	SKEPTIC	60.6%	**60.8**	60.0	60.4	60.5	**61.3**	**60.9**

Table 2: Performance of three human judges and two meta-judges on a subset of 160 opinions, corresponding to the first fold of our cross-validation experiments in Section 5. Boldface indicates the largest value for each column.

experiments described in Section 5, contains all 40 reviews from each of four randomly chosen hotels. Unlike the Turkers, our student volunteers are not offered a monetary reward. Consequently, we consider their judgements to be more honest than those obtained via AMT.

Additionally, to test the extent to which the individual human judges are biased, we evaluate the performance of two virtual meta-judges. Specifically, the MAJORITY meta-judge predicts "*deceptive*" when at least two out of three human judges believe the review to be deceptive, and the SKEPTIC meta-judge predicts "*deceptive*" when *any* human judge believes the review to be deceptive.

Human and meta-judge performance is given in Table 2. It is clear from the results that human judges are not particularly effective at this task. Indeed, a two-tailed binomial test fails to reject the null hypothesis that JUDGE 2 and JUDGE 3 perform at-chance ($p = 0.003, 0.10, 0.48$ for the three judges, respectively). Furthermore, all three judges suffer from *truth-bias* (Vrij, 2008), a common finding in deception detection research in which human judges are more likely to classify an opinion as truthful than deceptive. In fact, JUDGE 2 classified fewer than 12% of the opinions as deceptive! Interestingly, this bias is effectively smoothed by the SKEPTIC meta-judge, which produces nearly perfectly class-balanced predictions. A subsequent reevaluation of human performance on this task suggests that the truth-bias can be reduced if judges are given the class-proportions in advance, although such prior knowledge is unrealistic; and ultimately, performance remains similar to that of Table 2.

Inter-annotator agreement among the three judges, computed using Fleiss' kappa, is 0.11. While there is no precise rule for interpreting kappa scores, Landis and Koch (1977) suggest that scores in the range (0.00, 0.20] correspond to "*slight agreement*" between annotators. The largest pairwise Cohen's kappa is 0.12, between JUDGE 2 and JUDGE 3—a value far below generally accepted pairwise agreement levels. We suspect that agreement among our human judges is so low *precisely because* humans are poor judges of deception (Vrij, 2008), and therefore they perform nearly at-chance respective to one another.

4 Automated Approaches to Deceptive Opinion Spam Detection

We consider three automated approaches to detecting deceptive opinion spam, each of which utilizes classifiers (described in Section 4.4) trained on the dataset of Section 3. The features employed by each strategy are outlined here.

4.1 Genre identification

Work in computational linguistics has shown that the frequency distribution of *part-of-speech* (POS) tags in a text is often dependent on the genre of the text (Biber et al., 1999; Rayson et al., 2001). In our genre identification approach to deceptive opinion spam detection, we test if such a relationship exists for truthful and deceptive reviews by constructing, for each review, features based on the frequencies of each POS tag.[15] These features are also intended to provide a good baseline with which to compare our other automated approaches.

4.2 Psycholinguistic deception detection

The *Linguistic Inquiry and Word Count* (LIWC) software (Pennebaker et al., 2007) is a popular automated text analysis tool used widely in the social sciences. It has been used to detect personality

[15] We use the Stanford Parser (Klein and Manning, 2003) to obtain the relative POS frequencies.

traits (Mairesse et al., 2007), to study tutoring dynamics (Cade et al., 2010), and, most relevantly, to analyze deception (Hancock et al., 2008; Mihalcea and Strapparava, 2009; Vrij et al., 2007).

While LIWC does not include a text classifier, we can create one with features derived from the LIWC output. In particular, LIWC counts and groups the number of instances of nearly 4,500 keywords into 80 psychologically meaningful dimensions. We construct one feature for each of the 80 LIWC dimensions, which can be summarized broadly under the following four categories:

1. Linguistic processes: Functional aspects of text (e.g., the average number of words per sentence, the rate of misspelling, swearing, etc.)
2. Psychological processes: Includes all social, emotional, cognitive, perceptual and biological processes, as well as anything related to time or space.
3. Personal concerns: Any references to work, leisure, money, religion, etc.
4. Spoken categories: Primarily filler and agreement words.

While other features have been considered in past deception detection work, notably those of Zhou et al. (2004), early experiments found LIWC features to perform best. Indeed, the LIWC2007 software used in our experiments subsumes most of the features introduced in other work. Thus, we focus our psycholinguistic approach to deception detection on LIWC-based features.

4.3 Text categorization

In contrast to the other strategies just discussed, our text categorization approach to deception detection allows us to model both content and context with n-gram features. Specifically, we consider the following three n-gram feature sets, with the corresponding features lowercased and unstemmed: UNIGRAMS, BIGRAMS$^+$, TRIGRAMS$^+$, where the superscript $^+$ indicates that the feature set subsumes the preceding feature set.

4.4 Classifiers

Features from the three approaches just introduced are used to train Naïve Bayes and Support Vector Machine classifiers, both of which have performed well in related work (Jindal and Liu, 2008; Mihalcea and Strapparava, 2009; Zhou et al., 2008).

For a document $\vec{x}$, with label y, the *Naïve Bayes* (NB) classifier gives us the following decision rule:

$$\hat{y} = \arg\max_c \Pr(y = c) \cdot \Pr(\vec{x} \mid y = c) \quad (1)$$

When the class prior is *uniform*, for example when the classes are balanced (as in our case), (1) can be simplified to the maximum likelihood classifier (Peng and Schuurmans, 2003):

$$\hat{y} = \arg\max_c \Pr(\vec{x} \mid y = c) \quad (2)$$

Under (2), both the NB classifier used by Mihalcea and Strapparava (2009) and the language model classifier used by Zhou et al. (2008) are equivalent. Thus, following Zhou et al. (2008), we use the SRI Language Modeling Toolkit (Stolcke, 2002) to estimate individual language models, $\Pr(\vec{x} \mid y = c)$, for truthful and deceptive opinions. We consider all three n-gram feature sets, namely UNIGRAMS, BIGRAMS$^+$, and TRIGRAMS$^+$, with corresponding language models smoothed using the interpolated Kneser-Ney method (Chen and Goodman, 1996).

We also train *Support Vector Machine* (SVM) classifiers, which find a high-dimensional separating hyperplane between two groups of data. To simplify feature analysis in Section 5, we restrict our evaluation to *linear* SVMs, which learn a weight vector $\vec{w}$ and bias term b, such that a document $\vec{x}$ can be classified by:

$$\hat{y} = sign(\vec{w} \cdot \vec{x} + b) \quad (3)$$

We use SVM^{light} (Joachims, 1999) to train our linear SVM models on all three approaches and feature sets described above, namely POS, LIWC, UNIGRAMS, BIGRAMS$^+$, and TRIGRAMS$^+$. We also evaluate every combination of these features, but for brevity include only LIWC+BIGRAMS$^+$, which performs best. Following standard practice, document vectors are normalized to unit-length. For LIWC+BIGRAMS$^+$, we unit-length normalize LIWC and BIGRAMS$^+$ features individually before combining them.

			TRUTHFUL			DECEPTIVE		
Approach	**Features**	**Accuracy**	**P**	**R**	**F**	**P**	**R**	**F**
GENRE IDENTIFICATION	POS_{SVM}	73.0%	75.3	68.5	71.7	71.1	77.5	74.2
PSYCHOLINGUISTIC DECEPTION DETECTION	LIWC_{SVM}	76.8%	77.2	76.0	76.6	76.4	77.5	76.9
TEXT CATEGORIZATION	$\text{UNIGRAMS}_{\text{SVM}}$	88.4%	89.9	86.5	88.2	87.0	90.3	88.6
	$\text{BIGRAMS}^{+}_{\text{SVM}}$	89.6%	90.1	89.0	89.6	89.1	90.3	89.7
	$\text{LIWC+BIGRAMS}^{+}_{\text{SVM}}$	**89.8%**	89.8	**89.8**	**89.8**	**89.8**	89.8	**89.8**
	$\text{TRIGRAMS}^{+}_{\text{SVM}}$	89.0%	89.0	89.0	89.0	89.0	89.0	89.0
	$\text{UNIGRAMS}_{\text{NB}}$	88.4%	**92.5**	83.5	87.8	85.0	**93.3**	88.9
	$\text{BIGRAMS}^{+}_{\text{NB}}$	88.9%	89.8	87.8	88.7	88.0	90.0	89.0
	$\text{TRIGRAMS}^{+}_{\text{NB}}$	87.6%	87.7	87.5	87.6	87.5	87.8	87.6
HUMAN / META	JUDGE 1	**61.9%**	57.9	87.5	**69.7**	74.4	36.3	48.7
	JUDGE 2	56.9%	53.9	**95.0**	68.8	**78.9**	18.8	30.3
	SKEPTIC	60.6%	**60.8**	60.0	60.4	60.5	**61.3**	**60.9**

Table 3: Automated classifier performance for three approaches based on nested 5-fold cross-validation experiments. Reported precision, recall and F-score are computed using a micro-average, i.e., from the *aggregate* true positive, false positive and false negative rates, as suggested by Forman and Scholz (2009). Human performance is repeated here for JUDGE 1, JUDGE 2 and the SKEPTIC meta-judge, although they cannot be directly compared since the 160-opinion subset on which they are assessed only corresponds to the first cross-validation fold.

5 Results and Discussion

The deception detection strategies described in Section 4 are evaluated using a 5-fold *nested* cross-validation (CV) procedure (Quadrianto et al., 2009), where model parameters are selected for each test fold based on *standard* CV experiments on the training folds. Folds are selected so that each contains *all* reviews from four hotels; thus, learned models are always evaluated on reviews from unseen hotels.

Results appear in Table 3. We observe that automated classifiers outperform human judges for every metric, except truthful recall where JUDGE 2 performs best.[16] However, this is expected given that untrained humans often focus on unreliable cues to deception (Vrij, 2008). For example, one study examining deception in online dating found that humans perform at-chance detecting deceptive profiles because they rely on text-based cues that are unrelated to deception, such as second-person pronouns (Toma and Hancock, In Press).

Among the automated classifiers, baseline performance is given by the simple genre identification approach (POS_{SVM}) proposed in Section 4.1. Surprisingly, we find that even this simple automated classifier outperforms most human judges (one-tailed sign test $p = 0.06, 0.01, 0.001$ for the three judges, respectively, on the first fold). This result is best explained by theories of reality monitoring (Johnson and Raye, 1981), which suggest that truthful and deceptive opinions might be classified into informative and imaginative genres, respectively. Work by Rayson et al. (2001) has found strong distributional differences between informative and imaginative writing, namely that the former typically consists of more nouns, adjectives, prepositions, determiners, and coordinating conjunctions, while the latter consists of more verbs,[17] adverbs,[18] pronouns, and pre-determiners. Indeed, we find that the weights learned by POS_{SVM} (found in Table 4) are largely in agreement with these findings, notably except for adjective and adverb *superlatives*, the latter of which was found to be an exception by Rayson et al. (2001). However, that deceptive opinions contain more superlatives is not unexpected, since deceptive writing (but not necessarily imaginative writing in general) often contains exaggerated language (Buller and Burgoon, 1996; Hancock et al., 2008).

Both remaining automated approaches to detecting deceptive opinion spam outperform the simple

[16] As mentioned in Section 3.3, JUDGE 2 classified fewer than 12% of opinions as deceptive. While achieving 95% truthful recall, this judge's corresponding precision was not significantly better than chance (two-tailed binomial $p = 0.4$).

[17] *Past participle* verbs were an exception.

[18] *Superlative* adverbs were an exception.

TRUTHFUL/INFORMATIVE			DECEPTIVE/IMAGINATIVE		
Category	**Variant**	**Weight**	**Category**	**Variant**	**Weight**
NOUNS	Singular	0.008	VERBS	Base	-0.057
	Plural	0.002		Past tense	**0.041**
	Proper, singular	**-0.041**		Present participle	-0.089
	Proper, plural	0.091		Singular, present	-0.031
ADJECTIVES	General	0.002		Third person singular, present	**0.026**
	Comparative	0.058			
	Superlative	**-0.164**		Modal	-0.063
PREPOSITIONS	General	0.064	ADVERBS	General	**0.001**
DETERMINERS	General	0.009		Comparative	-0.035
COORD. CONJ.	General	0.094	PRONOUNS	Personal	-0.098
VERBS	Past participle	0.053		Possessive	-0.303
ADVERBS	Superlative	**-0.094**	PRE-DETERMINERS	General	**0.017**

Table 4: Average feature weights learned by POS_{SVM}. Based on work by Rayson et al. (2001), we expect weights on the left to be positive (predictive of *truthful* opinions), and weights on the right to be negative (predictive of *deceptive* opinions). Boldface entries are at odds with these expectations. We report average feature weights of *unit-normalized* weight vectors, rather than *raw* weights vectors, to account for potential differences in magnitude between the folds.

genre identification baseline just discussed. Specifically, the psycholinguistic approach (LIWC_{SVM}) proposed in Section 4.2 performs 3.8% more accurately (one-tailed sign test $p = 0.02$), and the standard text categorization approach proposed in Section 4.3 performs between 14.6% and 16.6% more accurately. However, best performance overall is achieved by combining features from these two approaches. Particularly, the combined model $\text{LIWC+BIGRAMS}^{+}_{\text{SVM}}$ is 89.8% accurate at detecting deceptive opinion spam.[19]

Surprisingly, models trained only on UNIGRAMS—the simplest n-gram feature set—outperform all non–text-categorization approaches, and models trained on BIGRAMS^{+} perform *even better* (one-tailed sign test $p = 0.07$). This suggests that a universal set of keyword-based deception cues (e.g., LIWC) is not the best approach to detecting deception, and a context-sensitive approach (e.g., BIGRAMS^{+}) might be necessary to achieve state-of-the-art deception detection performance.

To better understand the models learned by these automated approaches, we report in Table 5 the top 15 highest weighted features for each class (*truthful* and *deceptive*) as learned by $\text{LIWC+BIGRAMS}^{+}_{\text{SVM}}$ and LIWC_{SVM}. In agreement with theories of reality monitoring (Johnson and Raye, 1981), we observe that truthful opinions tend to include more sensorial and concrete language than deceptive opinions; in particular, truthful opinions are more specific about spatial configurations (e.g., small, bathroom, on, location). This finding is also supported by recent work by Vrij et al. (2009) suggesting that liars have considerable difficultly encoding spatial information into their lies. Accordingly, we observe an increased focus in deceptive opinions on aspects external to the hotel being reviewed (e.g., husband, business,

[19]The result is not significantly better than $\text{BIGRAMS}^{+}_{\text{SVM}}$.

$\text{LIWC+BIGRAMS}^{+}_{\text{SVM}}$		LIWC_{SVM}	
TRUTHFUL	DECEPTIVE	TRUTHFUL	DECEPTIVE
-	chicago	hear	i
...	my	number	family
on	hotel	allpunct	perspron
location	,_and	negemo	see
)	luxury	dash	pronoun
$\text{allpunct}_{\text{LIWC}}$	experience	exclusive	leisure
floor	hilton	we	exclampunct
(	business	sexual	sixletters
the_hotel	vacation	period	posemo
bathroom	i	otherpunct	comma
small	spa	space	cause
helpful	looking	human	auxverb
$	while	past	future
hotel_.	husband	inhibition	perceptual
other	my_husband	assent	feel

Table 5: Top 15 highest weighted truthful and deceptive features learned by $\text{LIWC+BIGRAMS}^{+}_{\text{SVM}}$ and LIWC_{SVM}. Ambiguous features are subscripted to indicate the source of the feature. LIWC features correspond to groups of keywords as explained in Section 4.2; more details about LIWC and the LIWC categories are available at `http://liwc.net`.

vacation).

We also acknowledge several findings that, on the surface, are in contrast to previous psycholinguistic studies of deception (Hancock et al., 2008; Newman et al., 2003). For instance, while deception is often associated with negative emotion terms, our deceptive reviews have more positive and fewer negative emotion terms. This pattern makes sense when one considers the goal of our deceivers, namely to create a positive review (Buller and Burgoon, 1996).

Deception has also previously been associated with decreased usage of first person singular, an effect attributed to psychological distancing (Newman et al., 2003). In contrast, we find increased first person singular to be among the largest indicators of deception, which we speculate is due to our deceivers attempting to enhance the credibility of their reviews by emphasizing their own presence in the review. Additional work is required, but these findings further suggest the importance of moving beyond a universal set of deceptive language features (e.g., LIWC) by considering both the contextual (e.g., BIGRAMS$^+$) and motivational parameters underlying a deception as well.

6 Conclusion and Future Work

In this work we have developed the first large-scale dataset containing *gold-standard* deceptive opinion spam. With it, we have shown that the detection of deceptive opinion spam is well beyond the capabilities of human judges, most of whom perform roughly at-chance. Accordingly, we have introduced three *automated* approaches to deceptive opinion spam detection, based on insights coming from research in computational linguistics and psychology. We find that while standard n-gram–based text categorization is the best individual detection approach, a *combination* approach using psycholinguistically-motivated features and n-gram features can perform slightly better.

Finally, we have made several theoretical contributions. Specifically, our findings suggest the importance of considering both the context (e.g., BIGRAMS$^+$) and motivations underlying a deception, rather than strictly adhering to a universal set of deception cues (e.g., LIWC). We have also presented results based on the feature weights learned by our classifiers that illustrate the difficulties faced by liars in encoding spatial information. Lastly, we have discovered a plausible relationship between deceptive opinion spam and imaginative writing, based on POS distributional similarities.

Possible directions for future work include an extended evaluation of the methods proposed in this work to both negative opinions, as well as opinions coming from other domains. Many additional approaches to detecting deceptive opinion spam are also possible, and a focus on approaches with high deceptive precision might be useful for production environments.

Acknowledgments

This work was supported in part by National Science Foundation Grants BCS-0624277, BCS-0904822, HSD-0624267, IIS-0968450, and NSCC-0904822, as well as a gift from Google, and the Jack Kent Cooke Foundation. We also thank, alphabetically, Rachel Boochever, Cristian Danescu-Niculescu-Mizil, Alicia Granstein, Ulrike Gretzel, Danielle Kirshenblat, Lillian Lee, Bin Lu, Jack Newton, Melissa Sackler, Mark Thomas, and Angie Yoo, as well as members of the Cornell NLP seminar group and the ACL reviewers for their insightful comments, suggestions and advice on various aspects of this work.

References

C. Akkaya, A. Conrad, J. Wiebe, and R. Mihalcea. 2010. Amazon mechanical turk for subjectivity word sense disambiguation. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazons Mechanical Turk, Los Angeles*, pages 195–203.

D. Biber, S. Johansson, G. Leech, S. Conrad, E. Finegan, and R. Quirk. 1999. *Longman grammar of spoken and written English*, volume 2. MIT Press.

C.F. Bond and B.M. DePaulo. 2006. Accuracy of deception judgments. *Personality and Social Psychology Review*, 10(3):214.

D.B. Buller and J.K. Burgoon. 1996. Interpersonal deception theory. *Communication Theory*, 6(3):203–242.

W.L. Cade, B.A. Lehman, and A. Olney. 2010. An exploration of off topic conversation. In *Human Language Technologies: The 2010 Annual Conference of*

the North American Chapter of the Association for Computational Linguistics, pages 669–672. Association for Computational Linguistics.

S.F. Chen and J. Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics.

C. Danescu-Niculescu-Mizil, G. Kossinets, J. Kleinberg, and L. Lee. 2009. How opinions are received by online communities: a case study on amazon.com helpfulness votes. In *Proceedings of the 18th international conference on World wide web*, pages 141–150. ACM.

H. Drucker, D. Wu, and V.N. Vapnik. 2002. Support vector machines for spam categorization. *Neural Networks, IEEE Transactions on*, 10(5):1048–1054.

G. Forman and M. Scholz. 2009. Apples-to-Apples in Cross-Validation Studies: Pitfalls in Classifier Performance Measurement. *ACM SIGKDD Explorations*, 12(1):49–57.

Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. 2004. Combating web spam with trustrank. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 576–587. VLDB Endowment.

J.T. Hancock, L.E. Curry, S. Goorha, and M. Woodworth. 2008. On lying and being lied to: A linguistic analysis of deception in computer-mediated communication. *Discourse Processes*, 45(1):1–23.

J. Jansen. 2010. Online product research. *Pew Internet & American Life Project Report*.

N. Jindal and B. Liu. 2008. Opinion spam and analysis. In *Proceedings of the international conference on Web search and web data mining*, pages 219–230. ACM.

T. Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. *Machine Learning: ECML-98*, pages 137–142.

T. Joachims. 1999. Making large-scale support vector machine learning practical. In *Advances in kernel methods*, page 184. MIT Press.

M.K. Johnson and C.L. Raye. 1981. Reality monitoring. *Psychological Review*, 88(1):67–85.

S.M. Kim, P. Pantel, T. Chklovski, and M. Pennacchiotti. 2006. Automatically assessing review helpfulness. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 423–430. Association for Computational Linguistics.

D. Klein and C.D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

J.R. Landis and G.G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159.

E.P. Lim, V.A. Nguyen, N. Jindal, B. Liu, and H.W. Lauw. 2010. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 939–948. ACM.

S.W. Litvin, R.E. Goldsmith, and B. Pan. 2008. Electronic word-of-mouth in hospitality and tourism management. *Tourism management*, 29(3):458–468.

F. Mairesse, M.A. Walker, M.R. Mehl, and R.K. Moore. 2007. Using linguistic cues for the automatic recognition of personality in conversation and text. *Journal of Artificial Intelligence Research*, 30(1):457–500.

R. Mihalcea and C. Strapparava. 2009. The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 309–312. Association for Computational Linguistics.

M.L. Newman, J.W. Pennebaker, D.S. Berry, and J.M. Richards. 2003. Lying words: Predicting deception from linguistic styles. *Personality and Social Psychology Bulletin*, 29(5):665.

A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. 2006. Detecting spam web pages through content analysis. In *Proceedings of the 15th international conference on World Wide Web*, pages 83–92. ACM.

M.P. O'Mahony and B. Smyth. 2009. Learning to recommend helpful hotel reviews. In *Proceedings of the third ACM conference on Recommender systems*, pages 305–308. ACM.

F. Peng and D. Schuurmans. 2003. Combining naive Bayes and n-gram language models for text classification. *Advances in Information Retrieval*, pages 547–547.

J.W. Pennebaker, C.K. Chung, M. Ireland, A. Gonzales, and R.J. Booth. 2007. The development and psychometric properties of LIWC2007. *Austin, TX, LIWC. Net*.

N. Quadrianto, A.J. Smola, T.S. Caetano, and Q.V. Le. 2009. Estimating labels from label proportions. *The Journal of Machine Learning Research*, 10:2349–2374.

P. Rayson, A. Wilson, and G. Leech. 2001. Grammatical word class variation within the British National Corpus sampler. *Language and Computers*, 36(1):295–306.

R.A. Rigby and D.M. Stasinopoulos. 2005. Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 54(3):507–554.

F. Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

M.Á. Serrano, A. Flammini, and F. Menczer. 2009. Modeling statistical properties of written text. *PloS one*, 4(4):5372.

A. Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *Seventh International Conference on Spoken Language Processing*, volume 3, pages 901–904. Citeseer.

C. Toma and J.T. Hancock. In Press. What Lies Beneath: The Linguistic Traces of Deception in Online Dating Profiles. *Journal of Communication*.

A. Vrij, S. Mann, S. Kristen, and R.P. Fisher. 2007. Cues to deception and ability to detect lies as a function of police interview styles. *Law and human behavior*, 31(5):499–518.

A. Vrij, S. Leal, P.A. Granhag, S. Mann, R.P. Fisher, J. Hillman, and K. Sperry. 2009. Outsmarting the liars: The benefit of asking unanticipated questions. *Law and human behavior*, 33(2):159–166.

A. Vrij. 2008. *Detecting lies and deceit: Pitfalls and opportunities*. Wiley-Interscience.

W. Weerkamp and M. De Rijke. 2008. Credibility improves topical blog post retrieval. *ACL-08: HLT*, pages 923–931.

M. Weimer, I. Gurevych, and M. Mühlhäuser. 2007. Automatically assessing the post quality in online discussions on software. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 125–128. Association for Computational Linguistics.

G. Wu, D. Greene, B. Smyth, and P. Cunningham. 2010. Distortion as a validation criterion in the identification of suspicious reviews. Technical report, UCD-CSI-2010-04, University College Dublin.

K.H. Yoo and U. Gretzel. 2009. Comparison of Deceptive and Truthful Travel Reviews. *Information and Communication Technologies in Tourism 2009*, pages 37–47.

L. Zhou, J.K. Burgoon, D.P. Twitchell, T. Qin, and J.F. Nunamaker Jr. 2004. A comparison of classification methods for predicting deception in computer-mediated communication. *Journal of Management Information Systems*, 20(4):139–166.

L. Zhou, Y. Shi, and D. Zhang. 2008. A Statistical Language Modeling Approach to Online Deception Detection. *IEEE Transactions on Knowledge and Data Engineering*, 20(8):1077–1081.

Joint Bilingual Sentiment Classification with Unlabeled Parallel Corpora

Bin Lu[1,3*], **Chenhao Tan**[2], **Claire Cardie**[2] **and Benjamin K. Tsou**[3,1]

[1] Department of Chinese, Translation and Linguistics, City University of Hong Kong, Hong Kong

[2] Department of Computer Science, Cornell University, Ithaca, NY, USA

[3] Research Centre on Linguistics and Language Information Sciences,
Hong Kong Institute of Education, Hong Kong

lubin2010@gmail.com, {chenhao, cardie}@cs.cornell.edu, btsou99@gmail.com

Abstract

Most previous work on multilingual sentiment analysis has focused on methods to adapt sentiment resources from resource-rich languages to resource-poor languages. We present a novel approach for joint bilingual sentiment classification at the sentence level that augments available labeled data in each language with unlabeled parallel data. We rely on the intuition that the sentiment labels for parallel sentences should be similar and present a model that jointly learns improved monolingual sentiment classifiers for each language. Experiments on multiple data sets show that the proposed approach (1) outperforms the monolingual baselines, significantly improving the accuracy for both languages by 3.44%-8.12%; (2) outperforms two standard approaches for leveraging unlabeled data; and (3) produces (albeit smaller) performance gains when employing pseudo-parallel data from machine translation engines.

1 Introduction

The field of sentiment analysis has quickly attracted the attention of researchers and practitioners alike (e.g. Pang et al., 2002; Turney, 2002; Hu and Liu, 2004; Wiebe et al., 2005; Breck et al., 2007; Pang and Lee, 2008). Indeed, sentiment analysis systems, which mine opinions from textual sources (e.g. news, blogs, and reviews), can be used in a wide variety of applications, including interpreting product reviews, opinion retrieval and political polling.

Not surprisingly, most methods for sentiment classification are supervised learning techniques, which require training data annotated with the appropriate sentiment labels (e.g. document-level or sentence-level positive vs. negative polarity). This data is difficult and costly to obtain, and must be acquired separately for each language under consideration.

Previous work in multilingual sentiment analysis has therefore focused on methods to adapt sentiment resources (e.g. lexicons) from resource-rich languages (typically English) to other languages, with the goal of transferring sentiment or subjectivity analysis capabilities from English to other languages (e.g. Mihalcea et al. (2007); Banea et al. (2008; 2010); Wan (2008; 2009); Prettenhofer and Stein (2010)). In recent years, however, sentiment-labeled data is gradually becoming available for languages other than English (e.g. Seki et al. (2007; 2008); Nakagawa et al. (2010); Schulz et al. (2010)). In addition, there is still much room for improvement in existing monolingual (including English) sentiment classifiers, especially at the sentence level (Pang and Lee, 2008).

This paper tackles the task of bilingual sentiment analysis. In contrast to previous work, we (1) assume that some amount of sentiment-labeled data is available for the language pair under study, and (2) investigate methods to simultaneously improve sentiment classification for *both languages*. Given the labeled data in each language, we propose an approach that exploits an *unlabeled* parallel corpus with the following

[*] The work was conducted when the first author was visiting Cornell University.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 320–330,
Portland, Oregon, June 19-24, 2011.

intuition: *two sentences or documents that are parallel (i.e. translations of one another) should exhibit the same sentiment — their sentiment labels (e.g. polarity, subjectivity, intensity) should be similar*. The proposed maximum entropy-based EM approach jointly learns two monolingual sentiment classifiers by treating the sentiment labels in the unlabeled parallel text as unobserved latent variables, and maximizes the regularized joint likelihood of the language-specific labeled data together with the inferred sentiment labels of the parallel text. Although our approach should be applicable at the document-level and for additional sentiment tasks, we focus on sentence-level polarity classification in this work.

We evaluate our approach for English and Chinese on two dataset combinations (see Section 4) and find that the proposed approach outperforms the monolingual baselines (i.e. maximum entropy and SVM classifiers) as well as two alternative methods for leveraging unlabeled data (transductive SVMs (Joachims, 1999b) and co-training (Blum and Mitchell, 1998)). Accuracy is significantly improved for both languages, by 3.44%-8.12%. We furthermore find that improvements, albeit smaller, are obtained when the parallel data is replaced with a pseudo-parallel (i.e. automatically translated) corpus. To our knowledge, this is the first multilingual sentiment analysis study to focus on methods for simultaneously improving sentiment classification for a pair of languages based on unlabeled data rather than resource adaptation from one language to another.

The rest of the paper is organized as follows. Section 2 introduces related work. In Section 3, the proposed joint model is described. Sections 4 and 5, respectively, provide the experimental setup and results; the conclusion (Section 6) follows.

2 Related Work

Multilingual Sentiment Analysis. There is a growing body of work on multilingual sentiment analysis. Most approaches focus on resource adaptation from one language (usually English) to other languages with few sentiment resources. Mihalcea et al. (2007), for example, generate subjectivity analysis resources in a new language from English sentiment resources by leveraging a bilingual dictionary or a parallel corpus. Banea et al. (2008; 2010) instead automatically translate the English resources using automatic machine translation engines for subjectivity classification. Prettenhofer and Stein (2010) investigate cross-lingual sentiment classification from the perspective of domain adaptation based on structural correspondence learning (Blitzer et al., 2006).

Approaches that do not explicitly involve resource adaptation include Wan (2009), which uses co-training (Blum and Mitchell, 1998) with English vs. Chinese features comprising the two independent "views" to exploit unlabeled Chinese data and a labeled English corpus and thereby improves Chinese sentiment classification. Another notable approach is the work of Boyd-Graber and Resnik (2010), which presents a generative model --- supervised multilingual latent Dirichlet allocation --- that jointly models topics that are consistent across languages, and employs them to better predict sentiment ratings.

Unlike the methods described above, we focus on simultaneously improving the performance of sentiment classification in a pair of languages by developing a model that relies on sentiment-labeled data in each language as well as unlabeled parallel text for the language pair.

Semi-supervised Learning. Another line of related work is semi-supervised learning, which combines labeled and unlabeled data to improve the performance of the task of interest (Zhu and Goldberg, 2009). Among the popular semi-supervised methods (e.g. EM on Naïve Bayes (Nigam et al., 2000), co-training (Blum and Mitchell, 1998), transductive SVMs (Joachims, 1999b), and co-regularization (Sindhwani et al., 2005; Amini et al., 2010)), our approach employs the EM algorithm, extending it to the bilingual case based on maximum entropy. We compare to co-training and transductive SVMs in Section 5.

Multilingual NLP for Other Tasks. Finally, there exists related work using bilingual resources to help other NLP tasks, such as word sense disambiguation (e.g. Ido and Itai (1994)), parsing (e.g. Burkett and Klein (2008); Zhao et al. (2009); Burkett et al. (2010)), information retrieval (Gao et al., 2009), named entity detection (Burkett et al., 2010); topic extraction (e.g. Zhang et al., 2010), text classification (e.g. Amini et al., 2010), and hyponym-relation acquisition (e.g. Oh et al., 2009).

In these cases, multilingual models increase performance because different languages contain different ambiguities and therefore present complementary views on the shared underlying labels. Our work shares a similar motivation.

3 A Joint Model with Unlabeled Parallel Text

We propose a maximum entropy-based statistical model. Maximum entropy (MaxEnt) models[1] have been widely used in many NLP tasks (Berger et al., 1996; Ratnaparkhi, 1997; Smith, 2006). The models assign the conditional probability of the label y given the observation x as follows:

$$p(y|x;\vec{\theta}) = \frac{1}{Z}\exp\left(\vec{\theta}\cdot\vec{f}(x,y)\right) \quad (1)$$

where $\vec{\theta}$ is a real-valued vector of feature weights and $\vec{f}$ is a feature function that maps pairs (x, y) to a nonnegative real-valued feature vector. Each feature has an associated parameter, θ_i, which is called its weight; and Z is the corresponding normalization factor.

Maximum likelihood parameter estimation (training) for such a model, with a set of labeled examples $\{(x_i, y_i)_{i=1}^n\}$, amounts to solving the following optimization problem:

$$\vec{\theta}^* = \arg\max_{\vec{\theta}} \prod_{i=1}^n p(y_i|x_i;\vec{\theta}) \quad (2)$$

3.1 Problem Definition

Given two languages L_1 and L_2, suppose we have two distinct (i.e. not parallel) sets of sentiment-labeled data, D_1 and D_2, written in L_1 and L_2, respectively. In addition, we have unlabeled (w.r.t. sentiment) bilingual (in L_1 and L_2) parallel data U that are defined as follows.

$$D_1 = (X_1, Y_1) = \{(x_i^1, \mathrm{y_i^1})_{i=1}^{l_1}\}$$
$$D_2 = (X_2, Y_2) = \{(x_i^2, \mathrm{y_i^2})_{i=1}^{l_2}\}$$
$$U = (X_1', X_2') = \left\{\left(x_i^{1'}, x_i^{2'}\right)_{i=1}^{u}\right\}$$

where $y_i \in \mathcal{Y} = \{+1, -1\}$ denotes the polarity of the i-th instance x_i (positive or negative); l_1 and l_2 are respectively the numbers of labeled instances in L_1 and L_2; $x_i^{1'}$ and $x_i^{2'}$ are parallel instances in L_1 and L_2, respectively (i.e. they are supposed to be translations of one another), whose labels $y_i^{1'}$ and $y_i^{2'}$ are unobserved, but according to the intuition outlined in Section 1, should be similar.

Given the input data D_1, D_2 and U, our task is to jointly learn two monolingual sentiment classifiers — one for L_1 and one for L_2. With MaxEnt, we learn from the input data:

$$f\colon \{D_1, D_2, U\} \rightarrow (\vec{\theta}_1^*, \vec{\theta}_2^*)$$

where $\vec{\theta}_1^*$ and $\vec{\theta}_2^*$ are the vectors of feature weights for L_1 and L_2, respectively (for brevity we denote them as θ_1 and θ_2 in the remaining sections). In this study, we focus on sentence-level sentiment classification, i.e. each x_i is a sentence, and $x_i^{1'}$ and $x_i^{2'}$ are parallel sentences.

3.2 The Joint Model

Given the problem definition above, we now present a novel model to exploit the correspondence of parallel sentences in unlabeled bilingual text. The model maximizes the following joint likelihood with respect to θ_1 and θ_2:

$$\begin{aligned}\mathcal{L}(\theta_1, \theta_2|D_1, D_2, U) &= p(Y_1|X_1;\theta_1)p(Y_2|X_2;\theta_2)\\ &\quad p(Y_1', Y_2'|X_1', X_2';\theta_1, \theta_2)\\ &= \prod_{v=1}^{2}\prod_{i=1}^{l_v} p(y_i^v|x_i^v;\theta_v)\\ &\quad \prod_{i=1}^{u} p\left(y_i^{1'}, y_i^{2'}|x_i^{1'}, x_i^{2'};\theta_1, \theta_2\right)\end{aligned} \quad (3)$$

where $v \in \{1,2\}$ denotes L_1 or L_2; the first term on the right-hand side is the likelihood of labeled data for both D_1 and D_2; and the second term is the likelihood of the unlabeled parallel data U.

If we assume that parallel sentences are perfect translations, the two sentences in each pair should have the same polarity label, which gives us:

$$\begin{aligned}&p\left(y_i^{1'}, y_i^{2'}|x_i^{1'}, x_i^{2'};\theta_1, \theta_2\right) =\\ &\textstyle\sum_{y_i'} p\left(y_i'|x_i^{1'};\theta_1\right)p\left(y_i'|x_i^{2'};\theta_2\right)\end{aligned} \quad (4)$$

where y_i' is the unobserved class label for the i-th instance in the unlabeled data. This probability directly models the sentiment label agreement between $x_i^{1'}$ and $x_i^{2'}$.

However, there could be considerable noise in real-world parallel data, i.e. the sentence pairs may be noisily parallel (or even comparable) instead of fully parallel (Munteanu and Marcu, 2005). In such noisy cases, the labels (positive or negative) could be different for the two monolingual sentences in a sentence pair. Although we do not know the exact probability that a sentence pair exhibits the same label, we can approximate it using their translation

[1]They are sometimes referred to as log-linear models, but also known as exponential models, generalized linear models, or logistic regression.

probabilities, which can be computed using word alignment toolkits such as Giza++ (Och and Ney, 2003) or the Berkeley word aligner (Liang et al., 2006). *The intuition here is that if the translation probability of two sentences is high, the probability that they have the same sentiment label should be high as well*. Therefore, by considering the noise in parallel data, we get:

$$p(y_i^{1'}, y_i^{2'} | x_i^{1'}, x_i^{2'}; \theta_1, \theta_2) = \sum_{y'}\{p(a_i)p(y'|x_i^{1'};\theta_1)p(y'|x_i^{2'};\theta_2)\} + \sum_{y'}\{(1-p(a_i))p(y'|x_i^{1'};\theta_1)p(\bar{y'}|x_i^{2'};\theta_2)\} \quad (5)$$

where $p(a_i)$ is the translation probability of the i-th sentence pair in U;[2] $\bar{y'}$ is the opposite of y'; the first term models the probability that $x_i^{1'}$ and $x_i^{2'}$ have the same label; and the second term models the probability that they have different labels.

By further considering the weight to ascribe to the unlabeled data vs. the labeled data (and the weight for the L2-norm regularization), we get the following regularized joint log likelihood to be maximized:

$$\log \mathcal{L}(\theta_1, \theta_2 | D_1, D_2, U) = \sum_{v=1}^{2} \log p(Y_v | X_v; \theta_v) + \lambda_1 \log p(Y_1', Y_2' | X_1', X_2'; \theta_1, \theta_2) - \frac{\lambda_2}{2} \sum_{v=1}^{2} \| \theta_v \|_2^2 \quad (6)$$

where the first term on the right-hand side is the log likelihood of the labeled data from both D_1 and D_2; the second is the log likelihood of the unlabeled parallel data U, multiplied by $\lambda_1 \geq 0$, a constant that controls the contribution of the unlabeled data; and $\lambda_2 \geq 0$ is a regularization constant that penalizes model complexity or large feature weights. When λ_1 is 0, the algorithm ignores the unlabeled data and degenerates to two MaxEnt models trained on only the labeled data.

3.3 The EM Algorithm on MaxEnt

To solve the optimization problem for the model, we need to jointly estimate the optimal parameters for the two monolingual classifiers by finding:

$$(\theta_1^*, \theta_2^*) = \arg\max_{(\theta_1, \theta_2)} \log \mathcal{L}(\theta_1, \theta_2 | D_1, D_2, U) \quad (7)$$

This can be done with an EM algorithm, whose steps are summarized in Algorithm 1. First, the MaxEnt parameters, θ_1 and θ_2, are estimated from just the labeled data. Then, in the E-step, the classifiers, based on current values of θ_1 and θ_2, compute $p(y_i|x_i)$ for each labeled example and assign probabilistically-weighted class labels to each unlabeled example. Next, in the M-step, the parameters, θ_1 and θ_2, are updated using both the original labeled data (D_1 and D_2) and the newly labeled data U. These last two steps are iterated until convergence or a predefined iteration limit T.

Algorithm 1. The MaxEnt-based EM Algorithm for Multilingual Sentiment Classification

Input: Labeled data D_1 and D_2
Unlabeled parallel data U

Output: Two monolingual MaxEnt classifiers with parameters θ_1^* and θ_2^*, respectively

1. **Train two initial monolingual models**
   Train and initialize $\theta_1^{(0)}$ and $\theta_2^{(0)}$ on the labeled data
2. **Jointly optimize two monolingual models**
   for $t = 1$ **to** T **do** // T: number of iterations
   E-Step:
   Compute $p(y|x)$ for each example in D_1, D_2 and U based on $\theta_1^{(t-1)}$ and $\theta_2^{(t-1)}$;
   Compute the expectation of the log likelihood with respect to $p(y|x)$;
   M-Step:
   Find $\theta_1^{(t)}$ and $\theta_2^{(t)}$ by maximizing the regularized joint log likelihood;
   Convergence:
   If the increase of the joint log likelihood is sufficiently small, break;
   end for
3. Output θ_1^* as $\theta_1^{(t)}$s, and θ_2^* as $\theta_2^{(t)}$

In the M-step, we can optimize the regularized joint log likelihood using any gradient-based optimization technique (Malouf, 2002). The gradient for Equation 3 based on Equation 4 is shown in Appendix A; those for Equations 5 and 6 can be derived similarly. In our experiments, we use the L-BFGS algorithm (Liu et al., 1989) and run EM until the change in regularized joint log likelihood is less than 1e-5 or we reach 100 iterations.[3]

[2]The probability should be rescaled within the range of [0, 1], where 0.5 means that we are completely unsure if the sentences are translations of each other or not, and only those translation pairs with a probability larger than 0.5 are meaningful for our purpose.

[3]Since the EM-based algorithm may find a local maximum of the objective function, the initialization of the parameters is important. Our experiments show that an effective maximum can usually be found by initializing the parameters with those learned from the labeled data; performance would be much worse if we initialize all the parameters to 0 or 1.

3.4 Pseudo-Parallel Labeled and Unlabeled Data

We also consider the case where a parallel corpus is not available: to obtain a pseudo-parallel corpus U (i.e. sentences in one language with their corresponding automatic translations), we use an automatic machine translation system (e.g. Google machine translation[4]) to translate unlabeled in-domain data from L_1 to L_2 or vice versa.

Since previous work (Banea et al., 2008; 2010; Wan, 2009) has shown that it could be useful to automatically translate the labeled data from the source language into the target language, we can further incorporate such translated labeled data into the joint model by adding the following component into Equation 6:

$$\lambda_3 \sum_{v=1}^{2} \sum_{i=1}^{l_{\bar{v}}} \log p\left(y_i^{\bar{v}} \middle| x_i^{\bar{v}^*}; \theta_v\right) \quad (8)$$

where $\bar{v}$ is the alternative class of v, $x_i^{\bar{v}^*}$ is the automatically translated example from $x_i^{\bar{v}}$; and $\lambda_3 \geq 0$ is a constant that controls the weight of the translated labeled data.

4 Experimental Setup

4.1 Data Sets and Preprocessing

The following labeled datasets are used in our experiments.

MPQA (Labeled English Data): The Multi-Perspective Question Answering (MPQA) corpus (Wiebe et al., 2005) consists of newswire documents manually annotated with phrase-level subjectivity information. We extract all sentences containing strong (i.e. intensity is *medium* or higher), sentiment-bearing (i.e. polarity is *positive* or *negative*) expressions following Choi and Cardie (2008). Sentences with both positive and negative strong expressions are then discarded, and the polarity of each remaining sentence is set to that of its sentiment-bearing expression(s).

NTCIR-EN (Labeled English Data) and NTCIR-CH (Labeled Chinese Data): The NTCIR Opinion Analysis task (Seki et al., 2007; 2008) provides sentiment-labeled news data in Chinese, Japanese and English. Only those sentences with a polarity label (positive or negative) agreed to by at least two annotators are extracted. We use the Chinese data from NTCIR-6 as our Chinese labeled data. Since far fewer sentences in the English data pass the annotator agreement filter, we combine the English data from NTCIR-6 and NTCIR-7. The Chinese sentences are segmented using the Stanford Chinese word segmenter (Tseng et al., 2005).

The number of sentences in each of these datasets is shown in Table 1. In our experiments, we evaluate two settings of the data: (1) MPQA+NTCIR-CH, and (2) NTCIR-EN+NTCIR-CH. In each setting, the English labeled data constitutes D_1 and the Chinese labeled data, D_2.

	MPQA	NTCIR-EN	NTCIR-CH
Positive	1,471 (30%)	528 (30%)	2,378 (55%)
Negative	3,487 (70%)	1,209 (70%)	1,916 (45%)
Total	4,958	1,737	4,294

Table 1: Sentence Counts for the Labeled Data

Unlabeled Parallel Text and its Preprocessing. For the unlabeled parallel text, we use the ISI Chinese-English parallel corpus (Munteanu and Marcu, 2005), which was extracted automatically from news articles published by Xinhua News Agency in the Chinese Gigaword (2nd Edition) and English Gigaword (2nd Edition) collections. Because sentence pairs in the ISI corpus are quite noisy, we rely on Giza++ (Och and Ney, 2003) to obtain a new translation probability for each sentence pair, and select the 100,000 pairs with the highest translation probabilities.[5]

We also try to remove neutral sentences from the parallel data since they can introduce noise into our model, which deals only with positive and negative examples. To do this, we train a single classifier from the combined Chinese and English labeled data for each data setting above by concatenating the original English and Chinese feature sets. We then classify each unlabeled sentence pair by combining the two sentences in each pair into one. We choose the most confidently predicted 10,000 positive and 10,000 negative pairs to constitute the unlabeled parallel corpus U for each data setting.

[4]http://translate.google.com/

[5]We removed sentence pairs with an original confidence score (given in the corpus) smaller than 0.98, and also removed the pairs that are too long (more than 60 characters in one sentence) to facilitate Giza++. We first obtain translation probabilities for both directions (i.e. Chinese to English and English to Chinese) with Giza++, take the log of the product of those two probabilities, and then divide it by the sum of lengths of the two sentences in each pair.

4.2 Baseline Methods

In our experiments, the proposed joint model is compared with the following baseline methods.

MaxEnt: This method learns a MaxEnt classifier for each language given the monolingual labeled data; the unlabeled data is not used.

SVM: This method learns an SVM classifier for each language given the monolingual labeled data; the unlabeled data is not used. SVM-light (Joachims, 1999a) is used for all the SVM-related experiments.

Monolingual TSVM (TSVM-M): This method learns two transductive SVM (TSVM) classifiers given the monolingual labeled data and the monolingual unlabeled data for each language.

Bilingual TSVM (TSVM-B): This method learns one TSVM classifier given the labeled training data in two languages together with the unlabeled sentences by combining the two sentences in each unlabeled pair into one. We expect this method to perform better than TSVM-M since the combined (bilingual) unlabeled sentences could be more helpful than the unlabeled monolingual sentences.

Co-Training with SVMs (Co-SVM): This method applies SVM-based co-training given both the labeled training data and the unlabeled parallel data following Wan (2009). First, two monolingual SVM classifiers are built based on only the corresponding labeled data, and then they are bootstrapped by adding the most confident predicted examples from the unlabeled data into the training set. We run bootstrapping for 100 iterations. In each iteration, we select the most confidently predicted 50 positive and 50 negative sentences from each of the two classifiers, and take the union of the resulting 200 sentence pairs as the newly labeled training data. (Examples with conflicting labels within the pair are not included.)

5 Results and Analysis

In our experiments, the methods are tested in the two data settings with the corresponding unlabeled parallel corpus as mentioned in Section 4.[6] We use 5-fold cross-validation and report average accuracy (also MicroF1 in this case) and MacroF1 scores. Unigrams are used as binary features for all models, as Pang et al. (2002) showed that binary features perform better than frequency features for sentiment classification. The weights for unlabeled data and regularization, λ_1 and λ_2, are set to 1 unless otherwise stated. Later, we will show that the proposed approach performs well with a wide range of parameter values.[7]

5.1 Method Comparison

We first compare the proposed joint model (**Joint**) with the baselines in Table 2. As seen from the table, the proposed approach outperforms all five baseline methods in terms of both accuracy and MacroF1 for both English and Chinese and in both of the data settings.[8] By making use of the unlabeled parallel data, our proposed approach improves the accuracy, compared to MaxEnt, by 8.12% (or 33.27% error reduction) on English and 3.44% (or 16.92% error reduction) on Chinese in the first setting, and by 5.07% (or 19.67% error reduction) on English and 3.87% (or 19.4% error reduction) on Chinese in the second setting.

Among the baselines, the best is Co-SVM; TSVMs do not always improve performance using the unlabeled data compared to the standalone SVM; and TSVM-B outperforms TSVM-M except for Chinese in the second setting. The MPQA data is more difficult in general compared to the NTCIR data. Without unlabeled parallel data, the performance on the Chinese data is better than on the English data, which is consistent with results reported in NTCIR-6 (Seki et al., 2007).

Overall, the unlabeled parallel data improves classification accuracy for both languages when using our proposed joint model and Co-SVM. The joint model makes better use of the unlabeled parallel data than Co-SVM or TSVMs presumably because of its attempt to jointly optimize the two monolingual models via soft (probabilistic) assignments of the unlabeled instances to classes in each iteration, instead of the hard assignments in Co-SVM and TSVMs. Although English sentiment

[6] The results reported in this section employ Equation 4. Preliminary experiments showed that Equation 5 does not significantly improve the performance in our case, which is reasonable since we choose only sentence pairs with the highest translation probabilities to be our unlabeled data (see Section 4.1).

[7] The code is at http://sites.google.com/site/lubin2010.

[8] Significance is tested using paired t-tests with $p<0.05$: $^{\epsilon}$ denotes statistical significance compared to the corresponding performance of MaxEnt; * denotes statistical significance compared to SVM; and $^{\Gamma}$ denotes statistical significance compared to Co-SVM.

	Setting 1: NTCIR-EN+NTCIR-CH				**Setting 2**: MPQA+NTCIR-CH			
	Accuracy		MacroF1		Accuracy		MacroF1	
	English	Chinese	English	Chinese	English	Chinese	English	Chinese
MaxEnt	75.59	79.67	66.61*	79.34	74.22	79.67	65.09*	79.34
SVM	76.34	81.02	61.12	80.75$^{\epsilon}$	76.74$^{\epsilon}$	81.02	61.35	80.75$^{\epsilon}$
TSVM-M	73.46	80.21	55.33	79.99	72.89	81.14	52.82	79.99
TSVM-B	78.36	81.60$^{\epsilon}$	65.53	81.42	76.42$^{\epsilon}$	78.51	61.66	78.32
Co-SVM	82.44$^{\epsilon*}$	82.79$^{\epsilon}$	72.61$^{\epsilon*}$	82.67$^{\epsilon*}$	78.18$^{\epsilon*}$	82.63$^{\epsilon*}$	68.03$^{\epsilon*}$	82.51$^{\epsilon*}$
Joint	**83.71**$^{\epsilon*}$	**83.11**$^{\epsilon*}$	**75.89**$^{\epsilon*\Gamma}$	**82.97**$^{\epsilon*}$	**79.29**$^{\epsilon*\Gamma}$	**83.54**$^{\epsilon*}$	**72.58**$^{\epsilon*\Gamma}$	**83.37**$^{\epsilon*}$

Table 2: Comparison of Results

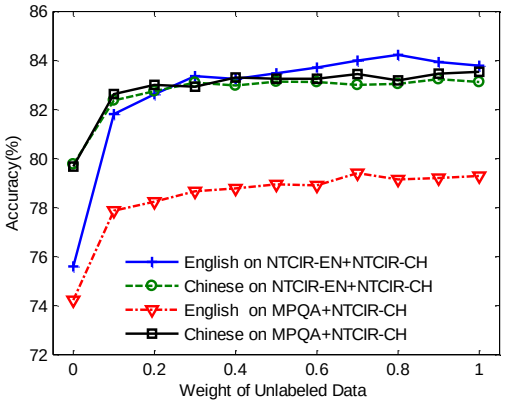


Figure 1. Accuracy vs. Weight of Unlabeled Data

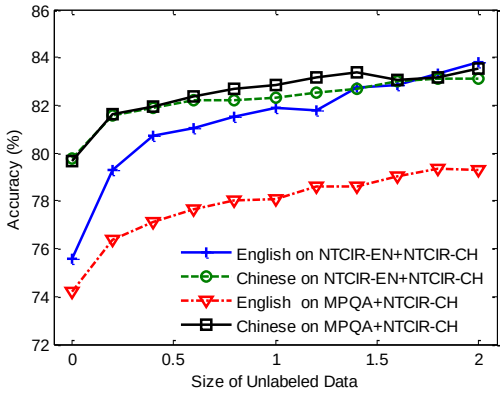


Figure 2. Accuracy vs. Amount of Unlabeled Data

classification alone is more difficult than Chinese for our datasets, we obtain greater performance gains for English by exploiting unlabeled parallel data as well as the Chinese labeled data.

5.2 Varying the Weight and Amount of Unlabeled Data

Figure 1 shows the accuracy curve of the proposed approach for the two data settings when varying the weight for the unlabeled data, λ_1, from 0 to 1. When λ_1 is set to 0, the joint model degenerates to two MaxEnt models trained with only the labeled data.

We can see that the performance gains for the proposed approach are quite remarkable even when λ_1 is set to 0.1; performance is largely stable after λ_1 reaches 0.4. Although MPQA is more difficult in general compared to the NTCIR data, we still see steady improvements in performance with unlabeled parallel data. Overall, the proposed approach performs quite well for a wide range of parameter values of λ_1.

Figure 2 shows the accuracy curve of the proposed approach for the two data settings when varying the amount of unlabeled data from 0 to 20,000 instances. We see that the performance of the proposed approach improves steadily by adding more and more unlabeled data. However, even with only 2,000 unlabeled sentence pairs, the proposed approach still produces large performance gains.

5.3 Results on Pseudo-Parallel Unlabeled Data

As discussed in Section 3.4, we generate pseudo-parallel data by translating the monolingual sentences in each setting using Google's machine translation system. Figures 3 and 4 show the performance of our model using the pseudo-parallel data versus the real parallel data, in the two settings, respectively. The EN->CH pseudo-parallel data consists of the English unlabeled data and its automatic Chinese translation, and vice versa.

Although not as significant as those with parallel data, we can still obtain improvements using the pseudo-parallel data, especially in the first setting. The difference between using parallel versus pseudo-parallel data is around 2-4% in Figures 3 and 4, which is reasonable since the quality of the pseudo-parallel data is not as good as that of the parallel data. Therefore, the performance using pseudo-parallel data is better with a small weight (e.g. λ_1= 0.1) in some cases.

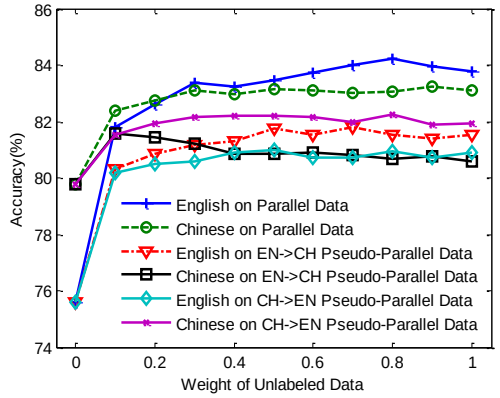


Figure 3. Accuracy with Pseudo-Parallel Unlabeled Data in **Setting 1**

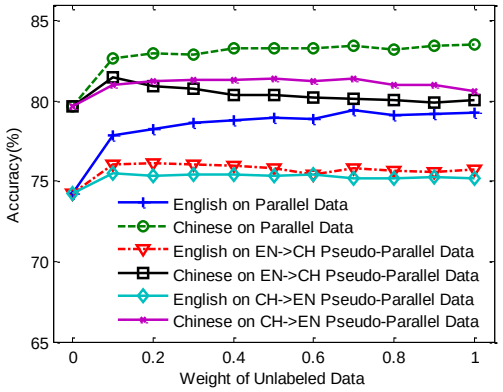


Figure 4. Accuracy with Pseudo-Parallel Unlabeled Data in **Setting 2**

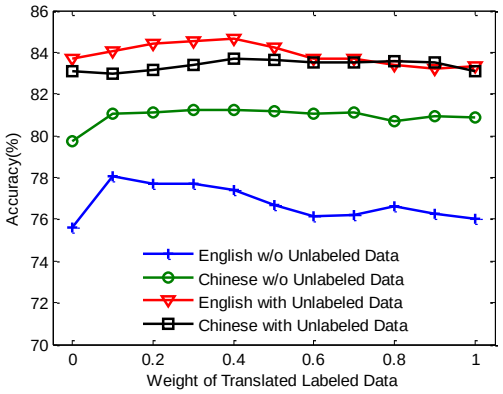


Figure 5. Accuracy with Pseudo-Parallel Labeled Data in **Setting 1**

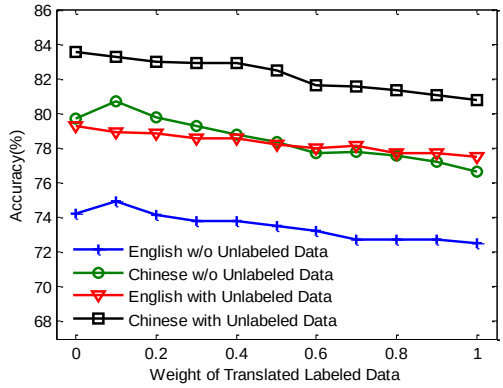


Figure 6. Accuracy with Pseudo-Parallel Labeled Data in **Setting 2**

5.4 Adding Pseudo-Parallel Labeled Data

In this section, we investigate how adding automatically translated labeled data might influence the performance as mentioned in Section 3.4. We use only the translated labeled data to train classifiers, and then directly classify the test data. The average accuracies in setting 1 are 66.61% and 63.11% on English and Chinese, respectively; while the accuracies in setting 2 are 58.43% and 54.07% on English and Chinese, respectively. This result is reasonable because of the language gap between the original language and the translated language. In addition, the class distributions of the English labeled data and the Chinese are quite different (30% vs. 55% for positive as shown in Table 1).

Figures 5 and 6 show the accuracies when varying the weight of the translated labeled data vs. the labeled data, with and without the unlabeled parallel data. From Figure 5 for setting 1, we can see that the translated data can be helpful given the labeled data and even the unlabeled data, as long as λ_3 is small; while in Figure 6, the translated data decreases the performance in most cases for setting 2. One possible reason is that in the first data setting, the NTCIR English data covers the same topics as the NTCIR Chinese data and thus direct translation is helpful, while the English and Chinese topics are quite different in the second data setting, and thus direct translation hurts the performance given the existing labeled data in each language.

5.5 Discussion

To further understand what contributions our proposed approach makes to the performance gain, we look inside the parameters in the MaxEnt models learned before and after adding the parallel unlabeled data. Table 3 shows the features in the model learned from the labeled data that have the largest weight change after adding the parallel data;

	Word	Weight		
		Before	After	Change
Positive	important	0.452	1.659	1.207
	cooperation	0.325	1.492	1.167
	support	0.533	1.483	0.950
	importance	0.450	1.193	0.742
	agreed	0.347	1.061	0.714
Negative	difficulties	0.018	0.663	0.645
	not	0.202	0.844	0.641
	never	0.245	0.879	0.634
	germany	0.035	0.664	0.629
	taiwan	0.590	1.216	0.626

Table 3. Original Features with Largest Weight Change

Positive		Negative	
Word	Weight	Word	Weight
friendly	0.701	german	0.783
principles	0.684	arduous	0.531
hopes	0.630	oppose	0.511
hoped	0.553	administrations	0.431
cooperative	0.552	oau[9]	0.408

Table 4. New Features Learned from Unlabeled Data

and Table 4 shows the newly learned features from the unlabeled data with the largest weights.

From Table 3[10] we can see that the weight changes of the original features are quite reasonable, e.g. the top words in the positive class are obviously positive and the proposed approach gives them higher weights. The new features also seem reasonable given the knowledge that the labeled and unlabeled data includes negative news about for specific topics (e.g. Germany, Taiwan),.

We also examine the process of joint training by checking the performance on test data and the agreement of the two monolingual models on the unlabeled parallel data in both settings. The average agreement across 5 folds is 85.06% and 73.87% in settings 1 and 2, respectively, before the joint training, and increases to 100% and 99.89%, respectively, after 100 iterations of joint training. Although the average agreement has already increased to 99.50% and 99.02% in settings 1 and 2, respectively, after 30 iterations, the performance on the test set steadily improves in both settings until around 50-60 iterations, and then becomes relatively stable after that.

Examination of those sentence pairs in setting 2 for which the two monolingual models still disagree after 100 iterations of joint training often produces sentences that are not quite parallel, e.g.:

English: The two sides attach great importance to international cooperation on protection and promotion of human rights.

Chinese: 双方认为,在人权问题上不能采取"双重标准",反对在国际关系中利用人权问题施压。(Both sides agree that double standards on the issue of human rights are to be avoided, and are opposed to using pressure on human rights issues in international relations.)

Since the two sentences discuss *human rights* from very different perspectives, it is reasonable that the two monolingual models will classify them with different polarities (i.e. positive for the English sentence and negative for the Chinese sentence) even after joint training.

6 Conclusion

In this paper, we study bilingual sentiment classification and propose a joint model to simultaneously learn better monolingual sentiment classifiers for each language by exploiting an unlabeled parallel corpus together with the labeled data available for each language. Our experiments show that the proposed approach can significantly improve sentiment classification for both languages. Moreover, the proposed approach continues to produce (albeit smaller) performance gains when employing pseudo-parallel data from machine translation engines.

In future work, we would like to apply the joint learning idea to other learning frameworks (e.g. SVMs), and to extend the proposed model to handle word-level parallel information, e.g. bilingual dictionaries or word alignment information. Another issue is to investigate how to improve multilingual sentiment analysis by exploiting comparable corpora.

Acknowledgments

We thank Shuo Chen, Long Jiang, Thorsten Joachims, Lillian Lee, Myle Ott, Yan Song, Xiaojun Wan, Ainur Yessenalina, Jingbo Zhu and the anonymous reviewers for many useful comments and discussion. This work was supported in part by National Science Foundation Grants BCS-0904822, BCS-0624277, IIS-0968450; and by a gift from Google. Chenhao Tan is supported by NSF (DMS-0808864), ONR (YIP-N000140910911), and a grant from Microsoft.

[9]This is an abbreviation for the Organization of African Unity.

[10]The features and weights in Tables 3 and 4 are extracted from the English model in the first fold of setting 1.

References

Massih-Reza Amini, Cyril Goutte, and Nicolas Usunier. 2010. Combining coregularization and consensus-based self-training for multilingual text categorization. In *Proceeding of SIGIR'10*.

Carmen Banea, Rada Mihalcea, and Janyce Wiebe. 2010. Multilingual subjectivity: Are more languages better? In *Proceedings of COLING'10*.

Carmen Banea, Rada Mihalcea, Janyce Wiebe, and Samer Hassan. 2008. Multilingual subjectivity analysis using machine translation. In *Proceedings of EMNLP'08*.

Adam L. Berger, Stephen A. Della Pietra and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1).

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspond-dence learning. In *Proceedings of EMNLP'06*.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT'98*.

Jordan Boyd-Graber and Philip Resnik. 2010. Holistic sentiment analysis across languages: Multilingual supervised Latent Dirichlet Allocation. In *Proceedings of EMNLP'10*.

Eric Breck, Yejin Choi, and Claire Cardie. 2007. Identifying expressions of opinion in context. In *Proceedings of IJCAI'07*.

David Burkett, Slav Petrov, John Blitzer, and Dan Klein. 2010. Learning better monolingual models with unannotated bilingual text. In *Proceedings of CoNLL'10*.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proceedings of EMNLP'08*.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of EMNLP'08*.

Wei Gao, John Blitzer, Ming Zhou, and Kam-Fai Wong. 2009. Exploiting bilingual information to improve web search. In *Proceedings of ACL/IJCNLP'09*.

Minqing Hu and Bing Liu. 2004. Mining opinion features in customer reviews. In *Proceedings of AAAI'04*.

Ido Dagan, and Alon Itai. 1994. Word sense disambiguation using a second language monolingual corpus, Computational Linguistics, 20(4): 563-596.

Thorsten Joachims. 1999a. Making Large-Scale SVM Learning Practical. In: *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola (ed.), MIT Press.

Thorsten Joachims. 1999b. Transductive inference for text classification using support vector machines. In *Proceedings of ICML'99*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of NAACL'06*.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, (45): 503–528.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of CoNLL'02*.

Rada Mihalcea, Carmen Banea, and Janyce Wiebe. 2007. Learning multilingual subjective language via cross-lingual projections. In *Proceedings of ACL'07*.

Dragos S. Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4): 477–504.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using CRFs with hidden variables. In *Proceedings of NAACL/HLT '10*.

Kamal Nigam, Andrew K. Mccallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2): 103–134.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1): 19-51.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis, *Foundations and Trends in Information Retrieval*, Now Publishers.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of EMNLP'02*.

Peter Prettenhofer and Benno Stein. 2010. Cross-language text classification using structural correspondence learning. In *Proceedings of ACL'10*.

Adwait Ratnaparkhi. 1997. A simple introduction to maximum entropy models for natural language processing. Technical Report 97-08, University of Pennsylvania.

Julia M. Schulz, Christa Womser-Hacker, and Thomas Mandl. 2010. Multilingual corpus development for opinion mining. In *Proceedings of LREC'10*.

Yohei Seki, David Kirk Evans, Lun-Wei Ku, Le Sun, Hsin-His Chen, and Noriko Kando. 2008. Overview of multilingual opinion analysis task at NTCIR-7. In *Proceedings of the NTCIR-7 Workshop*.

Yohei Seki, David K. Evans, Lun-Wei Ku, Le Sun, Hsin-His Chen, Noriko Kando, and Chin-Yew Lin. 2007. Overview of opinion analysis pilot task at NTCIR-6. In *Proceedings of the NTCIR-6 Workshop*.

Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. 2005. A co-regularization approach to semi-supervised learning with multiple views. In *Proceedings of ICML'05*.

Noah A. Smith. 2006. Novel estimation methods for unsupervised discovery of latent structure in natural language text. Ph.D. thesis, Department of Computer Science, Johns Hopkins University.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky and Christopher Manning. 2005. A conditional random field word segmenter. In *Proeedings of the 4th SIGHAN Workshop*.

Peter D. Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews, In *Proceedings of ACL'02*.

Xiaojun Wan. 2008. Using Bilingual Knowledge and Ensemble Techniques for Unsupervised Chinese Sentiment Analysis. In *Proceedings of EMNLP'08.*

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of ACL/AFNLP'09*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2- 3): 165-210.

Duo Zhang, Qiaozhu Mei, and ChengXiang Zhai. 2010. Cross-lingual latent topic extraction, In *Proceedings of ACL'10*.

Hai Zhao, Yan Song, Chunyu Kit, and Guodong Zhou. 2009. Cross language dependency parsing using a bilingual lexicon. In *Proceedings of ACL/IJCNLP'09*.

Xiaojin Zhu and Andrew B. Goldberg. 2009. *Introduction to Semi-Supervised Learning*. Morgan & Claypool Publishers.

Appendix A. Equation Deduction

In this appendix, we derive the gradient for the objective function in Equation 3, which is used in parameter estimation. As mentioned in Section 3.3, the parameters can be learned by finding:

$$\begin{aligned}(\theta_1^*, \theta_2^*) &= \underset{(\theta_1,\theta_2)}{\operatorname{argmax}} \mathcal{L}(\theta_1, \theta_2 | D_1, D_2, U) \\ &= \underset{(\theta_1,\theta_2)}{\operatorname{argmax}} \log \mathcal{L}(\theta_1, \theta_2 | D_1, D_2, U) \\ &= \underset{(\theta_1,\theta_2)}{\operatorname{argmax}} \{\log p(Y_1|X_1;\theta_1) p(Y_2|X_2;\theta_2) \\ &\quad + \textstyle\sum_{i=1}^{u} \log p(y_i^{1'}, y_i^{2'} | x_i^{1'}, x_i^{2'}; \theta_1, \theta_2)\}\end{aligned}$$

Since the first term on the right-hand side is just the expression for the standard MaxEnt problem, we will focus on the gradient for the second term, and denote $\log p(y_i^{1'}, y_i^{2'} | x_i^{1'}, x_i^{2'}; \theta_1, \theta_2)$ as $(*)$.

Let $v \in \{1,2\}$ denote L_1 or L_2, and θ_k^v be the kth weight in the vector θ_v. For brevity, we drop the $'$ in the above notation, and write x_i^v to denote $x_i^{v'}$. Then the partial derivative of (*) based on Equation 4 with respect to θ_k^v is as follows:

$$\frac{\partial(*)}{\partial \theta_k^v} = \frac{\sum_{y_i^*} p\left(y_i^* | x_i^{\bar{v}}; \theta_{\bar{v}}\right) \frac{\partial}{\partial \theta_k^v} p(y_i^* | x_i^v; \theta_v)}{\sum_{y_i'} p(y_i' | x_i^v; \theta_v) p(y_i' | x_i^{\bar{v}}; \theta_{\bar{v}})} \quad (1)$$

Further, we obtain:

$$\begin{aligned}&\frac{\partial}{\partial \theta_k^v} p(y_i^* | x_i^v; \theta_v) = \frac{\partial}{\partial \theta_k^v} \frac{\exp\left(\vec{\theta}_v \vec{f}(x_i^v, y_i^*)\right)}{\sum_{y_i'} \exp\left(\vec{\theta}_v \vec{f}(x_i^v, y_i')\right)} \\ &= \frac{\exp\left(\vec{\theta}_v \vec{f}(x_i^v, y_i^*)\right)}{\sum_{y_i'} \exp\left(\vec{\theta}_v \vec{f}(x_i^v, y_i')\right)} f_k^v(x_i^v, y_i^*) - \\ &\quad \frac{\exp\left(\vec{\theta}_v \vec{f}(x_i^v, y_i^*)\right)}{\left[\sum_{y_i'} \exp\left(\vec{\theta}_v \vec{f}(x_i^v, y_i')\right)\right]^2} \sum_{y_i'} \left\{\exp\left(\vec{\theta}_v \vec{f}(x_i^v, y_i')\right) f_k^v(x_i^v, y_i')\right\} \\ &= p(y_i^* | x_i^v; \theta_v) \left\{f_k^v(x_i^v, y_i^*) - \textstyle\sum_{y_i'} p(y_i' | x_i^v; \theta_v) f_k^v(x_i^v, y_i')\right\} \quad (2)\end{aligned}$$

Merge (2) into (1), we get:

$$\begin{aligned}\frac{\partial(*)}{\partial \theta_k^v} &= \frac{1}{\sum_{y_i'} p(y_i' | x_i^v; \theta_v) p(y_i' | x_i^{\bar{v}}; \theta_{\bar{v}})} \sum_{y_i^*} \{p\left(y_i^* | x_i^{\bar{v}}; \theta_{\bar{v}}\right) p(y_i^* | x_i^v; \theta_v) \\ &\quad [f_k^v(x_i^v, y_i^*) - \textstyle\sum_{y_i'} p(y_i' | x_i^v; \theta_v) f_k^v(x_i^v, y_i')]\} \\ &= \textstyle\sum_{y_i^*} p\left(y_i^* | x_i^1; \theta_1\right) p\left(y_i^* | x_i^2; \theta_2\right) f_k^v(x_i^v, y_i^*) - \\ &\quad \textstyle\sum_{y_i'} p(y_i' | x_i^v; \theta_v) f_k^v(x_i^v, y_i') \\ &= \textstyle\sum_{y_i^*} f_k^v(x_i^v, y_i^*) \{p(y_i^* | x_i^1; \theta_1) p(y_i^* | x_i^2; \theta_2) - p(y_i^* | x_i^v; \theta_v)\}\end{aligned}$$

A Pilot Study of Opinion Summarization in Conversations

Dong Wang **Yang Liu**
The University of Texas at Dallas
dongwang,yangl@hlt.utdallas.edu

Abstract

This paper presents a pilot study of opinion summarization on conversations. We create a corpus containing extractive and abstractive summaries of speaker's opinion towards a given topic using 88 telephone conversations. We adopt two methods to perform extractive summarization. The first one is a sentence-ranking method that linearly combines scores measured from different aspects including topic relevance, subjectivity, and sentence importance. The second one is a graph-based method, which incorporates topic and sentiment information, as well as additional information about sentence-to-sentence relations extracted based on dialogue structure. Our evaluation results show that both methods significantly outperform the baseline approach that extracts the longest utterances. In particular, we find that incorporating dialogue structure in the graph-based method contributes to the improved system performance.

1 Introduction

Both sentiment analysis (opinion recognition) and summarization have been well studied in recent years in the natural language processing (NLP) community. Most of the previous work on sentiment analysis has been conducted on reviews. Summarization has been applied to different genres, such as news articles, scientific articles, and speech domains including broadcast news, meetings, conversations and lectures. However, opinion summarization has not been explored much. This can be useful for many domains, especially for processing the increasing amount of conversation recordings (telephone conversations, customer service, round-table discussions or interviews in broadcast programs) where we often need to find a person's opinion or attitude, for example, "how does the speaker think about capital punishment and why?". This kind of questions can be treated as a topic-oriented opinion summarization task. Opinion summarization was run as a pilot task in Text Analysis Conference (TAC) in 2008. The task was to produce summaries of opinions on specified targets from a set of blog documents. In this study, we investigate this problem using spontaneous conversations. The problem is defined as, given a conversation and a topic, a summarization system needs to generate a summary of the speaker's opinion towards the topic.

This task is built upon opinion recognition and topic or query based summarization. However, this problem is challenging in that: (a) Summarization in spontaneous speech is more difficult than well structured text (Mckeown et al., 2005), because speech is always less organized and has recognition errors when using speech recognition output; (b) Sentiment analysis in dialogues is also much harder because of the genre difference compared to other domains like product reviews or news resources, as reported in (Raaijmakers et al., 2008); (c) In conversational speech, information density is low and there are often off topic discussions, therefore presenting a need to identify utterances that are relevant to the topic.

In this paper we perform an exploratory study on opinion summarization in conversations. We compare two unsupervised methods that have been

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 331–339,
Portland, Oregon, June 19-24, 2011.

widely used in extractive summarization: sentence-ranking and graph-based methods. Our system attempts to incorporate more information about topic relevancy and sentiment scores. Furthermore, in the graph-based method, we propose to better incorporate the dialogue structure information in the graph in order to select salient summary utterances. We have created a corpus of reasonable size in this study. Our experimental results show that both methods achieve better results compared to the baseline.

The rest of this paper is organized as follows. Section 2 briefly discusses related work. Section 3 describes the corpus and annotation scheme we used. We explain our opinion-oriented conversation summarization system in Section 4 and present experimental results and analysis in Section 5. Section 6 concludes the paper.

2 Related Work

Research in document summarization has been well established over the past decades. Many tasks have been defined such as single-document summarization, multi-document summarization, and query-based summarization. Previous studies have used various domains, including news articles, scientific articles, web documents, reviews. Recently there is an increasing research interest in speech summarization, such as conversational telephone speech (Zhu and Penn, 2006; Zechner, 2002), broadcast news (Maskey and Hirschberg, 2005; Lin et al., 2009), lectures (Zhang et al., 2007; Furui et al., 2004), meetings (Murray et al., 2005; Xie and Liu, 2010), voice mails (Koumpis and Renals, 2005). In general speech domains seem to be more difficult than well written text for summarization. In previous work, unsupervised methods like Maximal Marginal Relevance (MMR), Latent Semantic Analysis (LSA), and supervised methods that cast the extraction problem as a binary classification task have been adopted. Prior research has also explored using speech specific information, including prosodic features, dialog structure, and speech recognition confidence.

In order to provide a summary over opinions, we need to find out which utterances in the conversation contain opinion. Most previous work in sentiment analysis has focused on reviews (Pang and Lee, 2004; Popescu and Etzioni, 2005; Ng et al., 2006) and news resources (Wiebe and Riloff, 2005). Many kinds of features are explored, such as lexical features (unigram, bigram and trigram), part-of-speech tags, dependency relations. Most of prior work used classification methods such as naive Bayes or SVMs to perform the polarity classification or opinion detection. Only a handful studies have used conversational speech for opinion recognition (Murray and Carenini, 2009; Raaijmakers et al., 2008), in which some domain-specific features are utilized such as structural features and prosodic features.

Our work is also related to question answering (QA), especially opinion question answering. (Stoyanov et al., 2005) applies a subjectivity filter based on traditional QA systems to generate opinionated answers. (Balahur et al., 2010) answers some specific opinion questions like "Why do people criticize Richard Branson?" by retrieving candidate sentences using traditional QA methods and selecting the ones with the same polarity as the question. Our work is different in that we are not going to answer specific opinion questions, instead, we provide a summary on the speaker's opinion towards a given topic.

There exists some work on opinion summarization. For example, (Hu and Liu, 2004; Nishikawa et al., 2010) have explored opinion summarization in review domain, and (Paul et al., 2010) summarizes contrastive viewpoints in opinionated text. However, opinion summarization in spontaneous conversation is seldom studied.

3 Corpus Creation

Though there are many annotated data sets for the research of speech summarization and sentiment analysis, there is no corpus available for opinion summarization on spontaneous speech. Thus for this study, we create a new pilot data set using a subset of the Switchboard corpus (Godfrey and Holliman, 1997).[1] These are conversational telephone speech between two strangers that were assigned a topic to talk about for around 5 minutes. They were told to find the opinions of the other person. There are 70 topics in total. From the Switchboard cor-

[1] Please contact the authors to obtain the data.

pus, we selected 88 conversations from 6 topics for this study. Table 1 lists the number of conversations in each topic, their average length (measured in the unit of dialogue acts (DA)) and standard deviation of length.

topic	#Conv.	avg len	stdev
space flight and exploration	6	165.5	71.40
capital punishment	24		
gun control	15		
universal health insurance	9		
drug testing	12		
universal public service	22		

Table 1: Corpus statistics: topic description, number of conversations in each topic, average length (number of dialog acts), and standard deviation.

We recruited 3 annotators that are all undergraduate computer science students. From the 88 conversations, we selected 18 (3 from each topic) and let all three annotators label them in order to study inter-annotator agreement. The rest of the conversations has only one annotation.

The annotators have access to both conversation transcripts and audio files. For each conversation, the annotator writes an abstractive summary of up to 100 words for each speaker about his/her opinion or attitude on the given topic. They were told to use the words in the original transcripts if possible. Then the annotator selects up to 15 DAs (no minimum limit) in the transcripts for each speaker, from which their abstractive summary is derived. The selected DAs are used as the human generated extractive summary. In addition, the annotator is asked to select an overall opinion towards the topic for each speaker among five categories: strongly support, somewhat support, neutral, somewhat against, strongly against. Therefore for each conversation, we have an abstractive summary, an extractive summary, and an overall opinion for each speaker. The following shows an example of such annotation for speaker B in a dialogue about "capital punishment":

[**Extractive Summary**]

I think I've seen some statistics that say that, uh, it's more expensive to kill somebody than to keep them in prison for life.

committing them mostly is, you know, either crimes of passion or at the moment

or they think they're not going to get caught

but you also have to think whether it's worthwhile on the individual basis, for example, someone like, uh, jeffrey dahlmer,

by putting him in prison for life, there is still a possibility that he will get out again.

I don't think he could ever redeem himself,

but if you look at who gets accused and who are the ones who actually get executed, it's very racially related – and ethnically related

[**Abstractive Summary**]

B is against capital punishment except under certain circumstances. B finds that crimes deserving of capital punishment are "crimes of the moment" and as a result feels that capital punishment is not an effective deterrent. however, B also recognizes that on an individual basis some criminals can never "redeem" themselves.

[**Overall Opinion**]

Somewhat against

Table 2 shows the compression ratio of the extractive summaries and abstractive summaries as well as their standard deviation. Because in conversations, utterance length varies a lot, we use words as units when calculating the compression ratio.

	avg ratio	stdev
extractive summaries	0.26	0.13
abstractive summaries	0.13	0.06

Table 2: Compression ratio and standard deviation of extractive and abstractive summaries.

We measured the inter-annotator agreement among the three annotators for the 18 conversations (each has two speakers, thus 36 "documents" in total). Results are shown in Table 3. For the extractive or abstractive summaries, we use ROUGE scores (Lin, 2004), a metric used to evaluate automatic summarization performance, to measure the pairwise agreement of summaries from different annotators. ROUGE F-scores are shown in the table for different matches, unigram (R-1), bigram (R-2), and longest subsequence (R-L). For the overall opinion category, since it is a multiclass label (not binary decision), we use Krippendorff's α coefficient to measure human agreement, and the difference function for interval data: $\delta^2_{ck} = (c-k)^2$ (where c, k are the interval values, on a scale of 1 to 5 corresponding to the five categories for the overall opinion).

We notice that the inter-annotator agreement for extractive summaries is comparable to other speech

extractive summaries	R-1	0.61
	R-2	0.52
	R-L	0.61
abstractive summaries	R-1	0.32
	R-2	0.13
	R-L	0.25
overall opinion	$\alpha = 0.79$	

Table 3: Inter-annotator agreement for extractive and abstractive summaries, and overall opinion.

summary annotation (Liu and Liu, 2008). The agreement on abstractive summaries is much lower than extractive summaries, which is as expected. Even for the same opinion or sentence, annotators use different words in the abstractive summaries. The agreement for the overall opinion annotation is similar to other opinion/emotion studies (Wilson, 2008b), but slightly lower than the level recommended by Krippendorff for reliable data ($\alpha = 0.8$) (Hayes and Krippendorff, 2007), which shows it is even difficult for humans to determine what opinion a person holds (support or against something). Often human annotators have different interpretations about the same sentence, and a speaker's opinion/attitude is sometimes ambiguous. Therefore this also demonstrates that it is more appropriate to provide a summary rather than a simple opinion category to answer questions about a person's opinion towards something.

4 Opinion Summarization Methods

Automatic summarization can be divided into extractive summarization and abstractive summarization. Extractive summarization selects sentences from the original documents to form a summary; whereas abstractive summarization requires generation of new sentences that represent the most salient content in the original documents like humans do. Often extractive summarization is used as the first step to generate abstractive summary.

As a pilot study for the problem of opinion summarization in conversations, we treat this problem as an extractive summarization task. This section describes two approaches we have explored in generating extractive summaries. The first one is a sentence-ranking method, in which we measure the salience of each sentence according to a linear combination of scores from several dimensions. The second one is a graph-based method, which incorporates the dialogue structure in ranking. We choose to investigate these two methods since they have been widely used in text and speech summarization, and perform competitively. In addition, they do not require a large labeled data set for modeling training, as needed in some classification or feature based summarization approaches.

4.1 Sentence Ranking

In this method, we use Equation 1 to assign a score to each DA s, and select the most highly ranked ones until the length constriction is satisfied.

$$\begin{aligned} score(s) &= \lambda_{sim} sim(s, D) + \lambda_{rel} REL(s, topic) \\ &\quad + \lambda_{sent} sentiment(s) + \lambda_{len} length(s) \\ \sum_i \lambda_i &= 1 \end{aligned} \quad (1)$$

- $sim(s, D)$ is the cosine similarity between DA s and all the utterances in the dialogue from the same speaker, D. It measures the relevancy of s to the entire dialogue from the target speaker. This score is used to represent the salience of the DA. It has been shown to be an important indicator in summarization for various domains. For cosine similarity measure, we use TF*IDF (term frequency, inverse document frequency) term weighting. The IDF values are obtained using the entire Switchboard corpus, treating each conversation as a document.

- $REL(s, topic)$ measures the topic relevance of DA s. It is the sum of the topic relevance of all the words in the DA. We only consider the content words for this measure. They are identified using TreeTagger toolkit.[2] To measure the relevance of a word to a topic, we use Pairwise Mutual Information (PMI):

$$PMI(w, topic) = log_2 \frac{p(w\&topic)}{p(w)p(topic)} \quad (2)$$

[2] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html

where all the statistics are collected from the Switchboard corpus: $p(w\&topic)$ denotes the probability that word w appears in a dialogue of topic t, and $p(w)$ is the probability of w appearing in a dialogue of any topic. Since our goal is to rank DAs in the same dialog, and the topic is the same for all the DAs, we drop $p(topic)$ when calculating PMI scores. Because the value of $PMI(w, topic)$ is negative, we transform it into a positive one (denoted by $PMI^+(w, topic)$) by adding the absolute value of the minimum value. The final relevance score of each sentence is normalized to $[0, 1]$ using linear normalization:

$$\begin{aligned} REL_{orig}(s, topic) &= \sum_{w \in s} PMI^+(w, topic) \\ REL(s, topic) &= \frac{REL_{orig}(s, topic) - Min}{Max - Min} \end{aligned}$$

- $sentiment(s)$ indicates the probability that utterance s contains opinion. To obtain this, we trained a maximum entropy classifier with a bag-of-words model using a combination of data sets from several domains, including movie data (Pang and Lee, 2004), news articles from MPQA corpus (Wilson and Wiebe, 2003), and meeting transcripts from AMI corpus (Wilson, 2008a). Each sentence (or DA) in these corpora is annotated as "subjective" or "objective". We use each utterance's probability of being "subjective" predicted by the classifier as its sentiment score.

- $length(s)$ is the length of the utterance. This score can effectively penalize the short sentences which typically do not contain much important content, especially the backchannels that appear frequently in dialogues. We also perform linear normalization such that the final value lies in $[0, 1]$.

4.2 Graph-based Summarization

Graph-based methods have been widely used in document summarization. In this approach, a document is modeled as an adjacency matrix, where each node represents a sentence, and the weight of the edge between each pair of sentences is their similarity (cosine similarity is typically used). An iterative process is used until the scores for the nodes converge. Previous studies (Erkan and Radev, 2004) showed that this method can effectively extract important sentences from documents. The basic framework we use in this study is similar to the query-based graph summarization system in (Zhao et al., 2009). We also consider sentiment and topic relevance information, and propose to incorporate information obtained from dialog structure in this framework. The score for a DA s is based on its content similarity with all other DAs in the dialogue, the connection with other DAs based on the dialogue structure, the topic relevance, and its subjectivity, that is:

$$\begin{aligned} score(s) &= \lambda_{sim} \sum_{v \in C} \frac{sim(s, v)}{\sum_{z \in C} sim(z, v)} score(v) \\ &\quad + \lambda_{rel} \frac{REL(s, topic)}{\sum_{z \in C} REL(z, topic)} \\ &\quad + \lambda_{sent} \frac{sentiment(s)}{\sum_{z \in C} sentiment(z)} \\ &\quad + \lambda_{adj} \sum_{v \in C} \frac{ADJ(s, v)}{\sum_{z \in C} ADJ(z, v)} score(v) \\ \sum_i \lambda_i &= 1 \end{aligned} \quad (3)$$

where C is the set of all DAs in the dialogue; $REL(s, topic)$ and $sentiment(s)$ are the same as those in the above sentence ranking method; $sim(s, v)$ is the cosine similarity between two DAs s and v. In addition to the standard connection between two DAs with an edge weight $sim(s, v)$, we introduce new connections $ADJ(s, v)$ to model dialog structure. It is a directed edge from s to v, defined as follows:

- If s and v are from the same speaker and within the same turn, there is an edge from s to v and an edge from v to s with weight $1/dis(s, v)$ ($ADJ(s, v) = ADJ(v, s) = 1/dis(s, v)$), where $dis(s, v)$ is the distance between s and v, measured based on their DA indices. This way the DAs in the same turn can reinforce each other. For example, if we consider that

one DA is important, then the other DAs in the same turn are also important.

- If s and v are from the same speaker, and separated only by one DA from another speaker with length less than 3 words (usually backchannel), there is an edge from s to v as well as an edge from v to s with weight 1 ($ADJ(s, v) = ADJ(v, s) = 1$).

- If s and v form a question-answer pair from two speakers, then there is an edge from question s to answer v with weight 1 ($ADJ(s, v) = 1$). We use a simple rule-based method to determine question-answer pairs — sentence s has question marks or contains "wh-word" (i.e., "what, how, why"), and sentence v is the immediately following one. The motivation for adding this connection is, if the score of a question sentence is high, then the answer's score is also boosted.

- If s and v form an agreement or disagreement pair, then there is an edge from v to s with weight 1 ($ADJ(v, s) = 1$). This is also determined by simple rules: sentence v contains the word "agree" or "disagree", s is the previous sentence, and from a different speaker. The reason for adding this is similar to the above question-answer pairs.

- If there are multiple edges generated from the above steps between two nodes, then we use the highest weight.

Since we are using a directed graph for the sentence connections to model dialog structure, the resulting adjacency matrix is asymmetric. This is different from the widely used graph methods for summarization. Also note that in the first sentence ranking method or the basic graph methods, summarization is conducted for each speaker separately. Utterances from one speaker have no influence on the summary decision for the other speaker. Here in our proposed graph-based method, we introduce connections between the two speakers, so that the adjacency pairs between them can be utilized to extract salient utterances.

5 Experiments

5.1 Experimental Setup

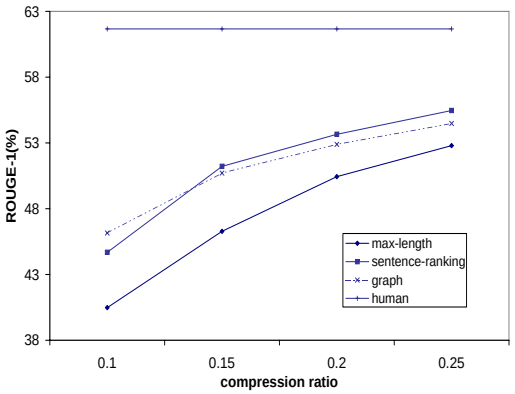
The 18 conversations annotated by all 3 annotators are used as test set, and the rest of 70 conversations are used as development set to tune the parameters (determining the best combination weights). In preprocessing we applied word stemming. We perform extractive summarization using different word compression ratios (ranging from 10% to 25%). We use human annotated dialogue acts (DA) as the extraction units. The system-generated summaries are compared to human annotated extractive and abstractive summaries. We use ROUGE as the evaluation metrics for summarization performance.

We compare our methods to two systems. The first one is a baseline system, where we select the longest utterances for each speaker. This has been shown to be a relatively strong baseline for speech summarization (Gillick et al., 2009). The second one is human performance. We treat each annotator's extractive summary as a system summary, and compare to the other two annotators' extractive and abstractive summaries. This can be considered as the upper bound of our system performance.

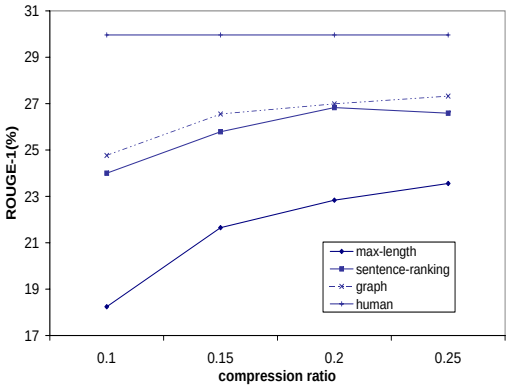
5.2 Results

From the development set, we used the grid search method to obtain the best combination weights for the two summarization methods. In the sentence-ranking method, the best parameters found on the development set are $\lambda_{sim} = 0, \lambda_{rel} = 0.3, \lambda_{sent} = 0.3, \lambda_{len} = 0.4$. It is surprising to see that the similarity score is not useful for this task. The possible reason is, in Switchboard conversations, what people talk about is diverse and in many cases only topic words (except stopwords) appear more than once. In addition, REL score is already able to catch the topic relevancy of the sentence. Thus, the similarity score is redundant here.

In the graph-based method, the best parameters are $\lambda_{sim} = 0, \lambda_{adj} = 0.3, \lambda_{rel} = 0.4, \lambda_{sent} = 0.3$. The similarity between each pair of utterances is also not useful, which can be explained with similar reasons as in the sentence-ranking method. This is different from graph-based summarization systems for text domains. A similar finding has also been shown in (Garg et al., 2009), where similarity be-



(a) compare to reference extractive summary



(b) compare to reference abstractive summary

Figure 1: ROUGE-1 F-scores compared to extractive and abstractive reference summaries for different systems: max-length, sentence-ranking method, graph-based method, and human performance.

tween utterances does not perform well in conversation summarization.

Figure 1 shows the ROUGE-1 F-scores comparing to human extractive and abstractive summaries for different compression ratios. Similar patterns are observed for other ROUGE scores such as ROUGE-2 or ROUGE-L, therefore they are not shown here. Both methods improve significantly over the baseline approach. There is relatively less improvement using a higher compression ratio, compared to a lower one. This is reasonable because when the compression ratio is low, the most salient utterances are not necessarily the longest ones, thus using more information sources helps better identify important sentences; but when the compression ratio is higher, longer utterances are more likely to be selected since they contain more content.

There is no significant difference between the two methods. When compared to extractive reference summaries, sentence-ranking is slightly better except for the compression ratio of 0.1. When compared to abstractive reference summaries, the graph-based method is slightly better. The two systems share the same topic relevance score (REL) and sentiment score, but the sentence-ranking method prefers longer DAs and the graph-based method prefers DAs that are emphasized by the ADJ matrix, such as the DA in the middle of a cluster of utterances from the same speaker, the answer to a question, etc.

5.3 Analysis

To analyze the effect of dialogue structure we introduce in the graph-based summarization method, we compare two configurations: $\lambda_{adj} = 0$ (only using REL score and sentiment score in ranking) and $\lambda_{adj} = 0.3$. We generate summaries using these two setups and compare with human selected sentences. Table 4 shows the number of false positive instances (selected by system but not by human) and false negative ones (selected by human but not by system). We use all three annotators' annotation as reference, and consider an utterance as positive if one annotator selects it. This results in a large number of reference summary DAs (because of low human agreement), and thus the number of false negatives in the system output is very high. As expected, a smaller compression ratio (fewer selected DAs in the system output) yields a higher false negative rate and a lower false positive rate. From the results, we can see that generally adding adjacency matrix information is able to reduce both types of errors except when the compression ratio is 0.15.

The following shows an example, where the third DA is selected by the system with $\lambda_{adj} = 0.3$, but not by $\lambda_{adj} = 0$. This is partly because the weight of the second DA is enhanced by the the question-

	$\lambda_{adj} = 0$		$\lambda_{adj} = 0.3$	
ratio	FP	FN	FP	FN
0.1	37	588	33	581
0.15	60	542	61	546
0.2	100	516	90	511
0.25	137	489	131	482

Table 4: The number of false positive (FP) and false negative (FN) instances using the graph-based method with $\lambda_{adj} = 0$ and $\lambda_{adj} = 0.3$ for different compression ratios.

answer pair (the first and the second DA), and thus subsequently boosting the score of the third DA.

A: Well what do you think?

B: Well, I don't know, I'm thinking about from one to ten what my no would be.

B: It would probably be somewhere closer to, uh, less control because I don't see, -

We also examined the system output and human annotation and found some reasons for the system errors:

(a) Topic relevance measure. We use the statistics from the Switchboard corpus to measure the relevance of each word to a given topic (PMI score), therefore only when people use the same word in different conversations of the topic, the PMI score of this word and the topic is high. However, since the size of the corpus is small, some topics only contain a few conversations, and some words only appear in one conversation even though they are topic-relevant. Therefore the current PMI measure cannot properly measure a word's and a sentence's topic relevance. This problem leads to many false negative errors (relevant sentences are not captured by our system).

(b) Extraction units. We used DA segments as units for extractive summarization, which can be problematic. In conversational speech, sometimes a DA segment is not a complete sentence because of overlaps and interruptions. We notice that annotators tend to select consecutive DAs that constitute a complete sentence, however, since each individual DA is not quite meaningful by itself, they are often not selected by the system. The following segment is extracted from a dialogue about "universal health insurance". The two DAs from speaker B are not selected by our system but selected by human annotators, causing false negative errors.

B: and it just can devastate –
A: and your constantly, -
B: – your budget, you know.

6 Conclusion and Future Work

This paper investigates two unsupervised methods in opinion summarization on spontaneous conversations by incorporating topic score and sentiment score in existing summarization techniques. In the sentence-ranking method, we linearly combine several scores in different aspects to select sentences with the highest scores. In the graph-based method, we use an adjacency matrix to model the dialogue structure and utilize it to find salient utterances in conversations. Our experiments show that both methods are able to improve the baseline approach, and we find that the cosine similarity between utterances or between an utterance and the whole document is not as useful as in other document summarization tasks.

In future work, we will address some issues identified from our error analysis. First, we will investigate ways to represent a sentence's topic relevance. Second, we will evaluate using other extraction units, such as applying preprocessing to remove disfluencies and concatenate incomplete sentence segments together. In addition, it would be interesting to test our system on speech recognition output and automatically generated DA boundaries to see how robust it is.

7 Acknowledgments

The authors thank Julia Hirschberg and Ani Nenkova for useful discussions. This research is supported by NSF awards CNS-1059226 and IIS-0939966.

References

Alexandra Balahur, Ester Boldrini, Andrés Montoyo, and Patricio Martínez-Barco. 2010. *Going beyond traditional QA systems: challenges and keys in opinion question answering*. In *Proceedings of COLING*.

Günes Erkan and Dragomir R. Radev. 2004. *LexRank: graph-based lexical centrality as salience in text summarization*. *Journal of Artificial Intelligence Research*.

Sadaoki Furui, Tomonori Kikuchi, Yousuke Shinnaka, and Chior i Hori. 2004. Speech-to-text and speech-to-speech summarization of spontaneous speech. *IEEE Transactions on Audio, Speech & Language Processing*, 12(4):401–408.

Nikhil Garg, Benoit Favre, Korbinian Reidhammer, and Dilek Hakkani Tür. 2009. *ClusterRank: a graph based method for meeting summarization*. In *Proceedings of Interspeech*.

Dan Gillick, Korbinian Riedhammer, Benoit Favre, and Dilek Hakkani-Tur. 2009. *A global optimization framework for meeting summarization*. In *Proceedings of ICASSP*.

John J. Godfrey and Edward Holliman. 1997. *Switchboard-1 Release 2*. In *Linguistic Data Consortium, Philadelphia*.

Andrew Hayes and Klaus Krippendorff. 2007. *Answering the call for a standard reliability measure for coding data*. *Journal of Communication Methods and Measures*, 1:77–89.

Minqing Hu and Bing Liu. 2004. *Mining and summarizing customer reviews*. In *Proceedings of ACM SIGKDD*.

Konstantinos Koumpis and Steve Renals. 2005. *Automatic summarization of voicemail messages using lexical and prosodic features*. *ACM - Transactions on Speech and Language Processing*.

Shih Hsiang Lin, Berlin Chen, and Hsin min Wang. 2009. A comparative study of probabilistic ranking models for chinese spoken document summarization. *ACM Transactions on Asian Language Information Processing*, 8(1).

Chin-Yew Lin. 2004. *ROUGE: a package for automatic evaluation of summaries*. In *Proceedings of ACL workshop on Text Summarization Branches Out*.

Fei Liu and Yang Liu. 2008. *What are meeting summaries? An analysis of human extractive summaries in meeting corpus*. In *Proceedings of SIGDial*.

Sameer Maskey and Julia Hirschberg. 2005. *Comparing lexical, acoustic/prosodic, structural and discourse features for speech summarization*. In *Proceedings of Interspeech*.

Kathleen Mckeown, Julia Hirschberg, Michel Galley, and Sameer Maskey. 2005. *From text to speech summarization*. In *Proceedings of ICASSP*.

Gabriel Murray and Giuseppe Carenini. 2009. *Detecting subjectivity in multiparty speech*. In *Proceedings of Interspeech*.

Gabriel Murray, Steve Renals, and Jean Carletta. 2005. *Extractive summarization of meeting recordings*. In *Proceedings of EUROSPEECH*.

Vincent Ng, Sajib Dasgupta, and S.M.Niaz Arifin. 2006. *Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews*. In *Proceedings of the COLING/ACL*.

Hitoshi Nishikawa, Takaaki Hasegawa, Yoshihiro Matsuo, and Genichiro Kikui. 2010. *Opinion summarization with integer linear programming formulation for sentence extraction and ordering*. In *Proceedings of COLING*.

Bo Pang and Lilian Lee. 2004. *A sentiment education: sentiment analysis using subjectivity summarization based on minimum cuts*. In *Proceedings of ACL*.

Michael Paul, ChengXiang Zhai, and Roxana Girju. 2010. *Summarizing contrastive viewpoints in opinionated text*. In *Proceedings of EMNLP*.

Ana-Maria Popescu and Oren Etzioni. 2005. *Extracting product features and opinions from reviews*. In *Proceedings of HLT-EMNLP*.

Stephan Raaijmakers, Khiet Truong, and Theresa Wilson. 2008. *Multimodal subjectivity analysis of multiparty conversation*. In *Proceedings of EMNLP*.

Veselin Stoyanov, Claire Cardie, and Janyce Wiebe. 2005. *Multi-perspective question answering using the OpQA corpus*. In *Proceedings of EMNLP/HLT*.

Janyce Wiebe and Ellen Riloff. 2005. *Creating subjective and objective sentence classifiers from unannotated texts*. In *Proceedings of CICLing*.

Theresa Wilson and Janyce Wiebe. 2003. *Annotating opinions in the world press*. In *Proceedings of SIGDial*.

Theresa Wilson. 2008a. *Annotating subjective content in meetings*. In *Proceedings of LREC*.

Theresa Wilson. 2008b. Fine-grained subjectivity and sentiment analysis: recognizing the intensity, polarity, and attitudes of private states. Ph.D. thesis, University of Pittsburgh.

Shasha Xie and Yang Liu. 2010. Improving supervised learning for meeting summarization using sampling and regression. *Computer Speech and Language*, 24:495–514.

Klaus Zechner. 2002. Automatic summarization of open-domain multiparty dialogues in dive rse genres. *Computational Linguistics*, 28:447–485.

Justin Jian Zhang, Ho Yin Chan, and Pascale Fung. 2007. *Improving lecture speech summarization using rhetorical information*. In *Proceedings of Biannual IEEE Workshop on ASRU*.

Lin Zhao, Lide Wu, and Xuanjing Huang. 2009. *Using query expansion in graph-based approach for query-focused multi-document summarization*. *Journal of Information Processing and Management*.

Xiaodan Zhu and Gerald Penn. 2006. *Summarization of spontaneous conversations*. In *Proceedings of Interspeech*.

Contrasting Opposing Views of News Articles on Contentious Issues

Souneil Park[1], KyungSoon Lee[2], Junehwa Song[1]

[1]Korea Advanced Institute of Science and Technology
291 Daehak-ro, Yuseong-gu, Daejeon, Republic of Korea
{spark,junesong}@nclab.kaist.ac.kr

[2]Chonbuk National University
664-14 1ga Deokjin-dong Jeonju, Jeonbuk, Republic of Korea
selfsolee@chonbuk.ac.kr

Abstract

We present disputant relation-based method for classifying news articles on contentious issues. We observe that the disputants of a contention are an important feature for understanding the discourse. It performs unsupervised classification on news articles based on disputant relations, and helps readers intuitively view the articles through the opponent-based frame. The readers can attain balanced understanding on the contention, free from a specific biased view. We applied a modified version of HITS algorithm and an SVM classifier trained with pseudo-relevant data for article analysis.

1 Introduction

The coverage of contentious issues of a community is an essential function of journalism. Contentious issues continuously arise in various domains, such as politics, economy, environment; each issue involves diverse participants and their different complex arguments. However, news articles are frequently biased and fail to fairly deliver conflicting arguments of the issue. It is difficult for ordinary readers to analyze the conflicting arguments and understand the contention; they mostly perceive the issue passively, often through a single article. Advanced news delivery models are required to increase awareness on conflicting views.

In this paper, we present *disputant relation-based method* for classifying news articles on contentious issues. We observe that the disputants of a contention, i.e., people who take a position and participate in the contention such as politicians, companies, stakeholders, civic groups, experts, commentators, etc., are an important feature for understanding the discourse. News producers primarily shape an article on a contention by selecting and covering specific disputants (Baker. 1994). Readers also intuitively understand the contention by identifying who the opposing disputants are.

The method helps readers intuitively view the news articles through the *opponent-based frame*. It performs classification in an unsupervised manner: it dynamically identifies opposing disputant groups and classifies the articles according to their positions. As such, it effectively helps readers contrast articles of a contention and attain balanced understanding, free from specific biased viewpoints.

The proposed method differs from those used in related tasks as it aims to perform classification under the opponent-based frame. Research on sentiment classification and debate stance recognition takes a topic-oriented view, and attempts to perform classification under the 'positive vs. negative' or 'for vs. against' frame for the given topic, e.g., positive vs. negative about iPhone.

However, such frames are often not appropriate for classifying news articles of a contention. The coverage of a contention often spans over different topics (Miller. 2001). For the contention on the health care bill, an article may discuss the enlarged coverage whereas another may discuss the increase of insurance premiums. In addition, we observe that opposing arguments of a contention are often complex to classify under these frames. For exam-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 340–349,
Portland, Oregon, June 19-24, 2011.

ple, in a political contention on holding a referendum on the Sejong project[1], the opposition parties strongly opposed and criticized the president office. Meanwhile, the president office argued that they were not considering holding the referendum and the contention arose from a misunderstanding. In such a case, it is difficult to classify any argument to the "positive" category of the frame.

We demonstrate that the opponent-based frame is clear and effective for contrasting opposing views of contentious issues. For the contention on the referendum, 'president office vs. opposition parties' provides an intuitive frame to understand the contention. The frame does not require the documents to discuss common topics nor the opposing arguments to be positive vs. negative.

Under the proposed frame, it becomes important to analyze which side is more centrally covered in an article. Unlike debate posts or product reviews news articles, in general, do not take a position explicitly (except a few types such as editorials). They instead quote a specific side, elaborate them, and provide supportive facts. On the other hand, the opposing disputants compete for news coverage to influence more readers and gain support (Miller et al. 2001). Thus, the method focuses on identifying the disputants of each side and classifying the articles based on the side it covers.

We applied a modified version of HITS algorithm to identify the key opponents of an issue, and used disputant extraction techniques combined with an SVM classifier for article analysis. We observe that the method achieves acceptable performance for practical use with basic language resources and tools, i.e., Named Entity Recognizer (Lee et al. 2006), POS tagger (Shim et al. 2002), and a translated positive/negative lexicon. As we deal with non-English (Korean) news articles, it is difficult to obtain rich resources and tools, e.g., WordNet, dependency parser, annotated corpus such as MPQA. When applied to English, we believe the method could be further improved by adopting them.

2 Background and Related Work

Research has been made on sentiment classification in document-level (Turney et al., 2002, Pang et al., 2002, Seki et al. 2008, Ounis et al. 2006). It aims to automatically identify and classify the sentiment of documents into positive or negative. Opinion summarization aims a similar goal, to identify different opinions on a topic and generate summaries of them. Paul et al. (2010) developed an unsupervised method for generating summaries of contrastive opinions on a common topic. These works make a number of assumptions that are difficult to apply to the discourse of contentious news issues. They assume that the input documents have a common opinion target, e.g., a movie. Many of them primarily deal with documents which explicitly reveal opinions on the selected target, e.g., movie reviews. They usually apply one static classification frame, positive vs. negative, to the topic.

The discourse of contentious issues in news articles show different characteristics from that studied in the sentiment classification tasks. First, the opponents of a contentious issue often discuss different topics, as discussed in the example above. Research in mass communication has showed that opposing disputants talk across each other, not by dialogue, i.e., they martial different facts and interpretations rather than to give different answers to the same topics (Schon et al., 1994).

Second, the frame of argument is not fixed as 'positive vs. negative'. We frequently observed both sides of a contention articulating negative arguments attacking each other. The forms of arguments are also complex and diverse to classify them as positive or negative; for example, an argument may just neglect the opponent's argument without positive or negative expressions, or emphasize a different discussion point.

In addition, a position of a contention can be communicated without explicit expression of opinion or sentiment. It is often conveyed through objective sentences that include carefully selected facts. For example, a news article can cast a negative light on a government program simply by covering the increase of deficit caused by it.

A number of works deal with debate stance recognition, which is a closely related task. They attempt to identify a position of a debate, such as ideological (Somasundaran et al., 2010, Lin et al., 2006) or product comparison debate (Somasundaran et al., 2009). They assume a debate frame, which is similar to the frame of the sentiment classification task, i.e., for vs. against the debate topic. All articles of a debate in their corpus cover a coherent debate topic, e.g., iPhone vs. Blackberry, and explicitly express opinions for or

[1] http://www.koreatimes.co.kr/www/news/nation/2010/07/116_61649.html

against to the topic, e.g., for or against iPhone or Blackberry. The proposed methods assume that the debate frame is known apriori. This debate frame is often not appropriate for contentious issues for similar reasons as the positive/negative frame. In contrast, our method does not assume a fixed debate frame, and rather develops one based on the opponents of the contention at hand.

The news corpus is also different from the debate corpus. News articles of a contentious issue are more diverse than debate articles conveying explicit argument of a specific side. There are news articles which cover both sides, facts without explicit opinions, and different topics unrelated to the arguments of either side.

Several works have used the relation between speakers or authors for classifying their debate stance (Thomas et al., 2006, Agrawal et al., 2003). However, these works also assume the same debate frame and use the debate corpus, e.g., floor debates in the House of Representatives, online debate forums. Their approaches are also supervised, and require training data for relation analysis, e.g., voting records of congresspeople.

3 Argument Frame Comparison

Establishing an appropriate argument frame is important. It provides a framework which enable readers to intuitively understand the contention. It also determines how classification methods should classify articles of the issue.

We conducted a user study to compare the opponent-based frame and the positive (for) vs. negative (against) frame. In the experiment, multiple human annotators classified the same set of news articles under each of the two frames. We compared which frame is clearer for the classification, and more effective for exposing opposing views.

We selected 14 contentious issues from Naver News (a popular news portal in Korea) issue archive. We randomly sampled about 20 articles per each issue, for a total of 250 articles. The selected issues range over diverse domains such as politics, local, diplomacy, economy; to name a few for example, the contention on the 4 river project, of which the key opponents are the government vs. catholic church; the entrance of big retailers to the supermarket business, of which the key opponents are the small store owners vs. big retail companies; the refusal to approve an integrated civil servants' union, of which the key opponents are government vs. Korean government employees' union.

We use an internationally known contention, i.e., the dispute about the Cheonan sinking incident, as an example to give more details on the disputants. Our data set includes 25 articles that were published after the South Korea's announcement of their investigation result. Many disputants appear in the articles, e.g., South Korean Government, South Korea defense secretary, North Korean Government, United States officials, Chinese experts, political parties of South Korea, etc.

Three annotators performed the classification. All of them were students. For impartiality, two of them were recruited from outside the team, who were not aware of this research.

The annotators performed two subtasks for classification. As for the positive vs. negative frame, first, we asked them to designate the main topic of the contention. Second, they classified the articles which mainly deliver arguments for the topic to the "positive" category and those delivering arguments against the topic to the "negative" category. The articles are classified to the "Other" category if they do not deal with the main topic nor cover positive or negative arguments.

As for the opponent-based frame, first, we asked them to designate the competing opponents. Second, we asked to classify articles to a specific side if the articles cover only the positions, arguments, or information supportive of that side or if they cover information detrimental or criticism to its opposite side. Other articles were classified to the "Other" category. Examples of this category include articles covering both sides fairly, describing general background or implications of the issue.

Issue #	Free-marginal kappa		Issue #	Free-marginal kappa	
	Pos.-Neg.	Opponent		Pos.-Neg.	Opponent
1	0.83	0.67	**8**	0.26	0.58
2	0.57	0.48	**9**	0.07	1.00
3	0.44	0.95	**10**	0.48	0.84
4	0.75	0.87	**11**	0.71	0.86
5	0.36	0.64	**12**	0.71	0.71
6	0.30	0.70	**13**	0.63	0.79
7	0.18	0.96	**14**	0.48	0.87
			Avg.	0.50	0.78

Table 1. Inter-rater agreement result.

The agreement in classification was higher for the opponent-based frame in most issues. This indicates that the annotators could apply the frame more clearly, resulting in smaller difference between them. The kappa measure was 0.78 on aver-

age. The kappa measure near 0.8 indicates a substantial level of agreement, and the value can be achieved, for example, when 8 or 9 out of 10 items are annotated equally (Table 1).

In addition, fewer articles were classified to the "Other" category under the opponent-based frame. The annotators classified about half of the articles to this category under the positive vs. negative frame whereas they classified about 35% to the category under the opponent-based frame. This is because the frame is more flexible to classify diverse articles of an issue, such as those covering arguments on different points, and those covering detrimental facts to a specific side without explicit positive or negative arguments.

The kappa measure was less than 0.5 for near half of the issues under the positive-negative frame. The agreement was low especially when the main topic of the contention was interpreted differently among the annotators; the main topic was interpreted differently for issue 3, 7, 8, and 9. Even when the topic was interpreted identically, the annotators were confused in judging complex arguments either as positive or negative. One annotator commented that "it was confusing as the arguments were not clearly for or against the topic often. Even when a disputant was assumed to have a positive attitude towards the topic, the disputant's main argument was not about the topic but about attacking the opponent" The annotators all agreed that the opponent-based frame is more effective to understand the contention.

4 Disputant relation-based method

Disputant relation-based method adopts the opponent-based frame for classification. It attempts to identify the two opposing groups of the issue at hand, and analyzes whether an article more reflects the position of a specific side. The method is based on the observation that there exists two opposing groups of disputants, and the groups compete for news coverage. They strive to influence readers' interpretation, evaluation of the issue and gain support from them (Miller et al. 2001). In this competing process, news articles may give more chance of speaking to a specific side, explain or elaborate them, or provide supportive facts of that side (Baker 1994).

The proposed method is performed in three stages: the first stage, disputant extraction, extracts the disputants appearing in an article set; the second stage, disputant partition, partitions the extracted disputants into two opposing groups; lastly, the news classification stage classifies the articles into three categories, i.e., two for the articles biased to each group, and one for the others.

4.1 Disputant Extraction

In this stage, the disputants who participate in the contention have to be extracted. We utilize that many disputants appear as the subject of quotes in the news article set. The articles actively quote or cover their action in order to deliver the contention lively. We used straight forward methods for extraction of subjects. The methods were effective in practice as quotes of articles frequently had a regular pattern.

The subjects of direct and indirect quotes are extracted. The sentences including an utterance inside double quotes are considered as direct quotes.

The sentences which convey an utterance without double quotes, and those describing the action of a disputant are considered as indirect quotes (See the translated example 1 below). The indirect quotes are identified based on the morphology of the ending word. The ending word of the indirect quotes frequently has a verb as its root or includes a verbalization suffix. Other sentences, typically, those describing the reporter's interpretation or comments are not considered as quotes. (See example sentence 2. The ending word of the original sentence is written in boldface).

(1) The *government* **clarified** that there won't be any talks unless *North Korea* apologizes for the attack.

(2) The *government*'s **belief** is that a stern response is the only solution for the current crisis

A named entity combined with a topic particle or a subject particle is identified as the subject of these quotes. We detect the name of an organization, person, or country using the Korean Named Entity Recognizer (Lee et al. 2006). A simple anaphora resolution is conducted to identify subjects also from abbreviated references or pronouns in subsequent quotes.

4.2 Disputant Partitioning

We develop key opponent-based partitioning method for disputant partitioning. The method first identifies two key opponents, each representing

one side, and uses them as a pivot for partitioning other disputants. The other disputants are divided according to their relation with the key opponents, i.e., which key opponent they stand for or against.

The intuition behind the method is that there usually exists key opponents who represent the contention, and many participants argue about the key opponents whereas they seldom recognize and talk about minor disputants. For instance, in the contention on "investigation result of the Cheonan sinking incident", the government of North Korea and that of South Korea are the key opponents; other disputants, such as politicians, experts, civic group of South Korea, the government of U.S., and that of China, mostly speak about the key opponents. Thus, it is effective to analyze where the disputants stand regarding their attitude toward the key opponents.

Selecting key opponents: In order to identify the key opponents of the issue, we search for the disputants who frequently criticize, and are also criticized by other disputants. As the key opponents get more news coverage, they have more chance to articulate their argument, and also have more chance to face counter-arguments by other disputants.

This is done in two steps. First, for each disputant, we analyze whom he or she criticizes and by whom he or she is criticized. The method goes through each sentence of the article set and searches for both disputant's criticisms and the criticisms about the disputant. Based on the criticisms, it analyzes relationships among disputants.

A sentence is considered to express the disputant's criticism to another disputant if the following holds: 1) the sentence is a quote, 2) the disputant is the subject of the quote, 3) another disputant appears in the quote, and 4) a negative lexicon appears in the sentence.

On the other hand, if the disputant is not the subject but appears in the quote, the sentence is considered to express a criticism about the disputant made by another disputant (See example 3. The disputants are written in italic, and negative words are in boldface.).

(3) the *government* defined that "the **attack** of *North Korea* is an act of **invasion** and also a **violation** of North-South Basic Agreement"

The negative lexicon we use is carefully built from the Wilson lexicon (Wilson et al. 2005). We translated all the terms in it using the Google translation, and manually inspected the translated result to filter out inappropriate translations and the terms that are not negative in the Korean context.

Second, we apply an adapted version of HITS graph algorithm to find major disputants. For this, the criticizing relationships obtained in the first step are represented in a graph. Each disputant is modeled as a node, and a link is made from a criticizing disputant to a criticized disputant.

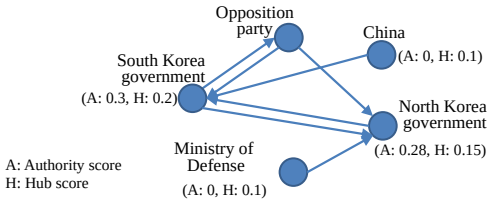


Figure 1. Example HITS graph illustration

Originally, the HITS algorithm (Kleinberg, 1999) is designed to rate Web pages regarding the link structure. The feature of the algorithm is that it separately models the value of outlinks and inlinks. Each node, i.e., a web page, has two scores: the authority score, which reflects the value of inlinks toward itself, and the hub score, which reflects the value of its outlinks to others. The hub score of a node increases if it links to nodes with high authority score, and the authority score increases if it is pointed by many nodes with high hub score.

We adopt the HITS algorithm due to above feature. It enables us to separately measure the significance of a disputant's criticism (using the hub score) and the criticism about the disputant (using the authority score). We aim to find the nodes which have both high hub score and high authority score; the key opponents will have many links to others and also be pointed by many nodes.

The modified HITS algorithm is shown in Figure 2. We make some adaptation to make the algorithm reflect the disputants' characteristics. The initial hub score of a node is set to the number of quotes in which the corresponding disputant is the subject. The initial authority score is set to the number of quotes in which the disputant appears but not as the subject. In addition, the weight of each link (from a criticizing disputant to a criticized disputant) is set to the number of sentences that express such criticism.

We select the nodes which show relatively high hub score and high authority score compared to other nodes. We rank the nodes according to the sum of hub and authority scores, and select from

the top ranking node. The node is not selected if its hub or authority score is zero. The selection is finished if more than two nodes are selected and the sum of hub and authority scores is less than half of the sum of the previously selected node.

Modified HITS(G,W,k)

$G = <V, E>$ where
V is a set of vertex, a vertex v_i represents a disputant
E is a set of edges, an edge e_{ij} represents a criticizing quote from disputant i to j
$W = \{w_{ij} |$ weight of edge $e_{ij}\}$

For all $v_i \in V$
$Auth^1(v_i)$ = # of quotes of which the subject is disputant i
$Hub^1(v_i)$ = # of quotes of which disputant i appears, but not as the subject

For $t = 1$ to k:
$Auth^{t+1}(v_i) = \sum_{j:\, e_{ji} \in E} Hub^t(v_j) \times w_{ji}$
$Hub^{t+1}(v_i) = \sum_{j:\, e_{ij} \in E} Auth^t(v_j) \times w_{ij}$
Normalize $Auth^{t+1}(v_i)$ and $Hub^{t+1}(v_i)$

Figure 2. Algorithm of the Modified HITS

More than two disputants can be selected if more than one disputant is active from a specific side. In such cases, we choose the two disputants whose criticizing relationship is the strongest among the selected ones, i.e., the two who show the highest ratio of criticism between them.

Partitioning minor disputants: Given the two key opponents, we partition the rest of disputants based on their relations with the key opponents. For this, we identify whether each disputant has positive or negative relations with the key opponents. The disputant is classified to the side of the key opponent who shows more positive relations. If the disputant shows more negative relations, the disputant is classified to the opposite side.

We analyze the relationship not only from the article set but also from the web news search results. The minor disputants may not be covered importantly in the article set; hence, it can be difficult to obtain sufficient data for analysis. The web news search results provide supplementary data for the analysis of relationships.

We develop four features to capture the positive and negative relationships between the disputants.

1) Positive Quote Rate (PQR_{ab}): Given two disputants (a key opponent *a*, and a minor disputant *b*), the feature measures the ratio of positive quotes between them. A sentence is considered as a positive quote if the following conditions hold: the sentence is a direct or indirect quote, the two disputants appear in the sentence, one is the subject of the quote, and a positive lexicon appears in the sentence. The number of such sentences is divided by the number of all quotes in which the two disputants appear and one appears as the subject.

2) Negative Quote Rate (NQR_{ab}): This feature is an opposite version of PQR. It measures the ratio of negative quotes between the two disputants. The same conditions are considered to detect negative quotes except that negative lexicon is used instead of positive lexicon.

3) Frequency of Standing Together (FST_{ab}): This feature attempts to capture whether the two disputants share a position, e.g., "*South Korea* and *U.S.* both criticized North Korea for…" It counts how many times they are co-located or connected with the conjunction "and" in the sentences.

4) Frequency of Division (FD_{ab}): This feature is an opposite version of the FST. It counts how many times they are not co-located in the sentences.

The same features are also calculated from the web news search results; we collect news articles of which the title includes the two disputants, i.e., a key opponent *a* and a minor disputant *b*.

The calculation method of PQR and NQR is slightly adapted since the titles are mostly not complete sentences. For PQR (NQR), it counts the titles which the two disputants appear with a positive (negative) lexicon. The counted number is divided by the number of total search results. The calculation method of FST and FD is the same except that they are calculated from the titles.

We combine the features obtained from web news search with the corresponding ones obtained from the article set by calculating a weighted sum. We currently give equal weights.

The disputants are partitioned by the following rule: given a minor disputant *a*, and the two key opponents *b* and *c*,

classify *a* to *b*'s side if,

$(PQR_{ab} - NQR_{ab}) > (PQR_{ac} - NQR_{ac})$ or $((FST_{ab} > FD_{ab})$ and $(FST_{ac} = 0))$;

classify *a* to *c*'s side if,

$(PQR_{ac} - NQR_{ac}) > (PQR_{ab} - NQR_{ab})$ or $((FST_{ac} > FD_{ac})$ and $(FST_{ab} = 0))$;

classify *a* to *other*, otherwise.

4.3 Article Classification

Each news article of the set is classified by analyzing which side is importantly covered. The method classifies the articles into three categories, either to one of the two sides or the category "other".

We observed that the major components which shape an article on a contention are quotes from disputants and journalists' commentary. Thus, our method considers two points for classification: first, from which side the article's quotes came; second, for the rest of the article's text, the similarity of the text to the arguments of each side.

As for the quotes of an article, the method calculates the proportion of the quotes from each side based on the disputant partitioning result. As for the rest of the sentences, a similarity analysis is conducted with an SVM classifier. The classifier takes a sentence as input, determines its class to one of the three categories, i.e., one of the two sides, or other. It is trained with the quotes from each side (tf.idf of unigram and bigram is used as features). The same number of quotes from each side is used for training. The training data is pseudo-relevant: it is automatically obtained based on the partitioning result of the previous stage.

An article is classified to a specific side if more of its quotes are from that side and more sentences are similar to that side: given an article *a*, and the two sides *b* and *c*,

classify *a* to *b* if $\frac{Q_b + S_b}{S_U} \geq \frac{\alpha \cdot Q_{bc} + \beta \cdot S_{bc}}{S_U}$

classify *a* to *c* if $\frac{Q_c + S_c}{S_U} \geq \frac{\alpha \cdot Q_{bc} + \beta \cdot S_{bc}}{S_U}$

classify *a* to *other*, otherwise.

where S_U: number of all sentences of the article
Q_i: number of quotes from the side *i*.
Q_{ij}: number of quotes from either side *i* or *j*.
S_i: number of sentences classified to *i* by SVM.
S_{ij}: number of sentences classified to either *i* or *j*.

We currently set the parameters heuristically. We set 0.7 and 0.6 for the two parameters α and β respectively. Thus, for an article written purely with quotes, the article is classified to a specific side if more than 70% of the quotes are from that side. On the other hand, for an article which does not include quotes from any side, more than 60% of the sentences have to be determined similar to a specific side's quotes. We set a lower value for β to classify articles with less number of biased sentences (Articles often include non-quote sentences unrelated to any side to give basic information).

5 Evaluation and Discussion

Our evaluation of the method is twofold: first, we evaluate the disputant partitioning results, second, the accuracy of classification. The method was evaluated using the same data set used for the classification frame comparison experiment.

A gold result was created through the three human annotators. To evaluate the disputant partitioning results, we had the annotators to extract the disputants of each issue, divide them into opposing two groups. We then created a gold partitioning result, by taking a union of the three annotators' results. A gold classification is also created from the classification of the annotators. We resolved the disagreements between the annotators' results by following the decision of the majority.

5.1 Evaluation of Disputant Partitioning

We evaluated the partitioning result of the two opposing groups, denoted as G1 and G2. The performance is measured using precision and recall. Table 2 presents the results. The precision of the partitioning was about 70% on average. The false positives were mostly the disputants who appear only a few times both in the article set and the news search results. As they appeared rarely, there was not enough data to infer their position. The effect of these false positives in article classification was limited.

The recall was slightly lower than precision. This was mainly because some disputants were omitted in the disputant extraction stage. The NER we used occasionally missed the names of unpopular organizations, e.g., civic groups, and the extraction rule failed to capture the subject in some complex sentences. However, most disputants who frequently appear in the article set were extracted and partitioned appropriately.

Issue #	G1		G2		Issue #	G1		G2	
	Pr.	Re.	Pr.	Re.		Pr.	Re.	Pr.	Re.
1	0.45	0.63	0.63	0.47	**8**	0.80	1.00	1.00	0.40
2	0.75	0.36	0.82	0.70	**9**	1.00	1.00	1.00	0.67
3	0.80	0.47	0.50	0.50	**10**	0.67	0.63	0.50	0.75
4	0.75	0.53	0.80	0.60	**11**	0.67	0.33	0.63	0.36
5	0.75	0.55	0.44	0.67	**12**	0.83	0.38	0.63	0.63
6	0.50	0.33	0.33	0.50	**13**	0.71	0.56	0.50	0.33
7	0.67	0.80	1.00	0.57	**14**	0.55	0.75	0.56	0.45
					Avg.	0.71	0.59	0.67	0.54

Table 2. Disputant Partitioning Result

5.2 Evaluation of Article Classification

We evaluate our method and compare it with two unsupervised methods below.

Similarity-based clustering (Sim.): The method implements a typical method. It clusters articles of an issue into three groups based on text similari

Issue #	Total	G1	G2	Other
1	24	9	9	6
2	23	11	4	8
3	18	2	10	6
4	25	9	12	4
5	18	5	9	4
6	10	0	5	5
7	22	4	11	7
8	10	3	3	4
9	13	0	3	10
10	15	5	7	3
11	15	6	5	4
12	13	2	3	8
13	19	12	5	2
14	25	13	10	2

Issue #	Method	wF	Group 1			Group 2			Other		
			F	P	R	F	P	R	F	P	R
1	DrC	0.47	0.64	0.47	1.00	0.62	1.00	0.44	N/A	0.00	0.00
	QbC	**0.50**	0.62	0.47	0.89	0.71	1.00	0.55	N/A	0.00	0.00
	Sim.	0.27	0.20	1.00	0.11	0.20	1.00	0.11	0.47	0.30	1.00
2	DrC	**0.65**	0.67	0.62	0.73	0.86	1.00	0.75	0.53	0.57	0.50
	QbC	**0.65**	0.76	0.80	0.73	0.60	0.50	0.75	0.53	0.57	0.50
	Sim.	0.37	0.63	0.48	0.91	N/A	0.00	0.00	0.22	1.00	0.13
3	DrC	0.72	0.57	0.40	1.00	0.67	1.00	0.50	0.86	0.75	1.00
	QbC	**0.74**	0.57	0.40	1.00	0.75	1.00	0.60	0.77	0.71	0.83
	Sim.	0.59	N/A	0.00	0.00	0.70	0.62	0.80	0.60	0.75	0.50
4	DrC	0.80	0.82	0.69	1.00	0.86	1.00	0.75	0.57	0.67	0.50
	QbC	**0.81**	0.90	0.82	1.00	0.86	1.00	0.75	0.44	0.40	0.50
	Sim.	0.67	0.80	1.00	0.67	0.80	0.67	1.00	N/A	0.00	0.00
5	DrC	**0.60**	0.63	0.50	0.83	0.71	0.83	0.63	0.33	0.50	0.25
	QbC	0.55	0.40	0.50	0.33	0.71	0.67	0.75	0.44	0.40	0.50
	Sim.	0.51	0.63	0.46	1.00	0.67	1.00	0.50	N/A	0.00	0.00
6	DrC	**0.89**	N/A	0.00	N/A	0.89	1.00	0.80	0.89	1.00	0.80
	QbC	0.50	N/A	0.00	N/A	0.50	0.67	0.40	0.50	0.67	0.40
	Sim.	0.55	N/A	0.00	N/A	0.77	0.63	1.00	0.33	1.00	0.20
7	DrC	**0.48**	0.67	1.00	0.50	0.71	0.55	1.00	N/A	N/A	0.00
	QbC	**0.48**	0.67	1.00	0.50	0.62	0.53	0.73	0.17	0.20	0.14
	Sim.	0.44	0.40	0.27	0.75	0.57	0.60	0.55	0.25	1.00	0.14
8	DrC	**0.90**	0.86	0.75	1.00	1.00	1.00	1.00	0.86	1.00	0.75
	QbC	0.48	0.57	0.50	0.67	0.57	0.50	0.67	0.33	0.50	0.25
	Sim.	0.56	0.67	0.67	0.67	0.50	0.40	0.67	0.50	1.00	0.33
9	DrC	**0.77**	N/A	0.00	N/A	0.57	0.50	0.67	0.82	1.00	0.70
	QbC	0.79	N/A	0.00	N/A	0.67	0.67	0.67	0.82	1.00	0.70
	Sim.	0.49	N/A	0.00	N/A	0.00	0.00	0.00	0.63	0.67	0.60
10	DrC	0.66	0.71	0.56	1.00	0.73	1.00	0.57	0.40	0.50	0.33
	QbC	**0.72**	0.77	0.63	1.00	0.77	0.83	0.71	0.50	1.00	0.33
	Sim.	0.40	0.33	1.00	0.20	0.44	1.00	0.29	0.40	0.25	1.00
11	DrC	**0.61**	0.73	0.80	0.67	0.50	0.43	0.60	0.57	0.67	0.50
	QbC	0.39	0.62	0.57	0.67	0.20	0.20	0.20	0.29	0.33	0.25
	Sim.	0.47	0.63	0.46	1.00	0.33	1.00	0.20	0.40	1.00	0.25
12	DrC	**0.67**	0.29	0.20	0.50	0.67	0.67	0.67	0.77	1.00	0.63
	QbC	0.38	0.33	0.25	0.50	0.44	0.33	0.67	0.36	0.47	0.25
	Sim.	0.43	N/A	0.00	0.00	0.55	0.38	1.00	0.50	0.75	0.38
13	DrC	**0.65**	0.79	0.69	0.92	0.33	1.00	0.20	0.67	1.00	0.50
	QbC	0.59	0.75	0.75	0.75	0.33	1.00	0.20	0.29	0.20	0.50
	Sim.	0.54	0.71	0.63	0.83	0.33	1.00	0.20	N/A	0.00	0.00
14	DrC	0.61	0.77	0.77	0.77	0.50	0.57	0.44	0.25	0.20	0.33
	QbC	**0.66**	0.83	0.75	0.92	0.53	0.67	0.44	0.33	0.33	0.33
	Sim.	0.37	0.29	1.00	0.17	0.60	0.43	1.00	N/A	0.00	0.00

*N/A: The metric could not be calculated in some cases. This happened when no articles were classified to a category.

Table 3. Number of articles of each issue and group (left), and classification performance (right)

ty. It uses tf.idf of unigram and bigram as features, and cosine similarity as the similarity measure. We used the K-means clustering algorithm.

Quote-based classification (QbC.): The method is a partial implementation of our method. The disputant extraction and disputant partitioning is performed identically; however, it classifies news articles merely based on quotes. An article is classified to one of the two opposing sides if more than 70% of the quotes are from that side, or to the "other" category otherwise.

Results: We evaluated the classification result of the three categories, the two groups G1 and G2, and the category Other. The performance is measured using precision, recall, and f-measure. We additionally used the weighted f-measure (wF) to aggregate the f-measure of the three categories. It is the weighted average of the three f-measures. The weight is proportional to the number of articles in each category of the gold result.

The disputant relation-based method (DrC) performed better than the two comparison methods. The overall average of the weighted f-measure among issues was 0.68, 0.59, and 0.48 for the DrC, QbC, and Sim. method, respectively (See Table 3). The performance of the similarity-based clustering was lower than that of the other two in most issues.

A number of works have reported that text similarity is reliable in stance classification in political domains. These experiments were conducted in political debate corpus (Lin et al. 2006). However, news article set includes a number of articles covering different topics irrelevant to the arguments of the disputants. For example, there can be an article describing general background of the contention. Similarity-based clustering approach reacted sensitively to such articles and failed to capture the difference of the covered side.

Quote-based classification performs better than similarity-based approach as it classifies articles primarily based on the quoted disputants. The performance is comparable to DrC in many issues. The method performs similarly to DrC if most articles of an issue include many qutes. DrC performs better for other issues which include a number of articles with only a few quotes.

Error analysis: As for our method, we observed three main reasons of misclassification.

1) Articles with few quotes: Although the proposed method better classifies such articles than the quote-based classification, there were some misclassifications. There are sentences that are not directly related to the argument of any side, e.g., plain description of an event, summarizing the development of the issue, etc. The method made errors while trying to decide to which side these sentences are close to. Detecting such sentences and avoiding decisions for them would be one way of improvement. Research on classification

of subjective and objective sentences would be helpful (Wiebe et al. 99).
2) Article criticizing the quoted disputants: There were some articles criticizing the quoted disputants. For example, an article quoted the president frequently but occasionally criticized him between the quotes. The method misclassified such articles as it interpreted that the article is mainly delivering the president's argument.
3) Errors in disputant partitioning: Some misclassifications were made due to the errors in the disputant partitioning stage, specifically, those who were classified to a wrong side. Articles which refer to such disputants many times were misclassified.

6 Conclusion

We study the problem of classifying news articles on contentious issues. It involves new challenges as the discourse of contentious issues is complex, and news articles show different characteristics from commonly studied corpus, such as product reviews. We propose opponent-based frame, and demonstrate that it is a clear and effective classification frame to contrast arguments of contentious issues. We develop disputant relation-based classification and show that the method outperforms a text similarity-based approach.

Our method assumes polarization for contentious issues. This assumption was valid for most of the tested issues. For a few issues, there were some participants who do not belong to either side; however, they usually did not take a particular position nor make strong arguments. Thus, the effect on classification performance was limited. Discovering and developing methods for issues which involve more than two disputants groups is a future work.

References

Rakesh Agrawal, Sridhar Rajagopalan, Ramakrishnan Srikant, and Yirong Xu. 2003. Mining newsgroups using networks arising from social behavior. In *Proceedings of WWW*.

Baker, B. 1994. How to Identify, Expose and Correct Liberal Media Bias. Media Research Center.

Mohit Bansal, Claire Cardie, and Lillian Lee. 2008. The power of negative thinking: Exploiting label disagreement in the min-cut classification framework. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-2008)*.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. In Journal of ACM, 46(5): 604-632.

Landis JR, Koch G. 1977. The measurement of observer agreement for categorical data. *Biometrics* 33:159-174.

Changki Lee, Yi-Gyu Hwang, Hyo-Jung Oh, Soojong Lim, Jeong Heo, Chung-Hee Lee, Hyeon-Jin Kim, Ji-Hyun Wang, Myung-Gil Jang. 2006. Fine-Grained Named Entity Recognition using Conditional Random Fields for Question Answering, In *Proceedings of Human & Cognitive Language Technology (HCLT)*, pp. 268~272. (in Korean)

Wei-Hao Lin, Theresa Wilson, Janyce Wiebe, and Alexander Hauptmann. 2006. Which side are you on? Identifying perspectives at the document and sentence levels. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-2006)*, pages 109–116, New York.

Mark M. Miller and Bonnie P. Riechert. 2001. Spiral Opportunity and Frame Resonance: Mapping Issue Cycle in News and Public Discourse. In *Framing Public Life: Perspectives on Media and our Understanding of the Social World*, NJ: Lawrence Erlbaum Associates.

I Ounis, M de Rijke, C Macdonald, G Mishne, and I Soboroff. 2006. Overview of the TREC-2006 Blog Track. In *Proceedings of TREC*.

Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques, *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing* (EMNLP).

Paul, M. J., Zhai, C., Girju, R. 2010. Summarizing Contrastive Viewpoints in Opinionated Text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (EMNLP).

Schon, D.A., and Rien, M. 1994. Frame reflection: Toward the resolution of intractable policy controversies. New York: Basic Books.

Y. Seki, D. Evans, L. Ku, L. Sun, H. Chen, and N. Kando. 2008. Overview of Multilingual Opinion Analysis Task at NTCIR-7. In *Proceedings of 7th NTCIR Evaluation Workshop*, pages 185-203

Kwangseob Shim and Jaehyung Yang. 2002. MACH : A Supersonic Korean Morphological Analyzer, *Proceedings of the 19th International Conference on Computational Linguistics* (*COLING-2002*), pp.939-945.

Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 226–234, Suntec, Singapore, August. Association for Computational Linguistics.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological online debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text (CAAGET '10)*.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get outthe vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 327–335, Sydney, Australia, July. Association for Computational Linguistics.

Turney, Peter D. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews, *Proceedings of ACL-02*, Philadelphia, Pennsylvania, 417-424

Wiebe, Janyce M., Bruce, Rebecca F., & O'Hara, Thomas P. 1999. Development and use of a gold standard data set for subjectivity classifications. *In Proc. 37th Annual Meeting of the Assoc. for Computational Linguistics (ACL-99)*. June, pp. 246-253.

T. Wilson, J. Wiebe, and P. Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (EMNLP).

Content Models with Attitude

Christina Sauper, Aria Haghighi, Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
csauper@csail.mit.edu, me@aria42.com, regina@csail.mit.edu

Abstract

We present a probabilistic topic model for jointly identifying properties and attributes of social media review snippets. Our model simultaneously learns a set of properties of a product and captures aggregate user sentiments towards these properties. This approach directly enables discovery of highly rated or inconsistent properties of a product. Our model admits an efficient variational mean-field inference algorithm which can be parallelized and run on large snippet collections. We evaluate our model on a large corpus of snippets from Yelp reviews to assess property and attribute prediction. We demonstrate that it outperforms applicable baselines by a considerable margin.

Coherent property cluster

+	The **martinis** were very good.
	The **drinks** - both **wine** and **martinis** - were tasty.
-	The **wine list** was pricey.
	Their **wine selection** is horrible.

Incoherent property cluster

+	The sushi is the **best I've ever had**.
	Best paella **I'd ever had**.
	The fillet was the **best steak we'd ever had**.
	It's the **best soup I've ever had**.

Table 1: Example clusters of restaurant review snippets. The first cluster represents a coherent *property* of the underlying product, namely the *cocktail* property, and assesses distinctions in user sentiment. The latter cluster simply shares a common attribute expression and does not represent snippets discussing the same product property. In this work, we aim to produce the first type of property cluster with correct sentiment labeling.

1 Introduction

Online product reviews have become an increasingly valuable and influential source of information for consumers. Different reviewers may choose to comment on different properties or aspects of a product; therefore their reviews focus on different qualities of the product. Even when they discuss the same properties, their experiences and, subsequently, evaluations of the product can differ dramatically. Thus, information in any single review may not provide a complete and balanced view representative of the product as a whole. To address this need, online retailers often use simple aggregation mechanisms to represent the spectrum of user sentiment. For instance, product pages on Amazon prominently display the distribution of numerical scores across reviews, providing access to reviews at different levels of satisfaction.

The goal of our work is to provide a mechanism for review content aggregation that goes beyond numerical scores. Specifically, we are interested in identifying fine-grained product properties across reviews (e.g., *battery life* for electronics or *pizza* for restaurants) as well as capturing attributes of these properties, namely aggregate user sentiment.

For this task, we assume as input a set of product review snippets (i.e., standalone phrases such as "battery life is the best I've found") rather than complete reviews. There are many techniques for extracting this type of snippet in existing work; we use the Sauper et al. (2010) system.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 350–358,
Portland, Oregon, June 19-24, 2011.

At first glance, this task can be solved using existing methods for review analysis. These methods can effectively extract product properties from individual snippets along with their corresponding sentiment. While the resulting property-attribute pairs form a useful abstraction for cross-review analysis, in practice direct comparison of these pairs is challenging.

Consider, for instance, the two clusters of restaurant review snippets shown in Figure 1. While both clusters have many words in common among their members, only the first describes a coherent property cluster, namely the *cocktail* property. The snippets of the latter cluster do not discuss a single product property, but instead share similar expressions of sentiment. To solve this issue, we need a method which can correctly identify both property and sentiment words.

In this work, we propose an approach that jointly analyzes the whole collection of product review snippets, induces a set of learned properties, and models the aggregate user sentiment towards these properties. We capture this idea using a Bayesian topic model where a set of properties and corresponding attribute tendencies are represented as hidden variables. The model takes product review snippets as input and explains how the observed text arises from the latent variables, thereby connecting text fragments with corresponding properties and attributes.

The advantages of this formulation are twofold. First, this encoding provides a common ground for comparing and aggregating review content in the presence of varied lexical realizations. For instance, this representation allows us to directly compare how many reviewers liked a given property of a product. Second, our model yields an efficient mean-field variational inference procedure which can be parallelized and run on a large number of review snippets.

We evaluate our approach in the domain of snippets taken from restaurant reviews on Yelp. In this collection, each restaurant has on average 29.8 snippets representing a wide spectrum of opinions about a restaurant. The evaluation we present demonstrates that the model can accurately retrieve clusters of review fragments that describe the same property, yielding 20% error reduction over a standalone clustering baseline. We also show that the model can effectively identify binary snippet attributes with 9.2% error reduction over applicable baselines, demonstrating that learning to identify attributes in the context of other product reviews yields significant gains. Finally, we evaluate our model on its ability to identify product properties for which there is significant sentiment disagreement amongst user snippets. This tests our model's capacity to jointly identify properties and assess attributes.

2 Related Work

Our work on review aggregation has connections to three lines of work in text analysis.

First, our work relates to research on extraction of product properties with associated sentiment from review text (Hu and Liu, 2004; Liu et al., 2005a; Popescu et al., 2005). These methods identify relevant information in a document using a wide range of methods such as association mining (Hu and Liu, 2004), relaxation labeling (Popescu et al., 2005) and supervised learning (Kim and Hovy, 2006). While our method also extracts product properties and sentiment, our focus is on multi-review aggregation. This task introduces new challenges which were not addressed in prior research that focused on per-document analysis.

A second related line of research is multi-document review summarization. Some of these methods directly apply existing domain-independent summarization methods (Seki et al., 2006), while others propose new methods targeted for opinion text (Liu et al., 2005b; Carenini et al., 2006; Hu and Liu, 2006; Kim and Zhai, 2009). For instance, these summaries may present contrastive view points (Kim and Zhai, 2009) or relay average sentiment (Carenini et al., 2006). The focus of this line of work is on how to select suitable sentences, assuming that relevant review features (such as numerical scores) are given. Since our emphasis is on multi-review analysis, we believe that the information we extract can benefit existing summarization systems.

Finally, a number of approaches analyze review documents using probabilistic topic models (Lu and Zhai, 2008; Titov and McDonald, 2008; Mei et al., 2007). While some of these methods focus primar-

ily on modeling ratable aspects (Titov and McDonald, 2008), others explicitly capture the mixture of topics and sentiments (Mei et al., 2007). These approaches are capable of identifying latent topics in the collection in opinion text (e.g., weblogs) as well as associated sentiment. While our model captures similar high-level intuition, it analyzes fine-grained properties expressed at the snippet level, rather than document-level sentiment. Delivering analysis at such a fine granularity requires a new technique.

3 Problem Formulation

In this section, we discuss the core random variables and abstractions of our model. We describe the generative models over these elements in Section 4.

Product: A product represents a reviewable object. For the experiments in this paper, we use restaurants as products.

Snippets: A snippet is a user-generated short sequence of tokens describing a product. Input snippets are deterministically taken from the output of the Sauper et al. (2010) system.

Property: A property corresponds to some fine-grained aspect of a product. For instance, the snippet "the pad thai was great" describes the *pad thai* property. We assume that each snippet has a single property associated with it. We assume a fixed number of possible properties K for each product.

For the corpus of restaurant reviews, we assume that the set of properties are specific to a given product, in order to capture fine-grained, relevant properties for each restaurant. For example, reviews from a sandwich shop may contrast the club sandwich with the turkey wrap, while for a more general restaurant, the snippets refer to sandwiches in general. For other domains where the properties are more consistent, it is straightforward to alter our model so that properties are shared across products.

Attribute: An attribute is a description of a property. There are multiple attribute *types*, which may correspond to semantic differences. We assume a fixed, pre-specified number of attributes N. For example, in the case of product reviews, we select $N = 2$ attributes corresponding to positive and negative sentiment. In the case of information extraction, it may be beneficial to use numeric and alphabetic types.

One of the goals of this work in the review domain is to improve sentiment prediction by exploiting correlations within a single property cluster. For example, if there are already many snippets with the attribute representing positive sentiment in a given property cluster, additional snippets are biased towards positive sentiment as well; however, data can always override this bias.

Snippets themselves are always observed; the goal of this work is to induce the latent property and attribute underlying each snippet.

4 Model

Our model generates the words of all snippets for each product in a collection of products. We use $s^{i,j,w}$ to represent the wth word of the jth snippet of the ith product. We use $\mathbf{s}$ to denote the collection of all snippet words. We also assume a fixed vocabulary of words V.

We present an overview of our generative model in Figure 1 and describe each component in turn:

Global Distributions: At the global level, we draw several unigram distributions: a global background distribution θ_B and attribute distributions θ_A^a for each attribute. The background distribution is meant to encode stop-words and domain white-noise, e.g., `food` in the restaurants domain. In this domain, the positive and negative attribute distributions encode words with positive and negative sentiments (e.g., *delicious* or *terrible*).

Each of these distributions are drawn from Dirichlet priors. The background distribution is drawn from a symmetric Dirichlet with concentration $\lambda_B = 0.2$. The positive and negative attribute distributions are initialized using seed words (V_{seed_a} in Figure 1). These seeds are incorporated into the attribute priors: a non-seed word gets ϵ hyperparameter and a seed word gets $\epsilon + \lambda_A$, where $\epsilon = 0.25$ and $\lambda_A = 1.0$.

Product Level: For the ith product, we draw property unigram distributions $\theta_P^{i,1}, \ldots, \theta_P^{i,K}$ for each of the possible K product properties. The property distribution represents product-specific content distributions over properties discussed in reviews of the product; for instance in the restaurant domains, properties may correspond to distinct menu items. Each $\theta_P^{i,k}$ is drawn from a symmetric Dirichlet prior

Global Level:

- Draw background distribution $\theta_B \sim \text{DIRICHLET}(\lambda_B V)$
- For each attribute type a,
  - Draw attribute distribution $\theta_A^a \sim \text{DIRICHLET}(\epsilon V + \lambda_A V_{seed_a})$

Product Level:

- For each product i,
  - Draw property distributions $\theta_P^k \sim \text{DIRICHLET}(\lambda_P V)$ for $k = 1, \ldots, K$
  - Draw property attribute binomial $\phi^{i,k} \sim \text{BETA}(\alpha_A, \beta_A)$ for $k = 1, \ldots, K$
  - Draw property multinomial $\psi^i \sim \text{DIRICHLET}(\lambda_M K)$

Snippet Level:

- For each snippet j in ith product,
  - Draw snippet property $Z_P^{i,j} \sim \psi^i$
  - Draw snippet attribute $Z_A^{i,j} \sim \phi^{Z_P^{ij}}$
  - Draw sequence of word topic indicators $Z_W^{i,j,w} \sim \Lambda | Z_W^{i,j,w-1}$
  - Draw snippet word given property $Z_P^{i,j}$ and attribute $Z_A^{i,j}$

$$s_{i,j,w} \sim \begin{cases} \theta_P^{i,Z_P^{i,j}}, & \text{when } Z_W^{i,j,w} = P \\ \theta_A^{Z_A^{i,j}}, & \text{when } Z_W^{i,j,w} = A \\ \theta_B, & \text{when } Z_W^{i,j,w} = B \end{cases}$$

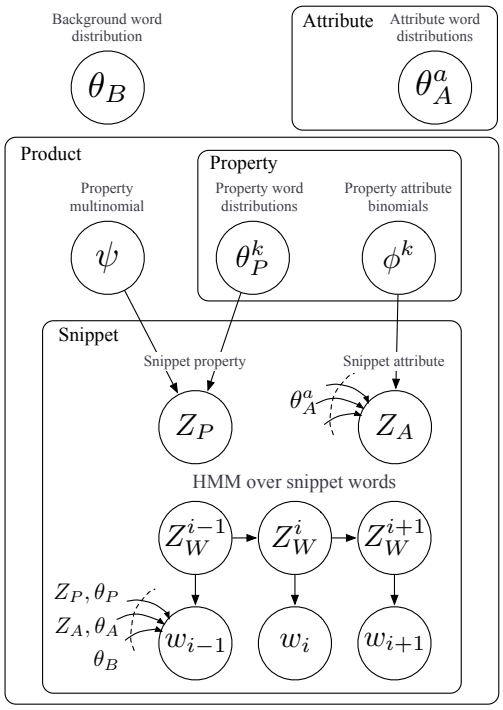


Figure 1: A high-level verbal and graphical description for our model in Section 4. We use $\text{DIRICHLET}(\lambda V)$ to denote a finite Dirichlet prior where the hyper-parameter counts are a scalar times the unit vector of vocabulary items. For the global attribute distribution, the prior hyper-parameter counts are ϵ for all vocabulary items and λ_A for V_{seed_a}, the vector of vocabulary items in the set of seed words for attribute a.

with hyper-parameter $\lambda_P = 0.2$.

For each property $k = 1, \ldots, K$. $\phi_{i,k}$, we draw a binomial distribution $\phi_{i,k}$. This represents the distribution over positive and negative attributes for that property; it is drawn from a beta prior using hyper-parameters $\alpha_A = 2$ and $\beta_A = 2$. We also draw a multinomial ψ^i over K possible properties from a symmetric Dirichlet distribution with hyper-parameter $\lambda_M = 1,000$. This distribution is used to draw snippet properties.

Snippet Level: For the jth snippet of the ith product, a property random variable $Z_P^{i,j}$ is drawn according to the multinomial ψ^i. Conditioned on this choice, we draw an attribute $Z_A^{i,j}$ (positive or negative) from the property attribute distribution $\phi^{i,Z_P^{j,j}}$.

Once the property $Z_P^{i,j}$ and attribute $Z_A^{i,j}$ have been selected, the tokens of the snippet are generated using a simple HMM. The latent state underlying a token, $Z_W^{i,j,w}$, indicates whether the wth word comes from the property distribution, attribute distribution, or background distribution; we use P, A, or B to denote these respective values of $Z_W^{i,j,w}$.

The sequence $Z_W^{i,j,1}, \ldots, Z_W^{i,j,m}$ is generated using a first-order Markov model. The full transition parameter matrix Λ parametrizes these decisions. Conditioned on the underlying $Z_W^{i,j,w}$, a word, $s^{i,j,w}$ is drawn from $\theta_P^{i,j}$, $\theta_A^{i,Z_P^{i,j}}$, or θ_B for the values P,A, or B respectively.

5 Inference

The goal of inference is to predict the snippet property and attribute distributions over each snippet given all the observed snippets $P(Z_P^{i,j}, Z_A^{i,j} | \mathbf{s})$ for all products i and snippets j. Ideally, we would like to marginalize out nuisance random variables and distributions. Specifically, we approximate the full

model posterior using variational inference:[1]

$$P(\psi, \boldsymbol{\theta_P}, \theta_B, \boldsymbol{\theta_A}, \phi, |\mathbf{s}) \approx Q(\psi, \boldsymbol{\theta_P}, \theta_B, \boldsymbol{\theta_A}, \phi)$$

where ψ, $\boldsymbol{\theta_P}$, ϕ denote the collection of latent distributions in our model. Here, we assume a full mean-field factorization of the variational distribution; see Figure 2 for the decomposition. Each variational factor $q(\cdot)$ represents an approximation of that variable's posterior given observed random variables. The variational distribution $Q(\cdot)$ makes the (incorrect) assumption that the posteriors amongst factors are independent. The goal of variational inference is to set factors $q(\cdot)$ so that it minimizes the KL divergence to the true model posterior:

$$\min_{Q(\cdot)} KL(P(\psi, \boldsymbol{\theta_P}, \theta_B, \boldsymbol{\theta_A}, \phi, |\mathbf{s}) \| Q(\psi, \boldsymbol{\theta_P}, \theta_B, \boldsymbol{\theta_A}, \phi)$$

We optimize this objective using coordinate descent on the $q(\cdot)$ factors. Concretely, we update each factor by optimizing the above criterion with all other factors fixed to current values. For instance, the update for the factor $q(Z_W^{i,j,w})$ takes the form:

$$q(Z_W^{i,j,w}) \leftarrow \mathbb{E}_{Q/q(Z_W^{i,j,w})} \lg P(\psi, \boldsymbol{\theta_P}, \theta_B, \boldsymbol{\theta_A}, \phi, \mathbf{s})$$

The full factorization of $Q(\cdot)$ and updates for all random variable factors are given in Figure 2. Updates of parameter factors are omitted; however these are derived through simple counts of the Z_A, Z_P, and Z_W latent variables. For related discussion, see Blei et al. (2003).

6 Experiments

In this section, we describe in detail our data set and present three experiments and their results.

Data Set Our data set consists of snippets from Yelp reviews generated by the system described in Sauper et al. (2010). This system is trained to extract snippets containing short descriptions of user sentiment towards some aspect of a restaurant.[2] We select only the snippets labeled by that system as referencing *food*, and we ignore restaurants with fewer than 20 snippets. There are 13,879 snippets in total, taken from 328 restaurants in and around the Boston/Cambridge area. The average snippet length is 7.8 words, and there are an average of 42.1 snippets per restaurant, although there is high variance in number of snippets for each restaurant. Figure 3 shows some example snippets.

The [P noodles] and the [P meat] were actually [+ pretty good].
I [+ recommend] the [P chicken noodle pho].
The [P noodles] were [- soggy].
The [P chicken pho] was also [+ good].

The [P spring rolls] and [P coffee] were [+ good] though.
The [P spring roll wrappers] were a [- little dry tasting].
My [+ favorites] were the [P crispy spring rolls].
The [P Crispy Tuna Spring Rolls] are [+ fantastic]!

The [P lobster roll] my mother ordered was [- dry] and [- scant].
The [P portabella mushroom] is my [+ go-to] [P sandwich].
The [P bread] on the [P sandwich] was [- stale].
The slice of [P tomato] was [- rather measly].

The [P shumai] and [P California maki sushi] were [+ decent].
The [P spicy tuna roll] and [P eel roll] were [+ perfect].
The [P rolls] with [P spicy mayo] were [- not so great].
I [+ love] [P Thai rolls].

Figure 3: Example snippets from our data set, grouped according to property. Property words are labeled **P** and colored blue, NEGATIVE attribute words are labeled **-** and colored red, and POSITIVE attribute words are labeled **+** and colored green. The grouping and labeling are *not* given in the data set and must be learned by the model.

For sentiment attribute seed words, we use 42 and 33 words for the positive and negative distributions respectively. These are hand-selected based on the restaurant review domain; therefore, they include domain-specific words such as *delicious* and *gross*.

Tasks We perform three experiments to evaluate our model's effectiveness. First, a cluster prediction task is designed to test the quality of the learned property clusters. Second, an attribute analysis task will evaluate the sentiment analysis portion of the model. Third, we present a task designed to test whether the system can correctly identify properties which have conflicting attributes, which tests both clustering and sentiment analysis.

[1] See Liang and Klein (2007) for an overview of variational techniques.

[2] For exact training procedures, please reference that paper.

Mean-field Factorization

$$Q(\boldsymbol{\psi}, \boldsymbol{\theta_P}, \theta_B, \boldsymbol{\theta_A}, \boldsymbol{\phi}) = q(\theta_B) \left(\prod_{a=1}^{N} q(\theta_A^a) \right) \left(\prod_{i}^{n} \left(\prod_{k=1}^{K} q(\theta_P^{i,k}) q(\phi^{i,k}) \right) \left(\prod_{j} q(Z_A^{i,j}) q(Z_P^{i,j}) \prod_{w} q(Z_W^{i,j,w}) \right) \right)$$

Snippet Property Indicator

$$\lg q(Z_P^{i,j} = k) \propto \mathbb{E}_{q(\psi^i)} \lg \psi^i(p) + \sum_{w} q(Z_W^{i,j,w} = P) \mathbb{E}_{q(\theta_P^{i,k})} \lg \theta_P^{i,k}(s^{i,j,w}) + \sum_{a=1}^{N} q(Z_A^{i,j} = a) \mathbb{E}_{q(\phi^{i,k})} \lg \phi^{i,k}(a)$$

Snippet Attribute Indicator

$$\lg q(Z_A^{i,j} = a) = \sum_{k} q(Z_P^{i,j} = k) \mathbb{E}_{q(\phi^{i,k})} \lg \phi^{i,k}(a) + \sum_{w} q(Z_W^{i,j,w} = A) \mathbb{E}_{q(\theta_A^a)} \lg \theta_A^a(s^{i,j,w})$$

Word Topic Indicator

$$\lg q(Z_W^{i,j,w} = P) \propto \lg P(Z_W = P) + \sum_{k} q(Z_P^{i,j} = k) \mathbb{E}_{q(\theta_P^{i,k})} \lg \theta_P^{i,j}(s^{i,j,w})$$

$$\lg q(Z_W^{i,j,w} = A) \propto \lg P(Z_W = A) + \sum_{a \in \{+,-\}} q(Z_A^{i,j} = a) \mathbb{E}_{q(\theta_A^a)} \lg \theta_A^a(s^{i,j,w})$$

$$\lg q(Z_W^{i,j,w} = B) \propto \lg P(Z_W = B) + \mathbb{E}_{q(\theta_B)} \lg \theta_B(s^{i,j,w})$$

Figure 2: The mean-field variational algorithm used during learning and inference to obtain posterior predictions over snippet properties and attributes, as described in Section 5. Mean-field inference consists of updating each of the latent variable factors as well as a straightforward update of latent parameters in round robin fashion.

6.1 Cluster prediction

The goal of this task is to evaluate the quality of property clusters; specifically the $Z_P^{i,j}$ variable in Section 4. In an ideal clustering, the predicted clusters will be cohesive (i.e., all snippets predicted for a given property are related to each other) and comprehensive (i.e., all snippets which are related to a property are predicted for it). For example, a snippet will be assigned the property *pad thai* if and only if that snippet mentions some aspect of the pad thai.

Annotation For this task, we use a set of gold clusters over 3,250 snippets across 75 restaurants collected through Mechanical Turk. In each task, a worker was given a set of 25 snippets from a single restaurant and asked to cluster them into as many clusters as they desired, with the option of leaving any number unclustered. This yields a set of gold clusters and a set of unclustered snippets. For verification purposes, each task was provided to two different workers. The intersection of both workers' judgments was accepted as the gold standard, so the model is not evaluated on judgments which disagree. In total, there were 130 unique tasks, each of which were provided to two workers, for a total output of 210 generated clusters.

Baseline The baseline for this task is a clustering algorithm weighted by TF*IDF over the data set as implemented by the publicly available CLUTO package.[3] This baseline will put a strong connection between things which are lexically similar. Because our model only uses property words to tie together clusters, it may miss correlations between words which are not correctly identified as property words. The baseline is allowed 10 property clusters per restaurant.

We use the MUC cluster evaluation metric for this task (Vilain et al., 1995). This metric measures the number of cluster merges and splits required to recreate the gold clusters given the model's output.

[3] Available at http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview with agglomerative clustering, using the cosine similarity distance metric.

	Precision	Recall	F1
Baseline	80.2	61.1	69.3
Our model	72.2	79.1	75.5

Table 2: Results using the MUC metric on the cluster prediction task. Note that while the precision of the baseline is higher, the recall and overall F1 of our model outweighs that. While MUC has a deficiency in that putting everything into a single cluster will artificially inflate the score, parameters on our model are set so that the model uses the same number of clusters as the baseline system.

Therefore, it can concisely show how accurate our clusters are as a whole. While it would be possible to artificially inflate the score by putting everything into a single cluster, the parameters on our model and the likelihood objective are such that the model prefers to use all available clusters, the same number as the baseline system.

Results Results for our cluster prediction task are in Table 2. While our system does suffer on precision in comparison to the baseline system, the recall gains far outweigh this loss, for a total error reduction of 20% on the MUC measure.

The most common cause of poor cluster choices in the baseline system is its inability to distinguish property words from attribute words. For example, if many snippets in a given restaurant use the word *delicious*, there may end up being a cluster based on that alone. Because our system is capable of distinguishing which words are property words (i.e., words relevant to clustering), it can choose clusters which make more sense overall. We show an example of this in Table 3.

6.2 Attribute analysis

We also evaluate the system's predictions of snippet attribute using the predicted posterior over the attribute distribution for the snippet (i.e., $Z_A^{i,j}$). For this task, we consider the binary judgment to be simply the one with higher value in $q(Z_A^{i,j})$ (see Section 5). The goal of this task is to evaluate whether our model correctly distinguishes attribute words.

Annotation For this task, we use a set of 260 total snippets from the Yelp reviews for 30 restaurants, evenly split into a training and test sets of 130 snippets each. These snippets are manually labeled POSITIVE or NEGATIVE. Neutral snippets are ignored for the purpose of this experiment.

The martini selection looked delicious The s'mores martini sounded excellent
The martinis were good The martinis are very good
The mozzarella was very fresh The fish and various meets were very well made

The best carrot cake I've ever eaten Carrot cake was deliciously moist
The carrot cake was delicious.
It was rich, creamy and delicious. The pasta Bolognese was rich and robust.

Table 3: Example phrases from clusters in both the baseline and our model. For each pair of clusters, the dashed line indicates separation by the baseline model, while the solid line indicates separation by our model. In the first example, the baseline mistakenly clusters some snippets about *martinis* with those containing the word *very*. In the second example, the same occurs with the word *delicious*.

Baseline We use two baselines for this task, one based on a standard discriminative classifier and one based on the seed words from our model.

The DISCRIMINATIVE baseline for this task is a standard maximum entropy discriminative binary classifier over unigrams. Given enough snippets from enough unrelated properties, the classifier should be able to identify that words like *great* indicate positive sentiment and those like *bad* indicate negative sentiment, while words like *chicken* are neutral and have no effect.

The SEED baseline simply counts the number of words from the positive and negative seed lists used by the model, V_{seed_+} and V_{seed_-}. If there are more words from V_{seed_+}, the snippet is labeled positive, and if there are more words from V_{seed_-}, the snippet is labeled negative. If there is a tie or there are no seed words, we split the prediction. Because the seed word lists are specifically slanted toward restaurant reviews (i.e., they contain words such as *delicious*), this baseline should perform well.

Results For this experiment, we measure the overall classification accuracy of each system (see Table

	Accuracy
DISCRIMINATIVE baseline	75.9
SEED baseline	78.2
Our model	80.2

Table 4: Attribute prediction accuracy of the full system compared to the DISCRIMINATIVE and SEED baselines. The advantage of our system is its ability to distinguish property words from attribute words in order to restrict judgment to only the relevant terms.

The naan was **hot** and **fresh**
All the veggies were really **fresh** and **crisp**.
Perfect mix of **fresh flavors** and **comfort** food
The lo main smelled and tasted **rancid**
My grilled cheese sandwich was a little **gross**

Table 5: Examples of sentences correctly labeled by our system but incorrectly labeled by the DISCRIMINATIVE baseline; the key sentiment words are highlighted. Notice that these words are not the most common sentiment words; therefore, it is difficult for the classifier to make a correct generalization. Only two of these words are seed words for our model (*fresh* and *gross*).

4). Our system outperforms both supervised baselines.

As in the cluster prediction case, the main flaw with the DISCRIMINATIVE baseline system is its inability to recognize which words are relevant for the task at hand, in this case the attribute words. By learning to separate attribute words from the other words in the snippets, our full system is able to more accurately judge their sentiment. Examples of these cases are found in Table 5.

The obvious flaw in the SEED baseline is the inability to pre-specify every possible sentiment word; our model's performance indicates that it is learning something beyond just these basic words.

6.3 Conflict identification

Our final task requires both correct cluster prediction and correct sentiment judgments. In many domains, it is interesting to know not only whether a product is rated highly, but also whether there is conflicting sentiment or debate. In the case of restaurant reviews, it is relevant to know whether the dishes are consistently good or whether there is some variation in quality.

Judgment P	Judgment A		Attribute / Snippet
Yes	Yes	-	The salsa isn't great
		+	Chips and salsa are sublime
		-	The grits were good, but not great.
		+	Grits were the perfect consistency
		-	The tom yum kha was bland
		+	It's the best Thai soup I ever had
		-	The naan is a bit doughy and undercooked
		+	The naan was pretty tasty
		-	My reuben was a little dry.
		+	The reuben was a good reuben.
Yes	No	-	Belgian frites are crave-able
		+	The frites are very, very good.
No	Yes	-	The blackened chicken was meh
		+	Chicken enchiladas are yummy!
		-	The taste overall was mediocre
		+	The oysters are tremendous
No	No	-	The cream cheese wasn't bad
		+	Ice cream was just delicious

Table 6: Example property-attribute correctness for the conflict identification task, over both property and attribute. Property judgment (P) indicates whether the snippets are discussing the same item; attribute judgment (A) indicates whether there is a correct difference in attribute (sentiment), regardless of properties.

To evaluate this, we examine the output clusters which contain predictions of both positive and negative snippets. The goal is to identify whether these are true conflicts of sentiment or there was a failure in either property clustering or attribute classification.

For this task, the output clusters are manually annotated for correctness of both property and attribute judgments, as in Table 6. As there is no obvious baseline for this experiment, we treat it simply as an analysis of errors.

Results For this task, we examine the accuracy of conflict prediction, both with and without the correctly identified properties. The results by property-attribute correctness are shown in Table 7. From these numbers, we can see that 50% of the clusters are correct in both property (cohesiveness) and attribute (difference in sentiment) dimensions.

Overall, the properties are correctly identified (subject of NEG matches the subject of POS) 68% of the time and a correct difference in attribute is identified 67% of the time. Of the clusters which are correct in property, 74% show a correctly labeled

Judgment		
P	A	# Clusters
Yes	Yes	52
Yes	No	18
No	Yes	17
No	No	15

Table 7: Results of conflict analysis by correctness of property label (P) and attribute conflict (A). Examples of each type of correctness pair are show in in Table 6. 50% of the clusters are correct in both labels, and there are approximately the same number of errors toward both property and attribute.

difference in attribute.

7 Conclusion

We have presented a probabilistic topic model for identifying properties and attitudes of product review snippets. The model is relatively simple and admits an efficient variational mean-field inference procedure which is parallelized and can be run on a large number of snippets. We have demonstrated on multiple evaluation tasks that our model outperforms applicable baselines by a considerable margin.

Acknowledgments

The authors acknowledge the support of the NSF (CAREER grant IIS-0448168), NIH (grant 5-R01-LM009723-02), Nokia, and the DARPA Machine Reading Program (AFRL prime contract no. FA8750-09-C-0172). Thanks to Peter Szolovits and the MIT NLP group for their helpful comments. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Giuseppe Carenini, Raymond Ng, and Adam Pauls. 2006. Multi-document summarization of evaluative text. In *Proceedings of EACL*, pages 305–312.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of SIGKDD*, pages 168–177.

Minqing Hu and Bing Liu. 2006. Opinion extraction and summarization on the web. In *Proceedings of AAAI*.

Soo-Min Kim and Eduard Hovy. 2006. Automatic identification of pro and con reasons in online reviews. In *Proceedings of COLING/ACL*, pages 483–490.

Hyun Duk Kim and ChengXiang Zhai. 2009. Generating comparative summaries of contradictory opinions in text. In *Proceedings of CIKM*, pages 385–394.

P. Liang and D. Klein. 2007. Structured Bayesian nonparametric models with variational inference (tutorial). In *Proceedings of ACL*.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005a. Opinion observer: Analyzing and comparing opinions on the web. In *Proceedings of WWW*, pages 342–351.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005b. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of WWW*, pages 342–351.

Yue Lu and ChengXiang Zhai. 2008. Opinion integration through semi-supervised topic modeling. In *Proceedings of WWW*, pages 121–130.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. 2007. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of WWW*, pages 171–180.

Ana-Maria Popescu, Bao Nguyen, and Oren Etzioni. 2005. OPINE: Extracting product features and opinions from reviews. In *Proceedings of HLT/EMNLP*, pages 339–346.

Christina Sauper, Aria Haghighi, and Regina Barzilay. 2010. Incorporating content structure into text analysis applications. In *Proceedings of EMNLP*, pages 377–387.

Yohei Seki, Koji Eguchi, Noriko K, and Masaki Aono. 2006. Opinion-focused summarization and its analysis at DUC 2006. In *Proceedings of DUC*, pages 122–130.

Ivan Titov and Ryan McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of ACL*, pages 308–316.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of MUC*, pages 45–52.

Recognizing Named Entities in Tweets

Xiaohua Liu [‡] [†], **Shaodian Zhang**[*] [§], **Furu Wei** [†], **Ming Zhou** [†]
[‡]School of Computer Science and Technology
Harbin Institute of Technology, Harbin, 150001, China
[§]Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, 200240, China
[†]Microsoft Research Asia
Beijing, 100190, China
[†]{xiaoliu, fuwei, mingzhou}@microsoft.com
[§] zhangsd.sjtu@gmail.com

Abstract

The challenges of Named Entities Recognition (NER) for tweets lie in the insufficient information in a tweet and the unavailability of training data. We propose to combine a K-Nearest Neighbors (KNN) classifier with a linear Conditional Random Fields (CRF) model under a semi-supervised learning framework to tackle these challenges. The KNN based classifier conducts pre-labeling to collect global coarse evidence across tweets while the CRF model conducts sequential labeling to capture fine-grained information encoded in a tweet. The semi-supervised learning plus the gazetteers alleviate the lack of training data. Extensive experiments show the advantages of our method over the baselines as well as the effectiveness of KNN and semi-supervised learning.

1 Introduction

Named Entities Recognition (NER) is generally understood as the task of identifying mentions of rigid designators from text belonging to named-entity types such as persons, organizations and locations (Nadeau and Sekine, 2007). Proposed solutions to NER fall into three categories: 1) The rule-based (Krupka and Hausman, 1998); 2) the machine learning based (Finkel and Manning, 2009; Singh et al., 2010) ; and 3) hybrid methods (Jansche and Abney, 2002). With the availability of annotated corpora, such as ACE05, Enron (Minkov et al., 2005) and CoNLL03 (Tjong Kim Sang and De Meulder, 2003), the data driven methods now become the dominating methods.

However, current NER mainly focuses on formal text such as news articles (Mccallum and Li, 2003; Etzioni et al., 2005). Exceptions include studies on informal text such as emails, blogs, clinical notes (Wang, 2009). Because of the domain mismatch, current systems trained on non-tweets perform poorly on tweets, a new genre of text, which are short, informal, ungrammatical and noise prone. For example, the average F1 of the Stanford NER (Finkel et al., 2005) , which is trained on the CoNLL03 shared task data set and achieves state-of-the-art performance on that task, drops from 90.8% (Ratinov and Roth, 2009) to 45.8% on tweets.

Thus, building a domain specific NER for tweets is necessary, which requires a lot of annotated tweets or rules. However, manually creating them is tedious and prohibitively unaffordable. Proposed solutions to alleviate this issue include: 1) Domain adaption, which aims to reuse the knowledge of the source domain in a target domain. Two recent examples are Wu et al. (2009), which uses data that is informative about the target domain and also easy to be labeled to bridge the two domains, and Chiticariu et al. (2010), which introduces a high-level rule language, called NERL, to build the general and domain specific NER systems; and 2) semi-supervised learning, which aims to use the abundant unlabeled data to compensate for the lack of annotated data. Suzuki and Isozaki (2008) is one such example.

Another challenge is the limited information in tweet. Two factors contribute to this difficulty. One

[*] This work has been done while the author was visiting Microsoft Research Asia.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 359–367,
Portland, Oregon, June 19-24, 2011.

is the tweet's informal nature, making conventional features such as part-of-speech (POS) and capitalization not reliable. The performance of current NLP tools drops sharply on tweets. For example, OpenNLP [1], the state-of-the-art POS tagger, gets only an accuracy of 74.0% on our test data set. The other is the tweet's short nature, leading to the excessive abbreviations or shorthand in tweets, and the availability of very limited context information. Tackling this challenge, ideally, requires adapting related NLP tools to fit tweets, or normalizing tweets to accommodate existing tools, both of which are hard tasks.

We propose a novel NER system to address these challenges. Firstly, a K-Nearest Neighbors (KNN) based classifier is adopted to conduct word level classification, leveraging the similar and recently labeled tweets. Following the two-stage prediction aggregation methods (Krishnan and Manning, 2006), such pre-labeled results, together with other conventional features used by the state-of-the-art NER systems, are fed into a linear Conditional Random Fields (CRF) (Lafferty et al., 2001) model, which conducts fine-grained tweet level NER. Furthermore, the KNN and CRF model are repeatedly retrained with an incrementally augmented training set, into which high confidently labeled tweets are added. Indeed, it is the combination of KNN and CRF under a semi-supervised learning framework that differentiates ours from the existing. Finally, following Lev Ratinov and Dan Roth (2009), 30 gazetteers are used, which cover common names, countries, locations, temporal expressions, etc. These gazetteers represent general knowledge across domains. The underlying idea of our method is to combine global evidence from KNN and the gazetteers with local contextual information, and to use common knowledge and unlabeled tweets to make up for the lack of training data.

12,245 tweets are manually annotated as the test data set. Experimental results show that our method outperforms the baselines. It is also demonstrated that integrating KNN classified results into the CRF model and semi-supervised learning considerably boost the performance.

Our contributions are summarized as follows.

1. We propose to a novel method that combines a KNN classifier with a conventional CRF based labeler under a semi-supervised learning framework to combat the lack of information in tweet and the unavailability of training data.

2. We evaluate our method on a human annotated data set, and show that our method outperforms the baselines and that both the combination with KNN and the semi-supervised learning strategy are effective.

The rest of our paper is organized as follows. In the next section, we introduce related work. In Section 3, we formally define the task and present the challenges. In Section 4, we detail our method. In Section 5, we evaluate our method. Finally, Section 6 concludes our work.

2 Related Work

Related work can be roughly divided into three categories: NER on tweets, NER on non-tweets (e.g., news, bio-logical medicine, and clinical notes), and semi-supervised learning for NER.

2.1 NER on Tweets

Finin et al. (2010) use Amazons Mechanical Turk service [2] and CrowdFlower [3] to annotate named entities in tweets and train a CRF model to evaluate the effectiveness of human labeling. In contrast, our work aims to build a system that can automatically identify named entities in tweets. To achieve this, a KNN classifier with a CRF model is combined to leverage cross tweets information, and the semi-supervised learning is adopted to leverage unlabeled tweets.

2.2 NER on Non-Tweets

NER has been extensively studied on formal text, such as news, and various approaches have been proposed. For example, Krupka and Hausman (1998) use manual rules to extract entities of predefined types; Zhou and Ju (2002) adopt Hidden Markov Models (HMM) while Finkel et al. (2005) use CRF to train a sequential NE labeler, in which the BIO (meaning Beginning, the Inside and the Outside of

[1] http://sourceforge.net/projects/opennlp/

[2] https://www.mturk.com/mturk/

[3] http://crowdflower.com/

an entity, respectively) schema is applied. Other methods, such as classification based on Maximum Entropy models and sequential application of Perceptron or Winnow (Collins, 2002), are also practiced. The state-of-the-art system, e.g., the Stanford NER, can achieve an F1 score of over 92.0% on its test set.

Biomedical NER represents another line of active research. Machine learning based systems are commonly used and outperform the rule based systems. A state-of-the-art biomedical NER system (Yoshida and Tsujii, 2007) uses lexical features, orthographic features, semantic features and syntactic features, such as part-of-speech (POS) and shallow parsing.

A handful of work on other domains exists. For example, Wang (2009) introduces NER on clinical notes. A data set is manually annotated and a linear CRF model is trained, which achieves an F-score of 81.48% on their test data set; Downey et al. (2007) employ capitalization cues and n-gram statistics to locate names of a variety of classes in web text; most recently, Chiticariu et al. (2010) design and implement a high-level language NERL that is tuned to simplify the process of building, understanding, and customizing complex rule-based named-entity annotators for different domains.

Ratinov and Roth (2009) systematically study the challenges in NER, compare several solutions and report some interesting findings. For example, they show that a conditional model that does not consider interactions at the output level performs comparably to beam search or Viterbi, and that the BILOU (Beginning, the Inside and the Last tokens of multi-token chunks as well as Unit-length chunks) encoding scheme significantly outperforms the BIO schema (Beginning, the Inside and Outside of a chunk).

In contrast to the above work, our study focuses on NER for tweets, a new genre of texts, which are short, noise prone and ungrammatical.

2.3 Semi-supervised Learning for NER

Semi-supervised learning exploits both labeled and un-labeled data. It proves useful when labeled data is scarce and hard to construct while unlabeled data is abundant and easy to access.

Bootstrapping is a typical semi-supervised learning method. It iteratively adds data that has been confidently labeled but is also informative to its training set, which is used to re-train its model. Jiang and Zhai (2007) propose a balanced bootstrapping algorithm and successfully apply it to NER. Their method is based on instance re-weighting, which allows the small amount of the bootstrapped training sets to have an equal weight to the large source domain training set. Wu et al. (2009) propose another bootstrapping algorithm that selects bridging instances from an unlabeled target domain, which are informative about the target domain and are also easy to be correctly labeled. We adopt bootstrapping as well, but use human labeled tweets as seeds.

Another representative of semi-supervised learning is learning a robust representation of the input from unlabeled data. Miller et al. (2004) use word clusters (Brown et al., 1992) learned from unlabeled text, resulting in a performance improvement of NER. Guo et al. (2009) introduce Latent Semantic Association (LSA) for NER. In our pilot study of NER for tweets, we adopt bag-of-words models to represent a word in tweet, to concentrate our efforts on combining global evidence with local information and semi-supervised learning. We leave it to our future work to explore which is the best input representation for our task.

3 Task Definition

We first introduce some background about tweets, then give a formal definition of the task.

3.1 The Tweets

A tweet is a short text message containing no more than 140 characters in Twitter, the biggest micro-blog service. Here is an example of tweets: “mycraftingworld: #Win Microsoft Office 2010 Home and Student *2Winners* #Contest from @office and @momtobedby8 #Giveaway http://bit.ly/bCsLOr ends 11/14”, where ”mycraftingworld” is the name of the user who published this tweet. Words beginning with the “#” character, like “”#Win”, “#Contest” and “#Giveaway”, are hash tags, usually indicating the topics of the tweet; words starting with “@”, like “@office” and “@momtobedby8”, represent user names, and “http://bit.ly/bCsLOr” is a shortened link.

Twitter users are interested in named entities, such

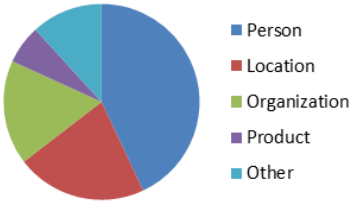


Figure 1: Portion of different types of named entities in tweets. This is based on an investigation of 12,245 randomly sampled tweets, which are manually labeled.

as person names, organization names and product names, as evidenced by the abundant named entities in tweets. According to our investigation on 12,245 randomly sampled tweets that are manually labeled, about 46.8% have at least one named entity. Figure 1 shows the portion of named entities of different types.

3.2 The Task

Given a tweet as input, our task is to identify both the boundary and the class of each mention of entities of predefined types. We focus on four types of entities in our study, i.e., persons, organizations, products, and locations, which, according to our investigation as shown in Figure 1, account for 89.0% of all the named entities.

Here is an example illustrating our task. The input is "...Me without you is like an iphone without apps, Justin Bieber without his hair, Lady gaga without her telephone, it just wouldn..." The expected output is as follows:"...Me without you is like an <PRODUCT >iphone</PRODUCT>without apps, <PERSON>Justin Bieber</PERSON>without his hair,<PERSON>Lady gaga</PERSON> without her telephone, it just wouldn...", meaning that "iphone" is a product, while "Justin Bieber" and "Lady gaga" are persons.

4 Our Method

Now we present our solution to the challenging task of NER for tweets. An overview of our method is first given, followed by detailed discussion of its core components.

4.1 Method Overview

NER task can be naturally divided into two sub-tasks, i.e., boundary detection and type classification. Following the common practice , we adopt a sequential labeling approach to jointly resolve these sub-tasks, i.e., for each word in the input tweet, a label is assigned to it, indicating both the boundary and entity type. Inspired by Ratinov and Roth (2009), we use the BILOU schema.

Algorithm 1 outlines our method, where: $train_s$ and $train_k$ denote two machine learning processes to get the CRF labeler and the KNN classifier, respectively; $repr_w$ converts a word in a tweet into a bag-of-words vector; the $repr_t$ function transforms a tweet into a feature matrix that is later fed into the CRF model; the knn function predicts the class of a word; the $update$ function applies the predicted class by KNN to the inputted tweet; the crf function conducts word level NE labeling;τ and γ represent the minimum labeling confidence of KNN and CRF, respectively, which are experimentally set to 0.1 and 0.001; N (1,000 in our work) denotes the maximum number of new accumulated training data.

Our method, as illustrated in Algorithm 1, repeatedly adds the new confidently labeled tweets to the training set [4] and retrains itself once the number of new accumulated training data goes above the threshold N. Algorithm 1 also demonstrates one striking characteristic of our method: A KNN classifier is applied to determine the label of the current word before the CRF model. The labels of the words that confidently assigned by the KNN classifier are treated as visible variables for the CRF model.

4.2 Model

Our model is hybrid in the sense that a KNN classifier and a CRF model are sequentially applied to the target tweet, with the goal that the KNN classifier captures global coarse evidence while the CRF model fine-grained information encoded in a single tweet and in the gazetteers. Algorithm 2 outlines the training process of KNN, which records the labeled word vector for every type of label.

Algorithm 3 describes how the KNN classifier

[4]The training set ts has a maximum allowable number of items, which is 10,000 in our work. Adding an item into it will cause the oldest one being removed if it is full.

Algorithm 1 NER for Tweets.

Require: Tweet stream i; output stream o.
Require: Training tweets ts; gazetteers ga.

Initialize l_s, the CRF labeler: $l_s = train_s(ts)$.
Initialize l_k, the KNN classifier: $l_k = train_k(ts)$.
Initialize n, the # of new training tweets: $n = 0$.
while Pop a tweet t from i and $t \neq null$ **do**
    for Each word $w \in t$ **do**
        Get the feature vector $\vec{w}$: $\vec{w} = repr_w(w, t)$.
        Classify $\vec{w}$ with knn: $(c, cf) = knn(l_k, \vec{w})$.
        if $cf > \tau$ **then**
            Pre-label: $t = update(t, w, c)$.
        end if
    end for
    Get the feature vector $\vec{t}$: $\vec{t} = repr_t(t, ga)$.
    Label $\vec{t}$ with crf: $(t, cf) = crf(l_s, \vec{t})$.
    Put labeled result (t, cf) into o.
    if $cf > \gamma$ **then**
        Add labeled result t to ts , $n = n + 1$.
    end if
    if $n > N$ **then**
        Retrain l_s: $l_s = train_s(ts)$.
        Retrain l_k: $l_k = train_k(ts)$.
        $n = 0$.
    end if
end while
return o.

Algorithm 2 KNN Training.

Require: Training tweets ts.

Initialize the classifier l_k:$l_k = \varnothing$.
for Each tweet $t \in ts$ **do**
    for Each word,label pair $(w, c) \in t$ **do**
        Get the feature vector $\vec{w}$: $\vec{w} = repr_w(w, t)$.
        Add the $\vec{w}$ and c pair to the classifier: $l_k = l_k \cup \{(\vec{w}, c)\}$.
    end for
end for
return KNN classifier l_k.

predicts the label of the word. In our work, K is experimentally set to 20, which yields the best performance.

Two desirable properties of KNN make it stand out from its alternatives: 1) It can straightforwardly incorporate evidence from new labeled tweets and retraining is fast; and 2) combining with a CRF model, which is good at encoding the subtle interactions between words and their labels, compensates for KNN's incapability to capture fine-grained evidence involving multiple decision points.

Algorithm 3 KNN predication.

Require: KNN classifier l_k ;word vector $\vec{w}$.

Initialize nb, the neighbors of $\vec{w}$: $nb = neigbors(l_k, \vec{w})$.
Calculate the predicted class c^*: $c^* = argmax_c \sum_{(\vec{w}', c') \in nb} \delta(c, c') \cdot cos(\vec{w}, \vec{w}')$.
Calculate the labeling confidence cf: $cf = \frac{\sum_{(\vec{w}', c') \in nb} \delta(c, c') \cdot cos(\vec{w}, \vec{w}')}{\sum_{(\vec{w}', c') \in nb} cos(\vec{w}, \vec{w}')}$.
return The predicted label c^* and its confidence cf.

The Linear CRF model is used as the fine model, with the following considerations: 1) It is well-studied and has been successfully used in state-of-the-art NER systems (Finkel et al., 2005; Wang, 2009); 2) it can output the probability of a label sequence, which can be used as the labeling confidence that is necessary for the semi-supervised learning framework.

In our experiments, the CRF++ [5] toolkit is used to train a linear CRF model. We have written a Viterbi decoder that can incorporate partially observed labels to implement the crf function in Algorithm 1.

4.3 Features

Given a word in a tweet, the KNN classifier considers a text window of size 5 with the word in the middle (Zhang and Johnson, 2003), and extracts bag-of-word features from the window as features. For each word, our CRF model extracts similar features as Wang (2009) and Ratinov and Roth (2009), namely, orthographic features, lexical features and gazetteers related features. In our work, we use the gazetteers provided by Ratinov and Roth (2009).

Two points are worth noting here. One is that before feature extraction for either the KNN or the CRF, stop words are removed. The stop words used here are mainly from a set of frequently-used words [6]. The other is that tweet meta data is normalized, that is, every link becomes *LINK* and every

[5] http://crfpp.sourceforge.net/

[6] http://www.textfixer.com/resources/common-english-words.txt

account name becomes *ACCOUNT*. Hash tags are treated as common words.

4.4 Discussion

We now discuss several design considerations related to the performance of our method, i.e., additional features, gazetteers and alternative models.

Additional Features. Features related to chunking and parsing are not adopted in our final system, because they give only a slight performance improvement while a lot of computing resources are required to extract such features. The ineffectiveness of these features is linked to the noisy and informal nature of tweets. Word class (Brown et al., 1992) features are not used either, which prove to be unhelpful for our system. We are interested in exploring other tweet representations, which may fit our NER task, for example the LSA models (Guo et al., 2009).

Gazetteers. In our work, gazetteers prove to be substantially useful, which is consistent with the observation of Ratinov and Roth (2009). However, the gazetteers used in our work contain noise, which hurts the performance. Moreover, they are static, directly from Ratinov and Roth (2009), thus with a relatively lower coverage, especially for person names and product names in tweets. We are developing tools to clean the gazetteers. In future, we plan to feed the fresh entities correctly identified from tweets back into the gazetteers. The correctness of an entity can rely on its frequency or other evidence.

Alternative Models. We have replaced KNN by other classifiers, such as those based on Maximum Entropy and Support Vector Machines, respectively. KNN consistently yields comparable performance, while enjoying a faster retraining speed. Similarly, to study the effectiveness of the CRF model, it is replaced by its alternations, such as the HMM labeler and a beam search plus a maximum entropy based classifier. In contrast to what is reported by Ratinov and Roth (2009), it turns out that the CRF model gives remarkably better results than its competitors. Note that all these evaluations are on the same training and testing data sets as described in Section 5.1.

5 Experiments

In this section, we evaluate our method on a manually annotated data set and show that our system outperforms the baselines. The contributions of the combination of KNN and CRF as well as the semi-supervised learning are studied, respectively.

5.1 Data Preparation

We use the Twigg SDK [7] to crawl all tweets from April 20^{th} 2010 to April 25^{th} 2010, then drop non-English tweets and get about 11,371,389, from which 15,800 tweets are randomly sampled, and are then labeled by two independent annotators, so that the beginning and the end of each named entity are marked with <TYPE> and </TYPE>, respectively. Here TYPE is PERSON, PRODUCT, ORGANIZATION or LOCATION. 3555 tweets are dropped because of inconsistent annotation. Finally we get 12,245 tweets, forming the gold-standard data set. Figure 1 shows the portion of named entities of different types. On average, a named entity has 1.2 words. The gold-standard data set is evenly split into two parts: One for training and the other for testing.

5.2 Evaluation Metrics

For every type of named entity, Precision (Pre.), recall (Rec.) and F1 are used as the evaluation metrics. Precision is a measure of what percentage the output labels are correct, and recall tells us to what percentage the labels in the gold-standard data set are correctly labeled, while F1 is the harmonic mean of precision and recall. For the overall performance, we use the average Precision, Recall and F1, where the weight of each name entity type is proportional to the number of entities of that type. These metrics are widely used by existing NER systems to evaluate their performance.

5.3 Baselines

Two systems are used as baselines: One is the dictionary look-up system based on the gazetteers; the other is the modified version of our system without KNN and semi-supervised learning. Hereafter these two baselines are called NER_{DIC} and NER_{BA}, respectively. The OpenNLP and the Stanford parser (Klein and Manning, 2003) are used to extract linguistic features for the baselines and our method.

[7] It is developed by the Bing social search team, and currently is only internally available.

System	Pre.(%)	Rec.(%)	F1(%)
NER_{CB}	81.6	78.8	80.2
NER_{BA}	83.6	68.6	75.4
NER_{DIC}	32.6	25.4	28.6

Table 1: Overall experimental results.

System	Pre.(%)	Rec.(%)	F1(%)
NER_{CB}	78.4	74.5	76.4
NER_{BA}	83.6	68.4	75.2
NER_{DIC}	37.1	29.7	33.0

Table 2: Experimental results on PERSON.

System	Pre.(%)	Rec.(%)	F1(%)
NER_{CB}	81.3	65.4	72.5
NER_{BA}	82.5	58.4	68.4
NER_{DIC}	8.2	6.1	7.0

Table 3: Experimental results on PRODUCT.

System	Pre.(%)	Rec.(%)	F1(%)
NER_{CB}	80.3	77.5	78.9
NER_{BA}	81.6	69.7	75.2
NER_{DIC}	30.2	30.0	30.1

Table 4: Experimental results on LOCATION.

System	Pre.(%)	Rec.(%)	F1(%)
NER_{CB}	83.2	60.4	70.0
NER_{BA}	87.6	52.5	65.7
NER_{DIC}	54.5	11.8	19.4

Table 5: Experimental results on ORGANIZATION.

5.4 Basic Results

Table 1 shows the overall results for the baselines and ours with the name NER_{CB}. Here our system is trained as described in Algorithm 1, combining a KNN classifier and a CRF labeler, with semi-supervised learning enabled. As can be seen from Table 1, on the whole, our method significantly outperforms (with $p < 0.001$) the baselines. Tables 2-5 report the results on each entity type, indicating that our method consistently yields better results on all entity types.

5.5 Effects of KNN Classifier

Table 6 shows the performance of our method without combining the KNN classifier, denoted by NER_{CB-KNN}. A drop in performance is observed then. We further check the confidently predicted labels of the KNN classifier, which account for about 22.2% of all predications, and find that its F1 is as high as 80.2% while the baseline system based on the CRF model achieves only an F1 of 75.4%. This largely explains why the KNN classifier helps the CRF labeler. The KNN classifier is replaced with its competitors, and only a slight difference in performance is observed. We do observe that retraining KNN is obviously faster.

5.6 Effects of the CRF Labeler

Similarly, the CRF model is replaced by its alternatives. As is opposite to the finding of Ratinov and Roth (2009), the CRF model gives remarkably better results, i.e., 2.1% higher in F1 than its best followers (with $p < 0.001$). Table 7 shows the overall performance of the CRF labeler with various feature set combinations, where F_o, F_l and F_g denote the orthographic features, the lexical features and the gazetteers related features, respectively. It can be seen from Table 7 that the lexical and gazetteer related features are helpful. Other advanced features such as chunking are also explored but with no significant improvement.

5.7 Effects of Semi-supervised Learning

Table 8 compares our method with its modified version without semi-supervised learning, suggesting that semi-supervised learning considerably boosts the performance. To get more details about self-training, we evenly divide the test data into 10 parts and feed them into our method sequentially; we record the average F1 score on each part, as shown in Figure 2.

5.8 Error Analysis

Errors made by our system on the test set fall into three categories. The first kind of error, accounting for 35.5% of all errors, is largely related to slang expressions and informal abbreviations. For example, our method identifies "Cali", which actually means "California", as a PERSON in the tweet "i love Cali so much". In future, we can design a normalization

System	Pre.(%)	Rec.(%)	F1(%)
NER_{CB}	81.6	78.8	80.2
NER_{CB-KNN}	82.6	74.8	78.5

Table 6: Overall performance of our system with and without the KNN classifier, respectively.

Features	Pre.(%)	Rec.(%)	F1(%)
F_o	71.3	42.8	53.5
$F_o + F_l$	76.2	44.2	55.9
$F_o + F_g$	80.5	66.2	72.7
$F_o + F_l + F_g$	82.6	74.8	78.5

Table 7: Overview performance of the CRF labeler (combined with KNN) with different feature sets.

component to handle such slang expressions and informal abbreviations.

The second kind of error, accounting for 37.2% of all errors, is mainly attributed to the data sparseness. For example, for this tweet "come to see jaxon someday", our method mistakenly labels "jaxon" as a LOCATION, which actually denotes a PERSON. This error is understandable somehow, since this tweet is one of the earliest tweets that mention "jaxon", and at that time there was no strong evidence supporting that it represents a person. Possible solutions to these errors include continually enriching the gazetteers and aggregating additional external knowledge from other channels such as traditional news.

The last kind of error, which represents 27.3% of all errors, somehow links to the noise prone nature of tweets. Consider this tweet "wesley snipes ws cought 4 nt payin tax coz ths celebz dnt take it cirus.", in which "wesley snipes" is not identified as a PERSON but simply ignored by our method, because this tweet is too noisy to provide effective features. Tweet normalization technology seems a possible solution to alleviate this kind of error.

Features	Pre.(%)	Rec.(%)	F1(%)
NER_{CB}	81.6	78.8	80.2
NER'_{CB}	82.1	71.9	76.7

Table 8: Performance of our system with and without semi-supervised learning, respectively.

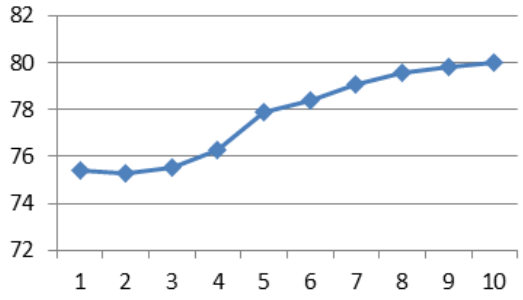


Figure 2: F1 score on 10 test data sets sequentially fed into the system, each with 600 instances. Horizontal and vertical axes represent the sequential number of the test data set and the averaged F1 score (%), respectively.

6 Conclusions and Future work

We propose a novel NER system for tweets, which combines a KNN classifier with a CRF labeler under a semi-supervised learning framework. The KNN classifier collects global information across recently labeled tweets while the CRF labeler exploits information from a single tweet and from the gazetteers. A serials of experiments show the effectiveness of our method, and particularly, show the positive effects of KNN and semi-supervised learning.

In future, we plan to further improve the performance of our method through two directions. Firstly, we hope to develop tweet normalization technology to make tweets friendlier to the NER task. Secondly, we are interested in integrating new entities from tweets or other channels into the gazetteers.

Acknowledgments

We thank Long Jiang, Changning Huang, Yunbo Cao, Dongdong Zhang, Zaiqing Nie for helpful discussions, and the anonymous reviewers for their valuable comments. We also thank Matt Callcut for his careful proofreading of an early draft of this paper.

References

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Comput. Linguist.*, 18:467–479.

Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Frederick Reiss, and Shivakumar Vaithyanathan. 2010. Domain adaptation of rule-based annotators for named-entity recognition tasks. In *EMNLP*, pages 1002–1012.

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8.

Doug Downey, Matthew Broadhead, and Oren Etzioni. 2007. Locating Complex Named Entities in Web Text. In *IJCAI*.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.*, 165(1):91–134.

Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating named entities in twitter data with crowdsourcing. In *CSLDAMT*, pages 80–88.

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested named entity recognition. In *EMNLP*, pages 141–150.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, pages 363–370.

Honglei Guo, Huijia Zhu, Zhili Guo, Xiaoxun Zhang, Xian Wu, and Zhong Su. 2009. Domain adaptation with latent semantic association for named entity recognition. In *NAACL*, pages 281–289.

Martin Jansche and Steven P. Abney. 2002. Information extraction from voicemail transcripts. In *EMNLP*, pages 320–327.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *ACL*, pages 264–271.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL*, pages 423–430.

Vijay Krishnan and Christopher D. Manning. 2006. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *ACL*, pages 1121–1128.

George R. Krupka and Kevin Hausman. 1998. Isoquest: Description of the netowl^{TM} extractor system as used in muc-7. In *MUC-7*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

Andrew Mccallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *HLT-NAACL*, pages 188–191.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *HLT-NAACL*, pages 337–342.

Einat Minkov, Richard C. Wang, and William W. Cohen. 2005. Extracting personal names from email: applying named entity recognition to informal text. In *HLT*, pages 443–450.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30:3–26.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*, pages 147–155.

Sameer Singh, Dustin Hillard, and Chris Leggetter. 2010. Minimally-supervised extraction of entities from text advertisements. In *HLT-NAACL*, pages 73–81.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *ACL*, pages 665–673.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In *HLT-NAACL*, pages 142–147.

Yefeng Wang. 2009. Annotating and recognising named entities in clinical notes. In *ACL-IJCNLP*, pages 18–26.

Dan Wu, Wee Sun Lee, Nan Ye, and Hai Leong Chieu. 2009. Domain adaptive bootstrapping for named entity recognition. In *EMNLP*, pages 1523–1532.

Kazuhiro Yoshida and Jun'ichi Tsujii. 2007. Reranking for biomedical named-entity recognition. In *BioNLP*, pages 209–216.

Tong Zhang and David Johnson. 2003. A robust risk minimization based named entity recognition system. In *HLT-NAACL*, pages 204–207.

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an hmm-based chunk tagger. In *ACL*, pages 473–480.

Lexical Normalisation of Short Text Messages: Makn Sens a #twitter

Bo Han and **Timothy Baldwin**
NICTA Victoria Research Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne
hanb@student.unimelb.edu.au tb@ldwin.net

Abstract

Twitter provides access to large volumes of data in real time, but is notoriously noisy, hampering its utility for NLP. In this paper, we target out-of-vocabulary words in short text messages and propose a method for identifying and normalising ill-formed words. Our method uses a classifier to detect ill-formed words, and generates correction candidates based on morphophonemic similarity. Both word similarity and context are then exploited to select the most probable correction candidate for the word. The proposed method doesn't require any annotations, and achieves state-of-the-art performance over an SMS corpus and a novel dataset based on Twitter.

1 Introduction

Twitter and other micro-blogging services are highly attractive for information extraction and text mining purposes, as they offer large volumes of real-time data, with around 65 millions tweets posted on Twitter per day in June 2010 (Twitter, 2010). The quality of messages varies significantly, however, ranging from high quality newswire-like text to meaningless strings. Typos, ad hoc abbreviations, phonetic substitutions, ungrammatical structures and emoticons abound in short text messages, causing grief for text processing tools (Sproat et al., 2001; Ritter et al., 2010). For instance, presented with the input *u must be talkin bout the paper but I was thinkin movies* ("You must be talking about the paper but I was thinking movies"),[1] the Stanford parser (Klein and Manning, 2003; de Marneffe et al., 2006) analyses *bout the paper* and *thinkin movies* as a clause and noun phrase, respectively, rather than a prepositional phrase and verb phrase. If there were some way of preprocessing the message to produce a more canonical lexical rendering, we would expect the quality of the parser to improve appreciably. Our aim in this paper is this task of lexical normalisation of noisy English text, with a particular focus on Twitter and SMS messages. In this paper, we will collectively refer to individual instances of typos, ad hoc abbreviations, unconventional spellings, phonetic substitutions and other causes of lexical deviation as "ill-formed words".

The message normalisation task is challenging. It has similarities with spell checking (Peterson, 1980), but differs in that ill-formedness in text messages is often intentional, whether due to the desire to save characters/keystrokes, for social identity, or due to convention in this text sub-genre. We propose to go beyond spell checkers, in performing deabbreviation when appropriate, and recovering the canonical word form of commonplace shorthands like *b4* "before", which tend to be considered beyond the remit of spell checking (Aw et al., 2006). The free writing style of text messages makes the task even more complex, e.g. with word lengthening such as *goooood* being commonplace for emphasis. In addition, the detection of ill-formed words is difficult due to noisy context.

Our objective is to restore ill-formed words to their canonical lexical forms in standard English. Through a pilot study, we compared OOV words in Twitter and SMS data with other domain corpora,

[1] Throughout the paper, we will provide a normalised version of examples as a gloss in double quotes.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 368–378,
Portland, Oregon, June 19-24, 2011.

revealing their characteristics in OOV word distribution. We found Twitter data to have an unsurprisingly long tail of OOV words, suggesting that conventional supervised learning will not perform well due to data sparsity. Additionally, many ill-formed words are ambiguous, and require context to disambiguate. For example, *Gooood* may refer to *Good* or *God* depending on context. This provides the motivation to develop a method which does not require annotated training data, but is able to leverage context for lexical normalisation. Our approach first generates a list of candidate canonical lexical forms, based on morphological and phonetic variation. Then, all candidates are ranked according to a list of features generated from noisy context and similarity between ill-formed words and candidates. Our proposed cascaded method is shown to achieve state-of-the-art results on both SMS and Twitter data.

Our contributions in this paper are as follows: (1) we conduct a pilot study on the OOV word distribution of Twitter and other text genres, and analyse different sources of non-standard orthography in Twitter; (2) we generate a text normalisation dataset based on Twitter data; (3) we propose a novel normalisation approach that exploits dictionary lookup, word similarity and word context, without requiring annotated data; and (4) we demonstrate that our method achieves state-of-the-art accuracy over both SMS and Twitter data.

2 Related work

The noisy channel model (Shannon, 1948) has traditionally been the primary approach to tackling text normalisation. Suppose the ill-formed text is T and its corresponding standard form is S, the approach aims to find $\arg\max P(S|T)$ by computing $\arg\max P(T|S)P(S)$, in which $P(S)$ is usually a language model and $P(T|S)$ is an error model. Brill and Moore (2000) characterise the error model by computing the product of operation probabilities on slice-by-slice string edits. Toutanova and Moore (2002) improve the model by incorporating pronunciation information. Choudhury et al. (2007) model the word-level text generation process for SMS messages, by considering graphemic/phonetic abbreviations and unintentional typos as hidden Markov model (HMM) state transitions and emissions, respectively (Rabiner, 1989). Cook and Stevenson (2009) expand the error model by introducing inference from different erroneous formation processes, according to the sampled error distribution. While the noisy channel model is appropriate for text normalisation, $P(T|S)$, which encodes the underlying error production process, is hard to approximate accurately. Additionally, these methods make the strong assumption that a token $t_i \in T$ only depends on $s_i \in S$, ignoring the context around the token, which could be utilised to help in resolving ambiguity.

Statistical machine translation (SMT) has been proposed as a means of context-sensitive text normalisation, by treating the ill-formed text as the source language, and the standard form as the target language. For example, Aw et al. (2006) propose a phrase-level SMT SMS normalisation method with bootstrapped phrase alignments. SMT approaches tend to suffer from a critical lack of training data, however. It is labor intensive to construct an annotated corpus to sufficiently cover ill-formed words and context-appropriate corrections. Furthermore, it is hard to harness SMT for the lexical normalisation problem, as even if phrase-level re-ordering is suppressed by constraints on phrase segmentation, word-level re-orderings within a phrase are still prevalent.

Some researchers have also formulated text normalisation as a speech recognition problem. For example, Kobus et al. (2008) firstly convert input text tokens into phonetic tokens and then restore them to words by phonetic dictionary lookup. Beaufort et al. (2010) use finite state methods to perform French SMS normalisation, combining the advantages of SMT and the noisy channel model. Kaufmann and Kalita (2010) exploit a machine translation approach with a preprocessor for syntactic (rather than lexical) normalisation.

Predominantly, however, these methods require large-scale annotated training data, limiting their adaptability to new domains or languages. In contrast, our proposed method doesn't require annotated data. It builds on the work on SMS text normalisation, and adapts it to Twitter data, exploiting multiple data sources for normalisation.

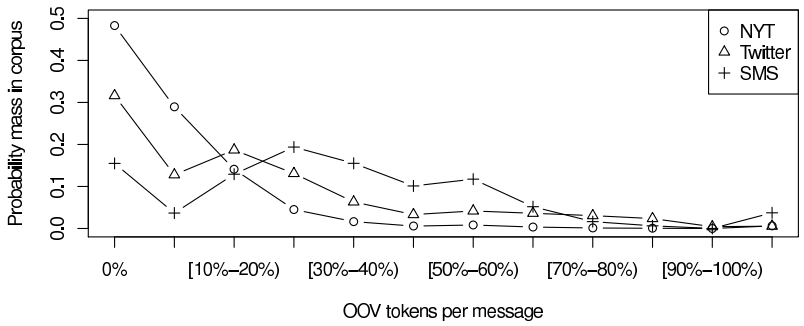


Figure 1: Out-of-vocabulary word distribution in English Gigaword (NYT), Twitter and SMS data

3 Scoping Text Normalisation

3.1 Task Definition of Lexical Normalisation

We define the task of text normalisation to be a mapping from "ill-formed" OOV lexical items to their standard lexical forms, focusing exclusively on English for the purposes of this paper. We define the task as follows:

- only OOV words are considered for normalisation;
- normalisation must be to a single-token word, meaning that we would normalise *smokin* to *smoking*, but not *imo* to *in my opinion*; a side-effect of this is to permit lower-register contractions such as *gonna* as the canonical form of *gunna* (given that *going to* is out of scope as a normalisation candidate, on the grounds of being multi-token).

Given this definition, our first step is to identify candidate tokens for lexical normalisation, where we examine all tokens that consist of alphanumeric characters, and categorise them into in-vocabulary (IV) and out-of-vocabulary (OOV) words, relative to a dictionary. The OOV word definition is somewhat rough, because it includes neologisms and proper nouns like *hopeable* or *WikiLeaks* which have not made their way into the dictionary. However, it greatly simplifies the candidate identification task, at the cost of pushing complexity downstream to the word detection task, in that we need to explicitly distinguish between correct OOV words and ill-formed OOV words such as typos (e.g. *earthquak* "earthquake"), register-specific single-word abbreviations (e.g. *lv* "love"), and phonetic substitutions (e.g. *2morrow* "tomorrow").

An immediate implication of our task definition is that ill-formed words which happen to coincide with an IV word (e.g. the misspelling of *can't* as *cant*) are outside the scope of this research. We also consider that deabbreviation largely falls outside the scope of text normalisation, as abbreviations can be formed freely in standard English. Note that single-word abbreviations such as *govt* "government" are very much within the scope of lexical normalisation, as they are OOV and match to a single token in their standard lexical form.

Throughout this paper, we use the GNU aspell dictionary (v0.60.6)[2] to determine whether a token is OOV. In tokenising the text, hyphenanted tokens and tokens containing apostrophes (e.g. *take-off* and *won't*, resp.) are treated as a single token. Twitter mentions (e.g. *@twitter*), hashtags (e.g. *#twitter*) and urls (e.g. *twitter.com*) are excluded from consideration for normalisation, but left in situ for context modelling purposes. Dictionary lookup of Internet slang is performed relative to a dictionary of 5021 items collected from the Internet.[3]

3.2 OOV Word Distribution and Types

To get a sense of the relative need for lexical normalisation, we perform analysis of the distribution of OOV words in different text types. In particular, we calculate the proportion of OOV tokens per message (or sentence, in the case of edited text), bin the messages according to the OOV token proportion, and plot the probability mass contained in each bin for a given text type. The three corpora we compare

[2]We remove all one character tokens, except *a* and *I*, and treat *RT* as an IV word.

[3]`http://www.noslang.com`

are the New York Times (NYT),[4] SMS,[5] and Twitter.[6] The results are presented in Figure 1.

Both SMS and Twitter have a relatively flat distribution, with Twitter having a particularly large tail: around 15% of tweets have 50% or more OOV tokens. This has implications for any context modelling, as we cannot rely on having only isolated occurrences of OOV words. In contrast, NYT shows a more Zipfian distribution, despite the large number of proper names it contains.

While this analysis confirms that Twitter and SMS are similar in being heavily laden with OOV tokens, it does not shed any light on the relative similarity in the makeup of OOV tokens in each case. To further analyse the two data sources, we extracted the set of OOV terms found exclusively in SMS and Twitter, and analysed each. Manual analysis of the two sets revealed that most OOV words found only in SMS were personal names. The Twitter-specific set, on the other hand, contained a heterogeneous collection of ill-formed words and proper nouns. This suggests that Twitter is a richer/noisier data source, and that text normalisation for Twitter needs to be more nuanced than for SMS.

To further analyse the ill-formed words in Twitter, we randomly selected 449 tweets and manually analysed the sources of lexical variation, to determine the phenomena that lexical normalisation needs to deal with. We identified 254 token instances of lexical normalisation, and broke them down into categories, as listed in Table 1. "Letter" refers to instances where letters are missing or there are extraneous letters, but the lexical correspondence to the target word form is trivially accessible (e.g. *shuld* "should"). "Number Substitution" refers to instances of letter–number substitution, where numbers have been substituted for phonetically-similar sequences of letters (e.g. *4* "for"). "Letter&Number" refers to instances which have both extra/missing letters and number substitution (e.g. *b4* "before"). "Slang" refers to instances of Internet slang (e.g. *lol* "laugh out loud"), as found in a slang dictionary (see Section 3.1). "Other" is the remainder of the instances, which is predominantly made up of occurrences of spaces having being deleted between words (e.g. *sucha* "such a"). If a given instance belongs to multiple error categories (e.g. "Letter&Number" and it is also found in a slang dictionary), we classify it into the higher-occurring category in Table 1.

Category	Ratio
Letter&Number	2.36%
Letter	72.44%
Number Substitution	2.76%
Slang	12.20%
Other	10.24%

Table 1: Ill-formed word distribution

From Table 1, it is clear that "Letter" accounts for the majority of ill-formed words in Twitter, and that most ill-formed words are based on morphophonemic variations. This empirical finding assists in shaping our strategy for lexical normalisation.

4 Lexical normalisation

Our proposed lexical normalisation strategy involves three general steps: (1) confusion set generation, where we identify normalisation candidates for a given word; (2) ill-formed word identification, where we classify a word as being ill-formed or not, relative to its confusion set; and (3) candidate selection, where we select the standard form for tokens which have been classified as being ill formed. In confusion set generation, we generate a set of IV normalisation candidates for each OOV word type based on morphophonemic variation. We call this set the confusion set of that OOV word, and aim to include all feasible normalisation candidates for the word type in the confusion set. The confusion candidates are then filtered for each token occurrence of a given OOV word, based on their local context fit with a language model.

4.1 Confusion Set Generation

Revisiting our manual analysis from Section 3.2, most ill-formed tokens in Twitter are morphophonemically derived. First, inspired by Kaufmann and Kalita (2010), any repititions of more than 3 letters are reduced back to 3 letters (e.g. *cooool* is re-

[4]Based on 44 million sentences from English Gigaword.

[5]Based on 12.6 thousand SMS messages from How and Kan (2005) and Choudhury et al. (2007).

[6]Based on 1.37 million tweets collected from the Twitter streaming API from Aug to Oct 2010, and filtered for monolingual English messages; see Section 5.1 for details of the language filtering methodology.

Criterion	Recall	Average Candidates
$T_c \leq 1$	40.4%	24
$T_c \leq 2$	76.6%	240
$T_p = 0$	55.4%	65
$T_p \leq 1$	83.4%	1248
$T_p \leq 2$	91.0%	9694
$T_c \leq 2 \vee T_p \leq 1$	88.8%	1269
$T_c \leq 2 \vee T_p \leq 2$	92.7%	9515

Table 2: Recall and average number of candidates for different confusion set generation strategies

duced to *coool*). Second, IV words within a threshold T_c character edit distance of the given OOV word are calculated, as is widely used in spell checkers. Third, the double metaphone algorithm (Philips, 2000) is used to decode the pronunciation of all IV words, and IV words within a threshold T_p edit distance of the given OOV word under phonemic transcription, are included in the confusion set; this allows us to capture OOV words such as *earthquick* "earthquake". In Table 2, we list the recall and average size of the confusion set generated by the final two strategies with different threshold settings, based on our evaluation dataset (see Section 5.1).

The recall for lexical edit distance with $T_c \leq 2$ is moderately high, but it is unable to detect the correct candidate for about one quarter of words. The combination of the lexical and phonemic strategies with $T_c \leq 2 \vee T_p \leq 2$ is more impressive, but the number of candidates has also soared. Note that increasing the edit distance further in both cases leads to an explosion in the average number of candidates, with serious computational implications for downstream processing. Thankfully, $T_c \leq 2 \vee T_p \leq 1$ leads to an extra increment in recall to 88.8%, with only a slight increase in the average number of candidates. Based on these results, we use $T_c \leq 2 \vee T_p \leq 1$ as the basis for confusion set generation.

Examples of ill-formed words where we are unable to generate the standard lexical form are clippings such as *fav* "favourite" and *convo* "conversation".

In addition to generating the confusion set, we rank the candidates based on a trigram language model trained over 1.5GB of clean Twitter data, i.e. tweets which consist of all IV words: despite the prevalence of OOV words in Twitter, the sheer volume of the data means that it is relatively easy to collect large amounts of all-IV messages. To train the language model, we used SRILM (Stolcke, 2002) with the `-<unk>` option. If we truncate the ranking to the top 10% of candidates, the recall drops back to 84% with a 90% reduction in candidates.

4.2 Ill-formed Word Detection

The next step is to detect whether a given OOV word in context is actually an ill-formed word or not, relative to its confusion set. To the best of our knowledge, we are the first to target the task of ill-formed word detection in the context of short text messages, although related work exists for text with lower relative occurrences of OOV words (Izumi et al., 2003; Sun et al., 2007). Due to the noisiness of the data, it is impractical to use full-blown syntactic or semantic features. The most direct source of evidence is IV words around an OOV word. Inspired by work on labelled sequential pattern extraction (Sun et al., 2007), we exploit large-scale edited corpus data to construct dependency-based features.

First, we use the Stanford parser (Klein and Manning, 2003; de Marneffe et al., 2006) to extract dependencies from the NYT corpus (see Section 3.2). For example, from a sentence such as *One obvious difference is the way they look*, we would extract dependencies such as `rcmod(way-6,look-8)` and `nsubj(look-8,they-7)`. We then transform the dependencies into relational features for each OOV word. Assuming that *way* were an OOV word, e.g., we would extract dependencies of the form `(look,way,+2)`, indicating that *look* occurs 2 words after *way*. We choose dependencies to represent context because they are an effective way of capturing key relationships between words, and similar features can easily be extracted from tweets. Note that we don't record the dependency type here, because we have no intention of dependency parsing text messages, due to their noisiness and the volume of the data. The counts of dependency forms are combined together to derive a confidence score, and the scored dependencies are stored in a dependency bank.

Given the dependency-based features, a linear kernel SVM classifier (Fan et al., 2008) is trained on clean Twitter data, i.e. the subset of Twitter messages without OOV words. Each word is repre-

sented by its IV words within a context window of three words to either side of the target word, together with their relative positions in the form of `(word1,word2,position)` tuples, and their score in the dependency bank. These form the positive training exemplars. Negative exemplars are automatically constructed by replacing target words with highly-ranked candidates from their confusion set. Note that the classifier does not require any hand annotation, as all training exemplars are constructed automatically.

To predict whether a given OOV word is ill-formed, we form an exemplar for each of its confusion candidates, and extract `(word1,word2,position)` features. If all its candidates are predicted to be negative by the model, we mark it as correct; otherwise, we treat it as ill-formed, and pass all candidates (not just positively-classified candidates) on to the candidate selection step. For example, given the message *way yu lookin shuld be a sin* and the OOV word *lookin*, we would generate context features for each candidate word such as `(way,looking,-2)`, and classify each such candidate.

In training, it is possible for the exact same feature vector to occur as both positive and negative exemplars. To prevent positive exemplars being contaminated from the automatic generation, we remove all negative instances in such cases. The `(word1,word2,position)` features are sparse and sometimes lead to conservative results in ill-formed word detection. That is, without valid features, the SVM classifier tends to label uncertain cases as correct rather than ill-formed words. This is arguably the right approach to normalisation, in choosing to under- rather than over-normalise in cases of uncertainty.

As the context for a target word often contains OOV words which don't occur in the dependency bank, we expand the dependency features to include context tokens up to a phonemic edit distance of 1 from context tokens in the dependency bank. In this way, we generate dependency-based features for context words such as *seee* "see" in `(seee, flm, +2)` (based on the target word *flm* in the context of *flm to seee*). However, expanded dependency features may introduce noise, and we therefore introduce expanded dependency weights $w_d \in \{0.0, 0.5, 1.0\}$ to ameliorate the effects of noise: a weight of $w_d = 0.0$ means no expansion, while 1.0 means expanded dependencies are indistinguishable from non-expanded (strict match) dependencies.

We separately introduce a threshold $t_d \in \{1, 2, ..., 10\}$ on the number of positive predictions returned by the detection classifier over the set of normalisation candidates for a given OOV token: the token is considered to be ill-formed iff t_d or more candidates are positively classified, i.e. predicted to be correct candidates.

4.3 Candidate Selection

For OOV words which are predicted to be ill-formed, we select the most likely candidate from the confusion set as the basis of normalisation. The final selection is based on the following features, in line with previous work (Wong et al., 2006; Cook and Stevenson, 2009).

Lexical edit distance, phonemic edit distance, prefix substring, suffix substring, and the longest common subsequence (LCS) are exploited to capture morphophonemic similarity. Both lexical and phonemic edit distance (ED) are normalised by the reciprocal of $exp(ED)$. The prefix and suffix features are intended to capture the fact that leading and trailing characters are frequently dropped from words, e.g. in cases such as *ish* and *talkin*. We calculate the ratio of the LCS over the maximum string length between ill-formed word and the candidate, since the ill-formed word can be either longer or shorter than (or the same size as) the standard form. For example, *mve* can be restored to either *me* or *move*, depending on context. We normalise these ratios following Cook and Stevenson (2009).

For context inference, we employ both language model- and dependency-based frequency features. Ranking by language model score is intuitively appealing for candidate selection, but our trigram model is trained only on clean Twitter data and ill-formed words often don't have sufficient context for the language model to operate effectively, as in *bt* "but" in *say 2 sum1 bt nt gonna say* "say to someone but not going to say". To consolidate the context modelling, we obtain dependencies from the dependency bank used in ill-formed word detection. Although text messages are of a different genre to edited newswire text, we assume they form similar

dependencies based on the common goal of getting across the message effectively. The dependency features can be used in noisy contexts and are robust to the effects of other ill-formed words, as they do not rely on contiguity. For example, *uz* "use" in *i did #tt uz me and yu*, dependencies can capture relationships like `aux(use-4, do-2)`, which is beyond the capabilities of the language model due to the hashtag being treated as a correct OOV word.

5 Experiments

5.1 Dataset and baselines

The aim of our experiments is to compare the effectiveness of different methodologies over text messages, based on two datasets: (1) an SMS corpus (Choudhury et al., 2007); and (2) a novel Twitter dataset developed as part of this research, based on a random sampling of 549 English tweets. The English tweets were annotated by three independent annotators. All OOV words were pre-identified, and the annotators were requested to determine: (a) whether each OOV word was ill-formed or not; and (b) what the standard form was for ill-formed words, subject to the task definition outlined in Section 3.1. The total number of ill-formed words contained in the SMS and Twitter datasets were 3849 and 1184, respectively.[7]

The language filtering of Twitter to automatically identify English tweets was based on the language identification method of Baldwin and Lui (2010), using the EuroGOV dataset as training data, a mixed unigram/bigram/trigram byte feature representation, and a skew divergence nearest prototype classifier.

We reimplemented the state-of-art noisy channel model of Cook and Stevenson (2009) and SMT approach of Aw et al. (2006) as benchmark methods. We implement the SMT approach in Moses (Koehn et al., 2007), with synthetic training and tuning data of 90,000 and 1000 sentence pairs, respectively. This data is randomly sampled from the 1.5GB of clean Twitter data, and errors are generated according to distribution of SMS corpus. The 10-fold cross-validated BLEU score (Papineni et al., 2002) over this data is 0.81.

[7]The Twitter dataset is available at `http://www.csse.unimelb.edu.au/research/lt/resources/lexnorm/`.

In addition to comparing our method with competitor methods, we also study the contribution of different feature groups. We separately compare dictionary lookup over our Internet slang dictionary, the contextual feature model, and the word similarity feature model, as well as combinations of these three.

5.2 Evaluation metrics

The evaluation of lexical normalisation consists of two stages (Hirst and Budanitsky, 2005): (1) ill-formed word detection, and (2) candidate selection. In terms of detection, we want to make sense of how well the system can identify ill-formed words and leave correct OOV words untouched. This step is crucial to further normalisation, because if correct OOV words are identified as ill-formed, the candidate selection step can never be correct. Conversely, if an ill-formed word is predicted to be correct, the candidate selection will have no chance to normalise it.

We evaluate detection performance by token-level precision, recall and F-score ($\beta = 1$). Previous work over the SMS corpus has assumed perfect ill-formed word detection and focused only on the candidate selection step, so we evaluate ill-formed word detection for the Twitter data only.

For candidate selection, we once again evaluate using token-level precision, recall and F-score. Additionally, we evaluate using the BLEU score over the normalised form of each message, as the SMT method can lead to perturbations of the token stream, vexing standard precision, recall and F-score evaluation.

5.3 Results and Analysis

First, we test the impact of the w_d and t_d values on ill-formed word detection effectiveness, based on dependencies from either the Spinn3r blog corpus (Blog: Burton et al. (2009)) or NYT. The results for precision, recall and F-score are presented in Figure 2.

Some conclusions can be drawn from the graphs. First, higher detection threshold values (t_d) give better precision but lower recall. Generally, as t_d is raised from 1 to 10, the precision improves slightly but recall drops dramatically, with the net effect that the F-score decreases monotonically. Thus, we use a

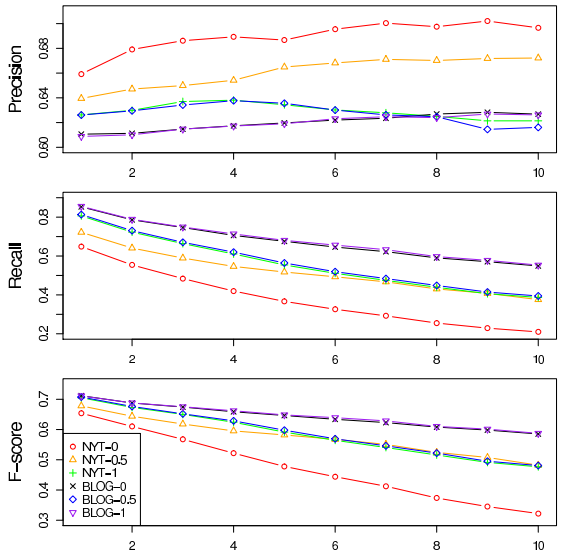


Figure 2: Ill-formed word detection precision, recall and F-score

smaller threshold, i.e. $t_d = 1$. Second, there are differences between the two corpora, with dependencies from the Blog corpus producing slightly lower precision but higher recall, compared with the NYT corpus. The lower precision for the Blog corpus appears to be due to the text not being as clean as NYT, introducing parser errors. Nevertheless, the difference in F-score between the two corpora is insignificant. Third, we obtain the best results, especially in terms of precision, for $w_d = 0.5$, i.e. with expanded dependencies, but penalised relative to non-expanded dependencies.

Overall, the best F-score is 71.2%, with a precision of 61.1% and recall of 85.3%, obtained over the Blog corpus with $t_d = 1$ and $w_d = 0.5$. Clearly there is significant room for immprovements in these results. We leave the improvement of ill-formed word detection for future work, and perform evaluation of candidate selection for Twitter assuming perfect ill-formed word detection, as for the SMS data.

From Table 3, we see that the general performance of our proposed method on Twitter is better than that on SMS. To better understand this trend, we examined the annotations in the SMS corpus, and found them to be looser than ours, because they have different task specifications than our lexical normalisation. In our annotation, the annotators only normalised ill-formed word if they had high confidence of how to normalise, as with *talkin* "talking". For ill-formed words where they couldn't be certain of the standard form, the tokens were left untouched. However, in the SMS corpus, annotations such as *sammis* "same" are also included. This leads to a performance drop for our method over the SMS corpus.

The noisy channel method of Cook and Stevenson (2009) shares similar features with word similarity ("WS"), However, when word similarity and context support are combined ("WS+CS"), our method outperforms the noisy channel method by about 7% and 12% in F-score over SMS and Twitter corpora, respectively. This can be explained as follows. First, the Cook and Stevenson (2009) method is type-based, so all token instances of a given ill-formed word will be normalised identically. In the Twitter data, however, the same word can be normalised differently depending on context, e.g. *hw* "how" in *so hw many time remaining so I can calculate it?* vs. *hw* "homework" in *I need to finish my hw first*. Second, the noisy channel method was developed specifically for SMS normalisation, in which clipping is the most prevalent form of lexical variation, while in the Twitter data, we commonly have instances of word lengthening for emphasis, such as *moviiie* "movie". Having said this, our method is superior to the noisy channel method over both the SMS and Twitter data.

The SMT approach is relatively stable on the two datasets, but well below the performance of our method. This is due to the limitations of the training data: we obtain the ill-formed words and their standard forms from the SMS corpus, but the ill-formed words in the SMS corpus are not sufficient to cover those in the Twitter data (and we don't have sufficient Twitter data to train the SMT method directly). Thus, novel ill-formed words are missed in normalisation. This shows the shortcoming of supervised data-driven approaches that require annotated data to cover all possibilities of ill-formed words in Twitter.

The dictionary lookup method ("DL") unsurprisingly achieves the best precision, but the recall on Twitter is not competitive. Consequently, the Twitter normalisation cannot be tackled with dictionary lookup alone, although it is an effective preprocessing strategy when combined with more ro-

Dataset	*Evaluation*	*NC*	*MT*	*DL*	*WS*	*CS*	*WS+CS*	*DL+WS+CS*
SMS	Precision	0.465	—	**0.927**	0.521	0.116	0.532	0.756
	Recall	0.464	—	0.597	0.520	0.116	0.531	**0.754**
	F-score	0.464	—	0.726	0.520	0.116	0.531	**0.755**
	BLEU	0.746	0.700	0.801	0.764	0.612	0.772	**0.876**
Twitter	Precision	0.452	—	**0.961**	0.551	0.194	0.571	0.753
	Recall	0.452	—	0.460	0.551	0.194	0.571	**0.753**
	F-score	0.452	—	0.622	0.551	0.194	0.571	**0.753**
	BLEU	0.857	0.728	0.861	0.878	0.797	0.884	**0.934**

Table 3: Candidate selection effectiveness on different datasets (*NC* = noisy channel model (Cook and Stevenson, 2009); *MT* = SMT (Aw et al., 2006); *DL* = dictionary lookup; *WS* = word similarity; *CS* = context support)

bust techniques such as our proposed method, and effective at capturing common abbreviations such as *gf* "girlfriend".

Of the component methods proposed in this research, word similarity ("WS") achieves higher precision and recall than context support ("CS"), signifying that many of the ill-formed words emanate from morphophonemic variations. However, when combined with word similarity features, context support improves over the basic method at a level of statistical significance (based on randomised estimation, $p < 0.05$: Yeh (2000)), indicating the complementarity of the two methods, especially on Twitter data. The best F-score is achieved when combining dictionary lookup, word similarity and context support ("DL+WS+CS"), in which ill-formed words are first looked up in the slang dictionary, and only if no match is found do we apply our normalisation method.

We found several limitations in our proposed approach by analysing the output of our method. First, not all ill-formed words offer useful context. Some highly noisy tweets contain almost all misspellings and unique symbols, and thus no context features can be extracted. This also explains why "CS" features often fail. For such cases, the method falls back to context-independent normalisation. We found that only 32.6% ill-formed words have all IV words in their context windows. Moreover, the IV words may not occur in the dependency bank, further decreasing the effectiveness of context support features. Second, the different features are linearly combined, where a weighted combination is likely to give better results, although it also requires a certain amount of well-sampled annotations for tuning.

6 Conclusion and Future Work

In this paper, we have proposed the task of lexical normalisation for short text messages, as found in Twitter and SMS data. We found that most ill-formed words are based on morphophonemic variation and proposed a cascaded method to detect and normalise ill-formed words. Our ill-formed word detector requires no explicit annotations, and the dependency-based features were shown to be somewhat effective, however, there was still a lot of room for improvement at ill-formed word detection. In normalisation, we compared our method with two benchmark methods from the literature, and achieved that highest F-score and BLEU score by integrating dictionary lookup, word similarity and context support modelling.

In future work, we propose to pursue a number of directions. First, we plan to improve our ill-formed word detection classifier by introducing an OOV word whitelist. Furthermore, we intend to alleviate noisy contexts with a bootstrapping approach, in which ill-formed words with high confidence and no ambiguity will be replaced by their standard forms, and fed into the normalisation model as new training data.

Acknowledgements

NICTA is funded by the Australian government as represented by Department of Broadband, Communication and Digital Economy, and the Australian Research Council through the ICT centre of Excellence programme.

References

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for SMS text normalization. In *Proceedings of the 21st International Con-*

ference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, pages 33–40, Sydney, Australia.

Timothy Baldwin and Marco Lui. 2010. Language identification: The long and the short of the matter. In *HLT '10: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 229–237, Los Angeles, USA.

Richard Beaufort, Sophie Roekhaut, Louise-Amélie Cougnon, and Cédrick Fairon. 2010. A hybrid rule/model-based finite-state framework for normalizing SMS messages. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 770–779, Uppsala, Sweden.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293, Hong Kong.

Kevin Burton, Akshay Java, and Ian Soboroff. 2009. The ICWSM 2009 Spinn3r Dataset. In *Proceedings of the Third Annual Conference on Weblogs and Social Media*, San Jose, USA.

Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupam Basu. 2007. Investigation and modeling of the structure of texting language. *International Journal on Document Analysis and Recognition*, 10:157–174.

Paul Cook and Suzanne Stevenson. 2009. An unsupervised model for text message normalization. In *CALC '09: Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 71–78, Boulder, USA.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Graeme Hirst and Alexander Budanitsky. 2005. Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11:87–111.

Yijue How and Min-Yen Kan. 2005. Optimizing predictive text entry for short message service on mobile phones. In *Human Computer Interfaces International (HCII 05)*, Las Vegas, USA.

Emi Izumi, Kiyotaka Uchimoto, Toyomi Saiga, Thepchai Supnithi, and Hitoshi Isahara. 2003. Automatic error detection in the Japanese learners' English spoken data. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 2*, pages 145–148, Sapporo, Japan.

Joseph Kaufmann and Jugal Kalita. 2010. Syntactic normalization of Twitter messages. In *International Conference on Natural Language Processing*, Kharagpur, India.

Dan Klein and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, pages 3–10, Whistler, Canada.

Catherine Kobus, Franois Yvon, and Graldine Damnati. 2008. Transcrire les SMS comme on reconnat la parole. In *Actes de la Confrence sur le Traitement Automatique des Langues (TALN'08)*, pages 128–138.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Philadelphia, USA.

James L. Peterson. 1980. Computer programs for detecting and correcting spelling errors. *Commun. ACM*, 23:676–687, December.

Lawrence Philips. 2000. The double metaphone search algorithm. *C/C++ Users Journal*, 18:38–43.

Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of Twitter conversations. In *HLT '10: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180, Los Angeles, USA.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656.

Richard Sproat, Alan W. Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. 2001. Normalization of non-standard words. *Computer Speech and Language*, 15(3):287 – 333.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *International Conference on Spo-*

ken Language Processing, pages 901–904, Denver, USA.

Guihua Sun, Gao Cong, Xiaohua Liu, Chin-Yew Lin, and Ming Zhou. 2007. Mining sequential patterns and tree patterns to detect erroneous sentences. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 81–88, Prague, Czech Republic.

Kristina Toutanova and Robert C. Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 144–151, Philadelphia, USA.

Twitter. 2010. Big goals, big game, big records. `http://blog.twitter.com/2010/06/big-goals-big-game-big-records.html`. Retrieved 4 August 2010.

Wilson Wong, Wei Liu, and Mohammed Bennamoun. 2006. Integrated scoring for spelling error correction, abbreviation expansion and case restoration in dirty text. In *Proceedings of the Fifth Australasian Conference on Data Mining and Analytics*, pages 83–89, Sydney, Australia.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, COLING '00, pages 947–953, Saarbrücken, Germany.

Topical Keyphrase Extraction from Twitter

Wayne Xin Zhao[†] **Jing Jiang**[‡] **Jing He**[†] **Yang Song**[†] **Palakorn Achananuparp**[‡]
Ee-Peng Lim[‡] **Xiaoming Li**[†]
[†]School of Electronics Engineering and Computer Science, Peking University
[‡]School of Information Systems, Singapore Management University
{batmanfly,peaceful.he,songyangmagic}@gmail.com,
{jingjiang,eplim,palakorna}@smu.edu.sg, lxm@pku.edu.cn

Abstract

Summarizing and analyzing Twitter content is an important and challenging task. In this paper, we propose to extract topical keyphrases as one way to summarize Twitter. We propose a context-sensitive topical PageRank method for keyword ranking and a probabilistic scoring function that considers both relevance and interestingness of keyphrases for keyphrase ranking. We evaluate our proposed methods on a large Twitter data set. Experiments show that these methods are very effective for topical keyphrase extraction.

1 Introduction

Twitter, a new microblogging website, has attracted hundreds of millions of users who publish short messages (a.k.a. *tweets*) on it. They either publish original tweets or *retweet* (i.e. forward) others' tweets if they find them interesting. Twitter has been shown to be useful in a number of applications, including tweets as social sensors of real-time events (Sakaki et al., 2010), the sentiment prediction power of Twitter (Tumasjan et al., 2010), etc. However, current explorations are still in an early stage and our understanding of Twitter content still remains limited. How to automatically understand, extract and summarize useful Twitter content has therefore become an important and emergent research topic.

In this paper, we propose to extract keyphrases as a way to summarize Twitter content. Traditionally, keyphrases are defined as a short list of terms to summarize the topics of a document (Turney, 2000). It can be used for various tasks such as document summarization (Litvak and Last, 2008) and indexing (Li et al., 2004). While it appears natural to use keyphrases to summarize Twitter content, compared with traditional text collections, keyphrase extraction from Twitter is more challenging in at least two aspects: 1) Tweets are much shorter than traditional articles and not all tweets contain useful information; 2) Topics tend to be more diverse in Twitter than in formal articles such as news reports.

So far there is little work on keyword or keyphrase extraction from Twitter. Wu et al. (2010) proposed to automatically generate personalized tags for Twitter users. However, user-level tags may not be suitable to summarize the overall Twitter content within a certain period and/or from a certain group of people such as people in the same region. Existing work on keyphrase extraction identifies keyphrases from either individual documents or an entire text collection (Turney, 2000; Tomokiyo and Hurst, 2003). These approaches are not immediately applicable to Twitter because it does not make sense to extract keyphrases from a single tweet, and if we extract keyphrases from a whole tweet collection we will mix a diverse range of topics together, which makes it difficult for users to follow the extracted keyphrases.

Therefore, in this paper, we propose to study the novel problem of extracting *topical keyphrases* for summarizing and analyzing Twitter content. In other words, we extract and organize keyphrases by topics learnt from Twitter. In our work, we follow the standard three steps of keyphrase extraction, namely, keyword ranking, candidate keyphrase generation

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 379–388,
Portland, Oregon, June 19-24, 2011.

and keyphrase ranking. For keyword ranking, we modify the Topical PageRank method proposed by Liu et al. (2010) by introducing topic-sensitive score propagation. We find that topic-sensitive propagation can largely help boost the performance. For keyphrase ranking, we propose a principled probabilistic phrase ranking method, which can be flexibly combined with any keyword ranking method and candidate keyphrase generation method. Experiments on a large Twitter data set show that our proposed methods are very effective in topical keyphrase extraction from Twitter. Interestingly, our proposed keyphrase ranking method can incorporate users' interests by modeling the *retweet* behavior. We further examine what topics are suitable for incorporating users' interests for topical keyphrase extraction.

To the best of our knowledge, our work is the first to study how to extract keyphrases from microblogs. We perform a thorough analysis of the proposed methods, which can be useful for future work in this direction.

2 Related Work

Our work is related to unsupervised keyphrase extraction. Graph-based ranking methods are the state of the art in unsupervised keyphrase extraction. Mihalcea and Tarau (2004) proposed to use TextRank, a modified PageRank algorithm to extract keyphrases. Based on the study by Mihalcea and Tarau (2004), Liu et al. (2010) proposed to decompose a traditional random walk into multiple random walks specific to various topics. Language modeling methods (Tomokiyo and Hurst, 2003) and natural language processing techniques (Barker and Cornacchia, 2000) have also been used for unsupervised keyphrase extraction. Our keyword extraction method is mainly based on the study by Liu et al. (2010). The difference is that we model the score propagation with topic context, which can lower the effect of noise, especially in microblogs.

Our work is also related to automatic topic labeling (Mei et al., 2007). We focus on extracting topical keyphrases in microblogs, which has its own challenges. Our method can also be used to label topics in other text collections.

Another line of relevant research is Twitter-related text mining. The most relevant work is by Wu et al. (2010), who directly applied TextRank (Mihalcea and Tarau, 2004) to extract keywords from tweets to tag users. Topic discovery from Twitter is also related to our work (Ramage et al., 2010), but we further extract keyphrases from each topic for summarizing and analyzing Twitter content.

3 Method

3.1 Preliminaries

Let $\mathcal{U}$ be a set of Twitter users. Let $\mathcal{C} = \{\{d_{u,m}\}_{m=1}^{M_u}\}_{u \in \mathcal{U}}$ be a collection of tweets generated by $\mathcal{U}$, where M_u is the total number of tweets generated by user u and $d_{u,m}$ is the m-th tweet of u. Let $\mathcal{V}$ be the vocabulary. $d_{u,m}$ consists of a sequence of words $(w_{u,m,1}, w_{u,m,2}, \ldots, w_{u,m,N_{u,m}})$ where $N_{u,m}$ is the number of words in $d_{u,m}$ and $w_{u,m,n} \in \mathcal{V}$ $(1 \leq n \leq N_{u,m})$. We also assume that there is a set of topics $\mathcal{T}$ over the collection $\mathcal{C}$.

Given $\mathcal{T}$ and $\mathcal{C}$, topical keyphrase extraction is to discover a list of keyphrases for each topic $t \in \mathcal{T}$. Here each keyphrase is a sequence of words.

To extract keyphrases, we first identify topics from the Twitter collection using topic models (Section 3.2). Next for each topic, we run a topical PageRank algorithm to rank keywords and then generate candidate keyphrases using the top ranked keywords (Section 3.3). Finally, we use a probabilistic model to rank the candidate keyphrases (Section 3.4).

3.2 Topic discovery

We first describe how we discover the set of topics $\mathcal{T}$. Author-topic models have been shown to be effective for topic modeling of microblogs (Weng et al., 2010; Hong and Davison, 2010). In Twitter, we observe an important characteristic of tweets: tweets are short and a single tweet tends to be about a single topic. So we apply a modified author-topic model called Twitter-LDA introduced by Zhao et al. (2011), which assumes a single topic assignment for an entire tweet.

The model is based on the following assumptions. There is a set of topics $\mathcal{T}$ in Twitter, each represented by a word distribution. Each user has her topic interests modeled by a distribution over the topics. When a user wants to write a tweet, she first chooses a topic based on her topic distribution. Then she chooses a

1. Draw $\phi^{\mathcal{B}} \sim \text{Dir}(\beta), \pi \sim \text{Dir}(\gamma)$
2. For each topic $t \in \mathcal{T}$,
   (a) draw $\phi^t \sim \text{Dir}(\beta)$
3. For each user $u \in \mathcal{U}$,
   (a) draw $\theta^u \sim \text{Dir}(\alpha)$
   (b) for each tweet $d_{u,m}$
      i. draw $z_{u,m} \sim \text{Multi}(\theta^u)$
      ii. for each word $w_{u,m,n}$
         A. draw $y_{u,m,n} \sim \text{Bernoulli}(\pi)$
         B. draw $w_{u,m,n} \sim \text{Multi}(\phi^{\mathcal{B}})$ if $y_{u,m,n} = 0$ and $w_{u,m,n} \sim \text{Multi}(\phi^{z_{u,m}})$ if $y_{u,m,n} = 1$

Figure 1: The generation process of tweets.

bag of words one by one based on the chosen topic. However, not all words in a tweet are closely related to the topic of that tweet; some are background words commonly used in tweets on different topics. Therefore, for each word in a tweet, the user first decides whether it is a background word or a topic word and then chooses the word from its respective word distribution.

Formally, let ϕ^t denote the word distribution for topic t and $\phi^{\mathcal{B}}$ the word distribution for background words. Let θ^u denote the topic distribution of user u. Let π denote a Bernoulli distribution that governs the choice between background words and topic words. The generation process of tweets is described in Figure 1. Each multinomial distribution is governed by some symmetric Dirichlet distribution parameterized by α, β or γ.

3.3 Topical PageRank for Keyword Ranking

Topical PageRank was introduced by Liu et al. (2010) to identify keywords for future keyphrase extraction. It runs topic-biased PageRank for each topic separately and boosts those words with high relevance to the corresponding topic. Formally, the topic-specific PageRank scores can be defined as follows:

$$R_t(w_i) = \lambda \sum_{j:w_j \to w_i} \frac{e(w_j, w_i)}{O(w_j)} R_t(w_j) + (1-\lambda) P_t(w_i), \quad (1)$$

where $R_t(w)$ is the topic-specific PageRank score of word w in topic t, $e(w_j, w_i)$ is the weight for the edge $(w_j \to w_i)$, $O(w_j) = \sum_{w'} e(w_j, w')$ and λ is a damping factor ranging from 0 to 1. The topic-specific preference value $P_t(w)$ for each word w is its random jumping probability with the constraint that $\sum_{w \in \mathcal{V}} P_t(w) = 1$ given topic t. A large $R_t(\cdot)$ indicates a word is a good candidate keyword in topic t. We denote this original version of the Topical PageRank as *TPR*.

However, the original *TPR* ignores the topic context when setting the edge weights; the edge weight is set by counting the number of co-occurrences of the two words within a certain window size. Taking the topic of "electronic products" as an example, the word "juice" may co-occur frequently with a good keyword "apple" for this topic because of Apple electronic products, so "juice" may be ranked high by this context-free co-occurrence edge weight although it is not related to electronic products. In other words, context-free propagation may cause the scores to be off-topic.

So in this paper, we propose to use a topic context sensitive PageRank method. Formally, we have

$$R_t(w_i) = \lambda \sum_{j:w_j \to w_i} \frac{e_t(w_j, w_i)}{O_t(w_j)} R_t(w_j) + (1-\lambda) P_t(w_i). \quad (2)$$

Here we compute the propagation from w_j to w_i in the context of topic t, namely, the edge weight from w_j to w_i is parameterized by t. In this paper, we compute edge weight $e_t(w_j, w_i)$ between two words by counting the number of co-occurrences of these two words in tweets *assigned to topic* t. We denote this context-sensitive topical PageRank as *cTPR*.

After keyword ranking using *cTPR* or any other method, we adopt a common candidate keyphrase generation method proposed by Mihalcea and Tarau (2004) as follows. We first select the top S keywords for each topic, and then look for combinations of these keywords that occur as frequent phrases in the text collection. More details are given in Section 4.

3.4 Probabilistic Models for Topical Keyphrase Ranking

With the candidate keyphrases, our next step is to rank them. While a standard method is to simply aggregate the scores of keywords inside a candidate keyphrase as the score for the keyphrase, here we propose a different probabilistic scoring function. Our method is based on the following hypotheses about good keyphrases given a topic:

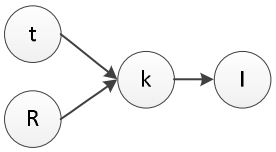


Figure 2: Assumptions of variable dependencies.

Relevance: A good keyphrase should be closely related to the given topic and also discriminative. For example, for the topic "news," "*president obama*" is a good keyphrase while "*math class*" is not.

Interestingness: A good keyphrase should be interesting and can attract users' attention. For example, for the topic "music," "*justin bieber*" is more interesting than "*song player*."

Sometimes, there is a trade-off between these two properties and a good keyphrase has to balance both.

Let R be a binary variable to denote relevance where 1 is relevant and 0 is irrelevant. Let I be another binary variable to denote interestingness where 1 is interesting and 0 is non-interesting. Let k denote a candidate keyphrase. Following the probabilistic relevance models in information retrieval (Lafferty and Zhai, 2003), we propose to use $P(R=1, I=1|t,k)$ to rank candidate keyphrases for topic t. We have

$$\begin{aligned}
& P(R=1, I=1|t,k) \\
= \; & P(R=1|t,k)P(I=1|t,k,R=1) \\
= \; & P(I=1|t,k,R=1)P(R=1|t,k) \\
= \; & P(I=1|k)P(R=1|t,k) \\
= \; & P(I=1|k) \times \frac{P(R=1|t,k)}{P(R=1|t,k)+P(R=0|t,k)} \\
= \; & P(I=1|k) \times \frac{1}{1+\frac{P(R=0|t,k)}{P(R=1|t,k)}} \\
= \; & P(I=1|k) \times \frac{1}{1+\frac{P(R=0,k|t)}{P(R=1,k|t)}} \\
= \; & P(I=1|k) \times \frac{1}{1+\frac{P(R=0|t)}{P(R=1|t)} \times \frac{P(k|t,R=0)}{P(k|t,R=1)}} \\
= \; & P(I=1|k) \times \frac{1}{1+\frac{P(R=0)}{P(R=1)} \times \frac{P(k|t,R=0)}{P(k|t,R=1)}}.
\end{aligned}$$

Here we have assumed that I is independent of t and R given k, i.e. the interestingness of a keyphrase is independent of the topic or whether the keyphrase is relevant to the topic. We have also assumed that R is independent of t when k is unknown, i.e. without knowing the keyphrase, the relevance is independent of the topic. Our assumptions can be depicted by Figure 2.

We further define $\delta = \frac{P(R=0)}{P(R=1)}$. In general we can assume that $P(R=0) \gg P(R=1)$ because there are much more non-relevant keyphrases than relevant ones, that is, $\delta \gg 1$. In this case, we have

$$\begin{aligned}
& \log P(R=1, I=1|t,k) \qquad (3) \\
= \; & \log\Big(P(I=1|k) \times \frac{1}{1+\delta \times \frac{P(k|t,R=0)}{P(k|t,R=1)}}\Big) \\
\approx \; & \log\Big(P(I=1|k) \times \frac{P(k|t,R=1)}{P(k|t,R=0)} \times \frac{1}{\delta}\Big) \\
= \; & \log P(I=1|k) + \log \frac{P(k|t,R=1)}{P(k|t,R=0)} - \log\delta.
\end{aligned}$$

We can see that the ranking score $\log P(R=1, I=1|t,k)$ can be decomposed into two components, a relevance score $\log \frac{P(k|t,R=1)}{P(k|t,R=0)}$ and an interestingness score $\log P(I=1|k)$. The last term $\log\delta$ is a constant and thus not relevant.

Estimating the relevance score

Let a keyphrase candidate k be a sequence of words $(w_1, w_2, \ldots, w_N)$. Based on an independent assumption of words given R and t, we have

$$\begin{aligned}
\log \frac{P(k|t,R=1)}{P(k|t,R=0)} &= \log \frac{P(w_1 w_2 \ldots w_N|t,R=1)}{P(w_1 w_2 \ldots w_N|t,R=0)} \\
&= \sum_{n=1}^{N} \log \frac{P(w_n|t,R=1)}{P(w_n|t,R=0)}. \qquad (4)
\end{aligned}$$

Given the topic model ϕ^t previously learned for topic t, we can set $P(w|t,R=1)$ to ϕ_w^t, i.e. the probability of w under ϕ^t. Following Griffiths and Steyvers (2004), we estimate ϕ_w^t as

$$\phi_w^t = \frac{\#(\mathcal{C}_t, w) + \beta}{\#(\mathcal{C}_t, \cdot) + \beta|\mathcal{V}|}. \qquad (5)$$

Here $\mathcal{C}_t$ denotes the collection of tweets assigned to topic t, $\#(\mathcal{C}_t, w)$ is the number of times w appears in $\mathcal{C}_t$, and $\#(\mathcal{C}_t, \cdot)$ is the total number of words in $\mathcal{C}_t$.

$P(w|t,R=0)$ can be estimated using a smoothed background model.

$$P(w|R=0,t) = \frac{\#(\mathcal{C}, w) + \mu}{\#(\mathcal{C}, \cdot) + \mu|\mathcal{V}|}. \qquad (6)$$

Here $\#(\mathcal{C},\cdot)$ denotes the number of words in the whole collection $\mathcal{C}$, and $\#(\mathcal{C},w)$ denotes the number of times w appears in the whole collection.

After plugging Equation (5) and Equation (6) into Equation (4), we get the following formula for the relevance score:

$$\begin{aligned}
&\log\frac{P(k|t,R=1)}{P(k|t,R=0)}\\
=&\sum_{w\in k}\Big(\log\frac{\#(\mathcal{C}_t,w)+\beta}{\#(\mathcal{C},w)+\mu}+\log\frac{\#(\mathcal{C},\cdot)+\mu|\mathcal{V}|}{\#(\mathcal{C}_t,\cdot)+\beta|\mathcal{V}|}\Big)\\
=&\Big(\sum_{w\in k}\log\frac{\#(\mathcal{C}_t,w)+\beta}{\#(\mathcal{C},w)+\mu}\Big)+|k|\eta, \qquad (7)
\end{aligned}$$

where $\eta=\frac{\#(\mathcal{C},\cdot)+\mu|\mathcal{V}|}{\#(\mathcal{C}_t,\cdot)+\beta|\mathcal{V}|}$ and $|k|$ denotes the number of words in k.

Estimating the interestingness score

To capture the interestingness of keyphrases, we make use of the retweeting behavior in Twitter. We use string matching with `RT` to determine whether a tweet is an original posting or a retweet. If a tweet is interesting, it tends to get retweeted multiple times. Retweeting is therefore a stronger indicator of user interests than tweeting. We use retweet ratio $\frac{|\text{ReTweets}_k|}{|\text{Tweets}_k|}$ to estimate $P(I=1|k)$. To prevent zero frequency, we use a modified add-one smoothing method. Finally, we get

$$\log P(I=1|k)=\log\frac{|\text{ReTweets}_k|+1.0}{|\text{Tweets}_k|+l_{avg}}. \qquad (8)$$

Here $|\text{ReTweets}_k|$ and $|\text{Tweets}_k|$ denote the numbers of retweets and tweets containing the keyphrase k, respectively, and l_{avg} is the average number of tweets that a candidate keyphrase appears in.

Finally, we can plug Equation (7) and Equation (8) into Equation (3) and obtain the following scoring function for ranking:

$$\begin{aligned}
\text{Score}_t(k)=&\log\frac{|\text{ReTweets}_k|+1.0}{|\text{Tweets}_k|+l_{avg}} \qquad (9)\\
&+\left(\sum_{w\in k}\log\frac{\#(\mathcal{C}_t,w)+\beta}{\#(\mathcal{C},w)+\mu}\right)+|k|\eta.
\end{aligned}$$

#user	#tweet	#term	#token
13,307	1,300,300	50,506	11,868,910

Table 1: Some statistics of the data set.

Incorporating length preference

Our preliminary experiments with Equation (9) show that this scoring function usually ranks longer keyphrases higher than shorter ones. However, because our candidate keyphrase are extracted without using any linguistic knowledge such as noun phrase boundaries, longer candidate keyphrases tend to be less meaningful as a phrase. Moreover, for our task of using keyphrases to summarize Twitter, we hypothesize that shorter keyphrases are preferred by users as they are more compact. We would therefore like to incorporate some length preference.

Recall that Equation (9) is derived from $P(R=1,I=1|t,k)$, but this probability does not allow us to directly incorporate any length preference. We further observe that Equation (9) tends to give longer keyphrases higher scores mainly due to the term $|k|\eta$. So here we heuristically incorporate our length preference by removing $|k|\eta$ from Equation (9), resulting in the following final scoring function:

$$\begin{aligned}
\text{Score}_t(k)=&\log\frac{|\text{ReTweets}_k|+1.0}{|\text{Tweets}_k|+l_{avg}} \qquad (10)\\
&+\left(\sum_{w\in k}\log\frac{\#(\mathcal{C}_t,w)+\beta}{\#(\mathcal{C},w)+\mu}\right).
\end{aligned}$$

4 Experiments

4.1 Data Set and Preprocessing

We use a Twitter data set collected from Singapore users for evaluation. We used *Twitter REST API*[1] to facilitate the data collection. The majority of the tweets collected were published in a 20-week period from December 1, 2009 through April 18, 2010. We removed common stopwords and words which appeared in fewer than 10 tweets. We also removed all users who had fewer than 5 tweets. Some statistics of this data set after cleaning are shown in Table 1.

We ran Twitter-LDA with 500 iterations of Gibbs sampling. After trying a few different numbers of

[1] http://apiwiki.twitter.com/w/page/22554663/REST-API-Documentation

topics, we empirically set the number of topics to 30. We set α to $50.0/|\mathcal{T}|$ as Griffiths and Steyvers (2004) suggested, but set β to a smaller value of 0.01 and γ to 20. We chose these parameter settings because they generally gave coherent and meaningful topics for our data set. We selected 10 topics that cover a diverse range of content in Twitter for evaluation of topical keyphrase extraction. The top 10 words of these topics are shown in Table 2.

We also tried the standard LDA model and the author-topic model on our data set and found that our proposed topic model was better or at least comparable in terms of finding meaningful topics. In addition to generating meaningful topics, Twitter-LDA is much more convenient in supporting the computation of tweet-level statistics (e.g. the number of co-occurrences of two words in a specific topic) than the standard LDA or the author-topic model because Twitter-LDA assumes a single topic assignment for an entire tweet.

4.2 Methods for Comparison

As we have described in Section 3.1, there are three steps to generate keyphrases, namely, keyword ranking, candidate keyphrase generation, and keyphrase ranking. We have proposed a context-sensitive topical PageRank method (*cTPR*) for the first step of keyword ranking, and a probabilistic scoring function for the third step of keyphrase ranking. We now describe the baseline methods we use to compare with our proposed methods.

Keyword Ranking

We compare our *cTPR* method with the original topical PageRank method (Equation (1)), which represents the state of the art. We refer to this baseline as *TPR*.

For both *TPR* and *cTPR*, the damping factor is empirically set to 0.1, which always gives the best performance based on our preliminary experiments. We use normalized $P(t|w)$ to set $P_t(w)$ because our preliminary experiments showed that this was the best among the three choices discussed by Liu et al. (2010). This finding is also consistent with what Liu et al. (2010) found.

In addition, we also use two other baselines for comparison: (1) *kwBL1*: ranking by $P(w|t) = \phi_w^t$. (2) *kwBL2*: ranking by $P(t|w) = \frac{P(t)\phi_w^t}{\sum_{t'} P(t')\phi_w^{t'}}$.

Keyphrase Ranking

We use *kpRelInt* to denote our relevance and interestingness based keyphrase ranking function $P(R = 1, I = 1|t, k)$, i.e. Equation (10). β and μ are empirically set to 0.01 and 500. Usually μ can be set to zero, but in our experiments we find that our ranking method needs a more uniform estimation of the background model. We use the following ranking functions for comparison:

- *kpBL1*: Similar to what is used by Liu et al. (2010), we can rank candidate keyphrases by $\sum_{w \in k} f(w)$, where $f(w)$ is the score assigned to word w by a keyword ranking method.
- *kpBL2*: We consider another baseline ranking method by $\sum_{w \in k} \log f(w)$.
- *kpRel*: If we consider only relevance but not interestingness, we can rank candidate keyphrases by $\sum_{w \in k} \log \frac{\#(\mathcal{C}_t, w)+\beta}{\#(\mathcal{C}, w)+\mu}$.

4.3 Gold Standard Generation

Since there is no existing test collection for topical keyphrase extraction from Twitter, we manually constructed our test collection. For each of the 10 selected topics, we ran all the methods to rank keywords. For each method we selected the top 3000 keywords and searched all the combinations of these words as phrases which have a frequency larger than 30. In order to achieve high phraseness, we first computed the minimum value of pointwise mutual information for all bigrams in one combination, and we removed combinations having a value below a threshold, which was empirically set to 2.135. Then we merged all these candidate phrases. We did not consider single-word phrases because we found that it would include too many frequent words that might not be useful for summaries.

We asked two judges to judge the quality of the candidate keyphrases. The judges live in Singapore and had used Twitter before. For each topic, the judges were given the top topic words and a short topic description. Web search was also available. For each candidate keyphrase, we asked the judges to score it as follows: 2 (relevant, meaningful and informative), 1 (relevant but either too general or too specific, or informal) and 0 (irrelevant or meaningless). Here in addition to relevance, the other two criteria, namely, whether a phrase is meaningful and informative, were studied by Tomokiyo and Hurst

T_2	T_4	T_5	T_{10}	T_{12}	T_{13}	T_{18}	T_{20}	T_{23}	T_{25}
eat	twitter	love	singapore	singapore	hot	iphone	song	study	win
food	tweet	idol	road	#singapore	rain	google	video	school	game
dinner	blog	adam	mrt	#business	weather	social	youtube	time	team
lunch	facebook	watch	sgreinfo	#news	cold	media	love	homework	match
eating	internet	april	east	health	morning	ipad	songs	tomorrow	play
ice	tweets	hot	park	asia	sun	twitter	bieber	maths	chelsea
chicken	follow	lambert	room	market	good	free	music	class	world
cream	msn	awesome	sqft	world	night	app	justin	paper	united
tea	followers	girl	price	prices	raining	apple	feature	math	liverpool
hungry	time	american	built	bank	air	marketing	twitter	finish	arsenal

Table 2: Top 10 Words of Sample Topics on our Singapore Twitter Dateset.

(2003). We then averaged the scores of the two judges as the final scores. The Cohen's Kappa coefficients of the 10 topics range from 0.45 to 0.80, showing fair to good agreement[2]. We further discarded all candidates with an average score less than 1. The number of the remaining keyphrases for each topic ranges from 56 to 282.

4.4 Evaluation Metrics

Traditionally keyphrase extraction is evaluated using precision and recall on all the extracted keyphrases. We choose not to use these measures for the following reasons: (1) Traditional keyphrase extraction works on single documents while we study topical keyphrase extraction. The gold standard keyphrase list for a single document is usually short and clean, while for each Twitter topic there can be many keyphrases, some are more relevant and interesting than others. (2) Our extracted topical keyphrases are meant for summarizing Twitter content, and they are likely to be directly shown to the users. It is therefore more meaningful to focus on the quality of the top-ranked keyphrases.

Inspired by the popular *nDCG* metric in information retrieval (Järvelin and Kekäläinen, 2002), we define the following normalized keyphrase quality measure (*nKQM*) for a method $\mathcal{M}$:

$$nKQM@K = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\frac{\sum_{j=1}^{K}\frac{1}{\log_2(j+1)}score(\mathcal{M}_{t,j})}{IdealScore_{(K,t)}},$$

where $\mathcal{T}$ is the set of topics, $\mathcal{M}_{t,j}$ is the j-th keyphrase generated by method $\mathcal{M}$ for topic t, $score(\cdot)$ is the average score from the two human judges, and $IdealScore_{(K,t)}$ is the normalization factor—score of the top K keyphrases of topic t under the ideal ranking. Intuitively, if $\mathcal{M}$ returns more good keyphrases in top ranks, its *nKQM* value will be higher.

We also use mean average precision (MAP) to measure the overall performance of keyphrase ranking:

$$MAP = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\frac{1}{N_{\mathcal{M},t}}\sum_{j=1}^{|\mathcal{M}_t|}\frac{N_{\mathcal{M},t,j}}{j}\mathbb{1}(score(\mathcal{M}_{t,j})\geq 1),$$

where $\mathbb{1}(\mathcal{S})$ is an indicator function which returns 1 when $\mathcal{S}$ is true and 0 otherwise, $N_{\mathcal{M},t,j}$ denotes the number of correct keyphrases among the top j keyphrases returned by $\mathcal{M}$ for topic t, and $N_{\mathcal{M},t}$ denotes the total number of correct keyphrases of topic t returned by $\mathcal{M}$.

4.5 Experiment Results

Evaluation of keyword ranking methods

Since keyword ranking is the first step for keyphrase extraction, we first compare our keyword ranking method *cTPR* with other methods. For each topic, we pooled the top 20 keywords ranked by all four methods. We manually examined whether a word is a good keyword or a noisy word based on topic context. Then we computed the average number of noisy words in the 10 topics for each method. As shown in Table 5, we can observe that *cTPR* performed the best among the four methods.

Since our final goal is to extract topical keyphrases, we further compare the performance of *cTPR* and *TPR* when they are combined with a keyphrase ranking algorithm. Here we use the two

[2]We find that judgments on topics related to social media (e.g. T_4) and daily life (e.g. T_{13}) tend to have a higher degree of disagreement.

	Method	nKQM@5	nKQM@10	nKQM@25	nKQM@50	MAP
kpBL1	*TPR*	0.5015	0.54331	0.5611	0.5715	0.5984
	kwBL1	0.6026	0.5683	0.5579	0.5254	0.5984
	kwBL2	0.5418	0.5652	0.6038	0.5896	0.6279
	cTPR	**0.6109**	**0.6218**	**0.6139**	**0.6062**	**0.6608**
kpBL2	*TPR*	0.7294	0.7172	0.6921	0.6433	0.6379
	kwBL1	0.7111	0.6614	0.6306	0.5829	0.5416
	kwBL2	0.5418	0.5652	0.6038	0.5896	0.6545
	cTPR	**0.7491**	**0.7429**	**0.6930**	**0.6519**	**0.6688**

Table 3: Comparisons of keyphrase extraction for *cTPR* and baselines.

Method	nKQM@5	nKQM@10	nKQM@25	nKQM@50	MAP
cTPR+kpBL1	0.61095	0.62182	0.61389	0.60618	0.6608
cTPR+kpBL2	0.74913	0.74294	0.69303	0.65194	0.6688
cTPR+kpRel	0.75361	0.74926	0.69645	0.65065	**0.6696**
cTPR+kpRelInt	**0.81061**	**0.75184**	**0.71422**	**0.66319**	0.6694

Table 4: Comparisons of keyphrase extraction for different keyphrase ranking methods.

kwBL1	*kwBL2*	*TPR*	*cTPR*
2	3	4.9	1.5

Table 5: Average number of noisy words among the top 20 keywords of the 10 topics.

baseline keyphrase ranking algorithms *kpBL1* and *kpBL2*. The comparison is shown in Table 3. We can see that *cTPR* is consistently better than the three other methods for both *kpBL1* and *kpBL2*.

Evaluation of keyphrase ranking methods

In this section we compare keypharse ranking methods. Previously we have shown that *cTPR* is better than *TPR*, *kwBL1* and *kwBL2* for keyword ranking. Therefore we use *cTPR* as the keyword ranking method and examine the keyphrase ranking method *kpRelInt* with *kpBL1*, *kpBL2* and *kpRel* when they are combined with *cTPR*. The results are shown in Table 4. From the results we can see the following: (1) Keyphrase ranking methods *kpRelInt* and *kpRel* are more effective than *kpBL1* and *kpBL2*, especially when using the *nKQM* metric. (2) *kpRelInt* is better than *kpRel*, especially for the *nKQM* metric. Interestingly, we also see that for the *nKQM* metric, *kpBL1*, which is the most commonly used keyphrase ranking method, did not perform as well as *kpBL2*, a modified version of *kpBL1*.

We also tested *kpRelInt* and *kpRel* on *TPR*, *kwBL1* and *kwBL2* and found that *kpRelInt* and *kpRel* are consistently better than *kpBL2* and *kpBL1*. Due to space limit, we do not report all the results here. These findings support our assumption that our proposed keyphrase ranking method is effective.

The comparison between *kpBL2* with *kpBL1* shows that taking the product of keyword scores is more effective than taking their sum. *kpRel* and *kpRelInt* also use the product of keyword scores. This may be because there is more noise in Twitter than traditional documents. Common words (e.g. "good") and domain background words (e.g. "Singapore") tend to gain higher weights during keyword ranking due to their high frequency, especially in graph-based method, but we do not want such words to contribute too much to keyphrase scores. Taking the product of keyword scores is therefore more suitable here than taking their sum.

Further analysis of interestingness

As shown in Table 4, *kpRelInt* performs better in terms of *nKQM* compared with *kpRel*. Here we study why it worked better for keyphrase ranking. The only difference between *kpRel* and *kpRelInt* is that *kpRelInt* includes the factor of user interests. By manually examining the top keyphrases, we find that the topics "Movie-TV" (T_5), "News" (T_{12}), "Music" (T_{20}) and "Sports" (T_{25}) particularly benefited from *kpRelInt* compared with other topics. We find that well-known named entities (e.g. celebrities, political leaders, football clubs and big companies) and significant events tend to be ranked higher by *kpRelInt* than *kpRel*.

We then counted the numbers of *entity and event keyphrases* for these four topics retrieved by different methods, shown in Table 6 . We can see that in these four topics, *kpRelInt* is consistently better than *kpRel* in terms of the number of *entity and event keyphrases* retrieved.

T_2	T_5	T_{10}	T_{12}	T_{20}	T_{25}
chicken rice	adam lambert	north east	president obama	justin bieber	manchester united
ice cream	jack neo	rent blk	magnitude earthquake	music video	champions league
fried chicken	american idol	east coast	volcanic ash	lady gaga	football match
curry rice	david archuleta	east plaza	prime minister	taylor swift	premier league
chicken porridge	robert pattinson	west coast	iceland volcano	demi lovato	f1 grand prix
curry chicken	alexander mcqueen	bukit timah	chile earthquake	youtube channel	tiger woods
beef noodles	april fools	street view	goldman sachs	miley cyrus	grand slam(tennis)
chocolate cake	harry potter	orchard road	coe prices	telephone video	liverpool fans
cheese fries	april fool	toa payoh	haiti earthquake	song lyrics	final score
instant noodles	andrew garcia	marina bay	#singapore #business	joe jonas	manchester derby

Table 7: Top 10 keyphrases of 6 topics from *cTPR+kpRelInt*.

Methods	T_5	T_{12}	T_{20}	T_{25}
cTPR+kpRel	8	9	16	11
cTPR+kpRelInt	10	12	17	14

Table 6: Numbers of entity and event keyphrases retrieved by different methods within top 20.

On the other hand, we also find that for some topics interestingness helped little or even hurt the performance a little, e.g. for the topics "Food" and "Traffic." We find that the keyphrases in these topics are stable and change less over time. This may suggest that we can modify our formula to handle different topics different. We will explore this direction in our future work.

Parameter settings

We also examine how the parameters in our model affect the performance.

λ: We performed a search from 0.1 to 0.9 with a step size of 0.1. We found $\lambda = 0.1$ was the optimal parameter for *cTPR* and *TPR*. However, *TPR* is more sensitive to λ. The performance went down quickly with λ increasing.

μ: We checked the overall performance with $\mu \in \{400, 450, 500, 550, 600\}$. We found that $\mu = 500 \approx 0.01|\mathcal{V}|$ gave the best performance generally for *cTPR*. The performance difference is not very significant between these different values of μ, which indicates that the our method is robust.

4.6 Qualitative evaluation of *cTPR+kpRelInt*

We show the top 10 keyphrases discovered by *cTPR+kRelInt* in Table 7. We can observe that these keyphrases are clear, interesting and informative for summarizing Twitter topics.

We hypothesize that the following applications can benefit from the extracted keyphrases:

Automatic generation of realtime trendy phrases: For exampoe, keyphrases in the topic "Food" (T_2) can be used to help online restaurant reviews.

Event detection and topic tracking: In the topic "News" top keyphrases can be used as candidate trendy topics for event detection and topic tracking.

Automatic discovery of important named entities: As discussed previously, our methods tend to rank important named entities such as celebrities in high ranks.

5 Conclusion

In this paper, we studied the novel problem of topical keyphrase extraction for summarizing and analyzing Twitter content. We proposed the context-sensitive topical PageRank (cTPR) method for keyword ranking. Experiments showed that *cTPR* is consistently better than the original *TPR* and other baseline methods in terms of top keyword and keyphrase extraction. For keyphrase ranking, we proposed a probabilistic ranking method, which models both relevance and interestingness of keyphrases. In our experiments, this method is shown to be very effective to boost the performance of keyphrase extraction for different kinds of keyword ranking methods. In the future, we may consider how to incorporate keyword scores into our keyphrase ranking method. Note that we propose to rank keyphrases by a general formula $P(R = 1, I = 1|t, k)$ and we have made some approximations based on reasonable assumptions. There should be other potential ways to estimate $P(R = 1, I = 1|t, k)$.

Acknowledgements

This work was done during Xin Zhao's visit to the Singapore Management University. Xin Zhao and Xiaoming Li are partially supported by NSFC under

the grant No. 60933004, 61073082, 61050009 and HGJ Grant No. 2011ZX01042-001-001.

References

Ken Barker and Nadia Cornacchia. 2000. Using noun phrase heads to extract document keyphrases. In *Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*, pages 40–52.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl. 1):5228–5235.

Liangjie Hong and Brian D. Davison. 2010. Empirical study of topic modeling in Twitter. In *Proceedings of the First Workshop on Social Media Analytics*.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446.

John Lafferty and Chengxiang Zhai. 2003. Probabilistic relevance models based on document and query generation. *Language Modeling and Information Retrieval*, 13.

Quanzhi Li, Yi-Fang Wu, Razvan Bot, and Xin Chen. 2004. Incorporating document keyphrases in search results. In *Proceedings of the 10th Americas Conference on Information Systems*.

Marina Litvak and Mark Last. 2008. Graph-based keyword extraction for single-document summarization. In *Proceedings of the Workshop on Multi-source Multilingual Information Extraction and Summarization*, pages 17–24.

Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 366–376.

Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. 2007. Automatic labeling of multinomial topic models. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 490–499.

R. Mihalcea and P. Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Daniel Ramage, Susan Dumais, and Dan Liebling. 2010. Characterizing micorblogs with topic models. In *Proceedings of the 4th International Conference on Weblogs and Social Media*.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes Twitter users: real-time event detection by social sensors. In *Proceedings of the 19th International World Wide Web Conference*.

Takashi Tomokiyo and Matthew Hurst. 2003. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, pages 33–40.

Andranik Tumasjan, Timm O. Sprenger, Philipp G. Sandner, and Isabell M. Welpe. 2010. Predicting elections with Twitter: What 140 characters reveal about political sentiment. In *Proceedings of the 4th International Conference on Weblogs and Social Media*.

Peter Turney. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval*, (4):303–336.

Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. 2010. TwitterRank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM International Conference on Web Search and Data Mining*.

Wei Wu, Bin Zhang, and Mari Ostendorf. 2010. Automatic generation of personalized annotation tags for twitter users. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 689–692.

Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Lim Ee-Peng, Hongfei Yan, and Xiaoming Li. 2011. Comparing Twitter and traditional media using topic models. In *Proceedings of the 33rd European Conference on Information Retrieval*.

Event Discovery in Social Media Feeds

Edward Benson, Aria Haghighi, and Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{eob, aria42, regina}@csail.mit.edu

Abstract

We present a novel method for record extraction from social streams such as Twitter. Unlike typical extraction setups, these environments are characterized by short, one sentence messages with heavily colloquial speech. To further complicate matters, individual messages may not express the full relation to be uncovered, as is often assumed in extraction tasks. We develop a graphical model that addresses these problems by learning a latent set of records and a record-message alignment simultaneously; the output of our model is a set of canonical records, the values of which are consistent with aligned messages. We demonstrate that our approach is able to accurately induce event records from Twitter messages, evaluated against events from a local city guide. Our method achieves significant error reduction over baseline methods.[1]

1 Introduction

We propose a method for discovering event records from social media feeds such as Twitter. The task of extracting event properties has been well studied in the context of formal media (e.g., newswire), but data sources such as Twitter pose new challenges. Social media messages are often short, make heavy use of colloquial language, and require situational context for interpretation (see examples in Figure 1). Not all properties of an event may be expressed in a single message, and the mapping between messages and canonical event records is not obvious.

Twitter Messages

Seated at @carnegiehall waiting for @CraigyFerg's show to begin
RT @leerader : getting REALLY stoked for #CraigyAtCarnegie sat night. Craig, , want to join us for dinner at the pub across the street? 5pm, be there!
@DJPaulyD absolutely killed it at Terminal 5 last night.
@DJPaulyD : DJ Pauly D Terminal 5 NYC Insanity ! #ohyeah @keadour @kellaferr24
Craig, nice seeing you at #noelnight this weekend @becksdavis!

Records

Artist	Venue
Craig Ferguson	Carnegie Hall
DJ Pauly D	Terminal 5

Figure 1: Examples of Twitter messages, along with automatically extracted records.

These properties of social media streams make existing extraction techniques significantly less effective. Despite these challenges, this data exhibits an important property that makes learning amenable: the multitude of messages referencing the same event.

Our goal is to induce a comprehensive set of event records given a seed set of example records, such as a city event calendar table. While such resources are widely available online, they are typically high precision, but low recall. Social media is a natural place to discover new events missed by curation, but mentioned online by someone planning to attend.

We formulate our approach as a structured graphical model which simultaneously analyzes individual messages, clusters them according to event, and induces a canonical value for each event property. At the message level, the model relies on a conditional random field component to extract field values such

[1]Data and code available at `http://groups.csail.mit.edu/rbg/code/twitter`

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 389–398,
Portland, Oregon, June 19-24, 2011.

as location of the event and artist name. We bias local decisions made by the CRF to be consistent with canonical record values, thereby facilitating consistency within an event cluster. We employ a factor-graph model to capture the interaction between each of these decisions. Variational inference techniques allow us to effectively and efficiently make predictions on a large body of messages.

A seed set of example records constitutes our only source of supervision; we do not observe alignment between these seed records and individual messages, nor any message-level field annotation. The output of our model consists of an event-based clustering of messages, where each cluster is represented by a single multi-field record with a canonical value chosen for each field.

We apply our technique to construct entertainment event records for the city calendar section of `NYC.com` using a stream of Twitter messages. Our method yields up to a 63% recall against the city table and up to 85% precision evaluated manually, significantly outperforming several baselines.

2 Related Work

A large number of information extraction approaches exploit redundancy in text collections to improve their accuracy and reduce the need for manually annotated data (Agichtein and Gravano, 2000; Yangarber et al., 2000; Zhu et al., 2009; Mintz et al., 2009a; Yao et al., 2010b; Hasegawa et al., 2004; Shinyama and Sekine, 2006). Our work most closely relates to methods for multi-document information extraction which utilize redundancy in input data to increase the accuracy of the extraction process. For instance, Mann and Yarowsky (2005) explore methods for fusing extracted information across multiple documents by performing extraction on each document independently and then merging extracted relations by majority vote. This idea of consensus-based extraction is also central to our method. However, we incorporate this idea into our model by simultaneously clustering output and labeling documents rather than performing the two tasks in serial fashion. Another important difference is inherent in the input data we are processing: it is not clear a priori which extraction decisions should agree with each other. Identifying messages that refer to the same event is a large part of our challenge.

Our work also relates to recent approaches for relation extraction with *distant supervision* (Mintz et al., 2009b; Bunescu and Mooney, 2007; Yao et al., 2010a). These approaches assume a database and a collection of documents that verbalize some of the database relations. In contrast to traditional supervised IE approaches, these methods do not assume that relation instantiations are annotated in the input documents. For instance, the method of Mintz et al. (2009b) induces the mapping automatically by bootstrapping from sentences that directly match record entries. These mappings are used to learn a classifier for relation extraction. Yao et al. (2010a) further refine this approach by constraining predicted relations to be consistent with entity types assignment. To capture the complex dependencies among assignments, Yao et al. (2010a) use a factor graph representation. Despite the apparent similarity in model structure, the two approaches deal with various types of uncertainties. The key challenge for our method is modeling message to record alignment which is not an issue in the previous set up.

Finally, our work fits into a broader area of text processing methods designed for social-media streams. Examples of such approaches include methods for conversation structure analysis (Ritter et al., 2010) and exploration of geographic language variation (Eisenstein et al., 2010) from Twitter messages. To our knowledge no work has yet addressed record extraction from this growing corpus.

3 Problem Formulation

Here we describe the key latent and observed random variables of our problem. A depiction of all random variables is given in Figure 2.

Message (x): Each message x is a single posting to Twitter. We use x^j to represent the j^{th} token of x, and we use $\mathbf{x}$ to denote the entire collection of messages. Messages are always observed during training and testing.

Record (R): A record is a representation of the canonical properties of an event. We use R_i to denote the i^{th} record and R_i^ℓ to denote the value of the ℓ^{th} property of that record. In our experiments, each record R_i is a tuple $\langle R_i^1, R_i^2 \rangle$ which represents that

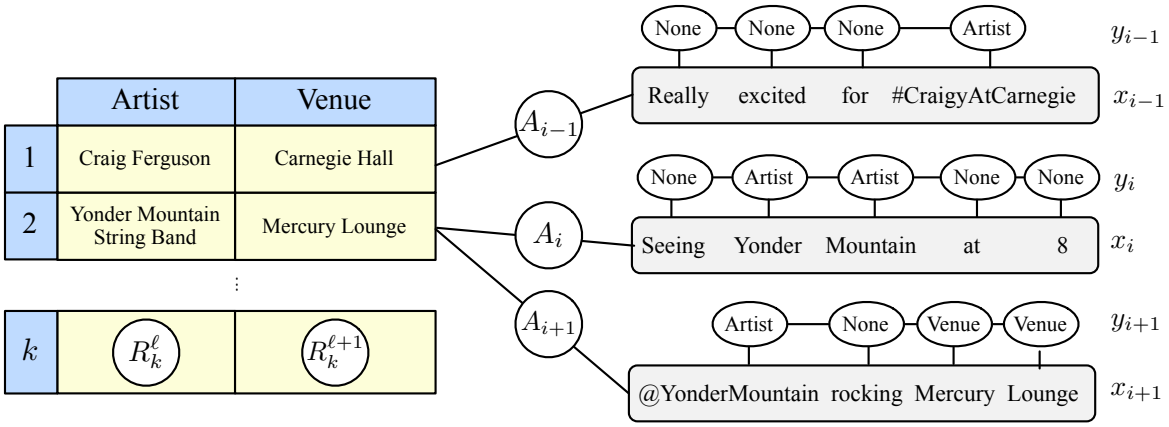


Figure 2: The key variables of our model. A collection of K latent records R_k, each consisting of a set of L properties. In the figure above, R_1^1 ="Craig Ferguson" and R_1^2 ="Carnegie Hall." Each tweet x_i is associated with a labeling over tokens y_i and is aligned to a record via the A_i variable. See Section 3 for further details.

record's values for the schema ⟨ARTIST, VENUE⟩. Throughout, we assume a known fixed number K of records $R_1, \ldots, R_K$, and we use $\mathbf{R}$ to denote this collection of records. For tractability, we consider a finite number of possibilities for each R_k^ℓ which are computed from the input $\mathbf{x}$ (see Section 5.1 for details). Records are observed during training and latent during testing.

Message Labels (y): We assume that each message has a sequence labeling, where the labels consist of the record fields (e.g., ARTIST and VENUE) as well as a NONE label denoting the token does not correspond to any domain field. Each token x^j in a message has an associated label y^j. Message labels are always latent during training and testing.

Message to Record Alignment (A): We assume that each message is aligned to some record such that the event described in the message is the one represented by that record. Each message x_i is associated with an alignment variable A_i that takes a value in $\{1, \ldots, K\}$. We use $\mathbf{A}$ to denote the set of alignments across all x_i. Multiple messages can and do align to the same record. As discussed in Section 4, our model will encourage tokens associated with message labels to be "similar" to corresponding aligned record values. Alignments are always latent during training and testing.

4 Model

Our model can be represented as a factor graph which takes the form,

$$P(R, A, y|x) \propto$$
$$\left(\prod_i \phi_{SEQ}(x_i, y_i)\right) \quad \text{(Seq. Labeling)}$$
$$\left(\prod_\ell \phi_{UNQ}(R^\ell)\right) \quad \text{(Rec. Uniqueness)}$$
$$\left(\prod_{i,\ell} \phi_{POP}(x_i, y_i, R^\ell_{A_i})\right) \quad \text{(Term Popularity)}$$
$$\left(\prod_i \phi_{CON}(x_i, y_i, R_{A_i})\right) \quad \text{(Rec. Consistency)}$$

where $\mathbf{R}^\ell$ denotes the sequence $R_1^\ell, \ldots, R_K^\ell$ of record values for a particular domain field ℓ. Each of the potentials takes a standard log-linear form:

$$\phi(z) = \theta^T f(z)$$

where θ are potential-specific parameters and $f(\cdot)$ is a potential-specific feature function. We describe each potential separately below.

4.1 Sequence Labeling Factor

The sequence labeling factor is similar to a standard sequence CRF (Lafferty et al., 2001), where the potential over a message label sequence decomposes

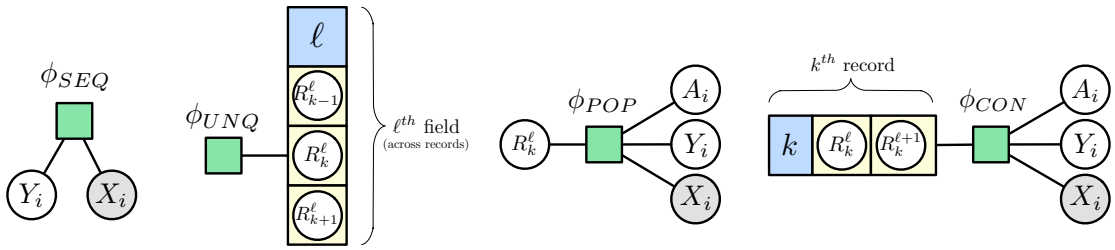


Figure 3: Factor graph representation of our model. Circles represent variables and squares represent factors. For readability, we depict the graph broken out as a set of templates; the full graph is the combination of these factor templates applied to each variable. See Section 4 for further details.

over pairwise cliques:

$$\phi_{SEQ}(x,y) = \exp\{\theta^T_{SEQ} f_{SEQ}(x,y)\}$$
$$= \exp\left\{\theta^T_{SEQ} \sum_j f_{SEQ}(x, y^j, y^{j+1})\right\}$$

This factor is meant to encode the typical message contexts in which fields are evoked (e.g. *going to see X tonight*). Many of the features characterize how likely a given token label, such as ARTIST, is for a given position in the message sequence conditioning arbitrarily on message text context.

The feature function $f_{SEQ}(x,y)$ for this component encodes each token's identity; word shape[2]; whether that token matches a set of regular expressions encoding common emoticons, time references, and venue types; and whether the token matches a bag of words observed in artist names (scraped from Wikipedia; 21,475 distinct tokens from 22,833 distinct names) or a bag of words observed in New York City venue names (scraped from `NYC.com`; 304 distinct tokens from 169 distinct names).[3] The only edge feature is label-to-label.

4.2 Record Uniqueness Factor

One challenge with Twitter is the so-called echo chamber effect: when a topic becomes popular, or "trends," it quickly dominates the conversation online. As a result some events may have only a few referent messages while other more popular events may have thousands or more. In such a circumstance, the messages for a popular event may collect to form multiple identical record clusters. Since we fix the number of records learned, such behavior inhibits the discovery of less talked-about events. Instead, we would rather have just two records: one with two aligned messages and another with thousands. To encourage this outcome, we introduce a potential that rewards fields for being unique across records.

The uniqueness potential $\phi_{UNQ}(\mathbf{R}^\ell)$ encodes the preference that each of the values $R^\ell, \ldots, R^\ell_K$ for each field ℓ do not overlap textually. This factor factorizes over pairs of records:

$$\phi_{UNQ}(\mathbf{R}^\ell) = \prod_{k \neq k'} \phi_{UNQ}(R^\ell_k, R^\ell_{k'})$$

where R^ℓ_k and $R^\ell_{k'}$ are the values of field ℓ for two records R_k and $R_{k'}$. The potential over this pair of values is given by:

$$\phi_{UNQ}(R^\ell_k, R^\ell_{k'}) = \exp\{-\theta^T_{SIM} f_{SIM}(R^\ell_k, R^\ell_{k'})\}$$

where f_{SIM} is computes the likeness of the two values at the token level:

$$f_{SIM}(R^\ell_k, R^\ell_{k'}) = \frac{|R^\ell_k \cap R^\ell_{k'}|}{\max(|R^\ell_k|, |R^\ell_{k'}|)}$$

This uniqueness potential does not encode any preference for record values; it simply encourages each field ℓ to be distinct across records.

4.3 Term Popularity Factor

The term popularity factor ϕ_{POP} is the first of two factors that guide the clustering of messages. Because speech on Twitter is colloquial, we would like these clusters to be amenable to many variations of the canonical record properties that are ultimately learned. The ϕ_{POP} factor accomplishes this by representing a lenient compatibility score between a

[2] e.g.: xxx, XXX, Xxx, or other

[3] These are just features, not a filter; we are free to extract any artist or venue regardless of their inclusion in this list.

message x, its labels y, and some candidate value v for a record field (e.g., *Dave Matthews Band*).

This factor decomposes over tokens, and we align each token x^j with the best matching token v^k in v (e.g., *Dave*). The token level sum is scaled by the length of the record value being matched to avoid a preference for long field values.

$$\phi_{POP}(x, y, R^\ell_A = v) = \sum_j \max_k \frac{\phi_{POP}(x^j, y^j, R^\ell_A = v^k)}{|v|}$$

This token-level component may be thought of as a compatibility score between the labeled token x^j and the record field assignment $R^\ell_A = v$. Given that token x^j aligns with the token v^k, the token-level component returns the sum of three parts, subject to the constraint that $y^j = \ell$:

- $IDF(x^j)\mathbb{I}[x^j = v^k]$, an equality indicator between tokens x^j and v^k, scaled by the inverse document frequency of x^j
- $\alpha IDF(x^j)\left(\mathbb{I}[x^{j-1} = v^{k-1}] + \mathbb{I}[x^{j+1} = v^{k+1}]\right)$, a small bonus of $\alpha = 0.3$ for matches on adjacent tokens, scaled by the IDF of x^j
- $\mathbb{I}[x^j = v^k$ and x contains $v]/|v|$, a bonus for a complete string match, scaled by the size of the value. This is equivalent to this token's contribution to a complete-match bonus.

4.4 Record Consistency Factor

While the uniqueness factor discourages a flood of messages for a single event from clustering into multiple event records, we also wish to discourage messages from multiple events from clustering into the same record. When such a situation occurs, the model may either resolve it by changing inconsistent token labelings to the NONE label or by reassigning some of the messages to a new cluster. We encourage the latter solution with a record consistency factor ϕ_{CON}.

The record consistency factor is an indicator function on the field values of a record being present and labeled correctly in a message. While the popularity factor encourages agreement on a per-label basis, this factor influences the joint behavior of message labels to agree with the aligned record. For a given record, message, and labeling, $\phi_{CON}(x, y, R_A) = 1$ if $\phi_{POP}(x, y, R^\ell_A) > 0$ for all ℓ, and 0 otherwise.

4.5 Parameter Learning

The weights of the CRF component of our model, θ_{SEQ}, are the only weights learned at training time, using a distant supervision process described in Section 6. The weights of the remaining three factors were hand-tuned[4] using our training data set.

5 Inference

Our goal is to predict a set of records $\mathbf{R}$. Ideally we would like to compute $P(\mathbf{R}|\mathbf{x})$, marginalizing out the nuisance variables $\mathbf{A}$ and $\mathbf{y}$. We approximate this posterior using variational inference.[5] Concretely, we approximate the full posterior over latent variables using a mean-field factorization:

$$P(\mathbf{R}, \mathbf{A}, \mathbf{y}|\mathbf{x}) \approx Q(\mathbf{R}, \mathbf{A}, \mathbf{y}) = \left(\prod_{k=1}^{K}\prod_{\ell} q(R^\ell_k)\right)\left(\prod_{i=1}^{n} q(A_i)q(y_i)\right)$$

where each variational factor $q(\cdot)$ represents an approximation of that variable's posterior given observed random variables. The variational distribution $Q(\cdot)$ makes the (incorrect) assumption that the posteriors amongst factors are independent. The goal of variational inference is to set factors $q(\cdot)$ to optimize the variational objective:

$$\min_{Q(\cdot)} KL(Q(\mathbf{R}, \mathbf{A}, \mathbf{y}) \| P(\mathbf{R}, \mathbf{A}, \mathbf{y}|\mathbf{x}))$$

We optimize this objective using coordinate descent on the $q(\cdot)$ factors. For instance, for the case of $q(y_i)$ the update takes the form:

$$q(y_i) \leftarrow \mathbb{E}_{Q/q(y_i)} \log P(\mathbf{R}, \mathbf{A}, \mathbf{y}|\mathbf{x})$$

where $Q/q(y_i)$ denotes the expectation under all variables except y_i. When computing a mean field update, we only need to consider the potentials involving that variable. The complete updates for each of the kinds of variables (y, A, and R^ℓ) can be found in Figure 4. We briefly describe the computations involved with each update.

$q(y)$ **update**: The $q(y)$ update for a single message yields an implicit expression in terms of pairwise cliques in y. We can compute arbitrary

[4] Their values are: $\theta_{UNQ} = -10$, $\theta^{\text{Phrase}}_{POP} = 5$, $\theta^{\text{Token}}_{POP} = 10$, $\theta_{CON} = 2e8$

[5] See Liang and Klein (2007) for an overview of variational techniques.

Message labeling update:

$$\begin{aligned}\ln q(y) &\propto \left\{\mathbb{E}_{Q/q(y)} \ln \phi_{SEQ}(x,y) + \ln\left[\phi_{POP}(x,y,R^\ell_A)\phi_{CON}(x,y,R_A)\right]\right\} \\ &= \ln \phi_{SEQ}(x,y) + \mathbb{E}_{Q/q(y)} \ln\left[\phi_{POP}(x,y,R^\ell_A)\phi_{CON}(x,y,R_A)\right] \\ &= \ln \phi_{SEQ}(x,y) + \sum_{z,v,\ell} q(A=z)q(y^j=\ell)q(R^\ell_z=v)\ln\left[\phi_{POP}(x,y,R^\ell_z=v)\phi_{CON}(x,y,R^\ell_z=v)\right]\end{aligned}$$

Mention record alignment update:

$$\begin{aligned}\ln q(A=z) &\propto \mathbb{E}_{Q/q(A)}\left\{\ln \phi_{SEQ}(x,y) + \ln\left[\phi_{POP}(x,y,R^\ell_A)\phi_{CON}(x,y,R_A)\right]\right\} \\ &\propto \mathbb{E}_{Q/q(A)}\left\{\ln\left[\phi_{POP}(x,y,R^\ell_A)\phi_{CON}(x,y,R_A)\right]\right\} \\ &= \sum_{z,v,\ell} q(R^\ell_z=v)\left\{\ln\left[\phi_{POP}(x,y,R^\ell_z=v)\phi_{CON}(x,y,R^\ell_z=v)\right]\right\} \\ &= \sum_{z,v,\ell} q(R^\ell_z=v)q(y^j_i=\ell)\ln\left[\phi_{POP}(x,y,R^\ell_z=v)\phi_{CON}(x,y,R^\ell_z=v)\right]\end{aligned}$$

Record Field update:

$$\begin{aligned}\ln q(R^\ell_k=v) &\propto \mathbb{E}_{Q/q(R^\ell_k)}\left\{\sum_{k'}\ln\phi_{UNQ}(R^\ell_{k'},v) + \sum_i \ln\left[\phi_{POP}(x_i,y_i,v)\phi_{CON}(x_i,y_i,v)\right]\right\} \\ &= \sum_{k'\neq k,v'}\Bigg(q(R^\ell_{k'}=v')\ln\phi_{UNQ}(v,v') \\ &+ \sum_i q(A_i=k)\sum_j q(y^j_i=\ell)\ln\left[\phi_{POP}(x,y,R^\ell_z=v,j)\phi_{CON}(x,y,R^\ell_z=v,j)\right]\Bigg)\end{aligned}$$

Figure 4: The variational mean-field updates used during inference (see Section 5). Inference consists of performing updates for each of the three kinds of latent variables: message labels (y), record alignments (A), and record field values (R^ℓ). All are relatively cheap to compute except for the record field update $q(R^\ell_k)$ which requires looping potentially over all messages. Note that at inference time all parameters are fixed and so we only need to perform updates for latent variable factors.

marginals for this distribution by using the forwards-backwards algorithm on the potentials defined in the update. Therefore computing the $q(y)$ update amounts to re-running forward backwards on the message where there is an expected potential term which involves the belief over other variables. Note that the popularity and consensus potentials (ϕ_{POP} and ϕ_{CON}) decompose over individual message tokens so this can be tractably computed.

$q(A)$ **update:** The update for individual record alignment reduces to being log-proportional to the expected popularity and consensus potentials.

$q(R^\ell_k)$ **update:** The update for the record field distribution is the most complex factor of the three. It requires computing expected similarity with other record field values (the ϕ_{UNQ} potential) and looping over all messages to accumulate a contribution from each, weighted by the probability that it is aligned to the target record.

5.1 Initializing Factors

Since a uniform initialization of all factors is a saddle-point of the objective, we opt to initialize the $q(y)$ factors with the marginals obtained using just the CRF parameters, accomplished by running forwards-backwards on all messages using only the

ϕ_{SEQ} potentials. The $q(R)$ factors are initialized randomly and then biased with the output of our baseline model. The $q(A)$ factor is initialized to uniform plus a small amount of noise.

To simplify inference, we pre-compute a finite set of values that each R_k^ℓ is allowed to take, conditioned on the corpus. To do so, we run the CRF component of our model (ϕ_{SEQ}) over the corpus and extract, for each ℓ, all spans that have a token-level probability of being labeled ℓ greater than $\lambda = 0.1$. We further filter this set down to only values that occur at least twice in the corpus.

This simplification introduces sparsity that we take advantage of during inference to speed performance. Because each term in ϕ_{POP} and ϕ_{CON} includes an indicator function based on a token match between a field-value and a message, knowing the possible values $\mathbf{v}$ of each R_k^ℓ enables us to precompute the combinations of (x, ℓ, v) for which nonzero factor values are possible. For each such tuple, we can also precompute the best alignment position k for each token x^j.

6 Evaluation Setup

Data We apply our approach to construct a database of concerts in New York City. We used Twitter's public API to collect roughly 4.7 Million tweets across three weekends that we subsequently filter down to 5,800 messages. The messages have an average length of 18 tokens, and the corpus vocabulary comprises 468,000 unique words[6]. We obtain labeled gold records using data scraped from the `NYC.com` music event guide; totaling 110 extracted records. Each gold record had two fields of interest: ARTIST and VENUE.

The first weekend of data (messages and events) was used for training and the second two weekends were used for testing.

Preprocessing Only a small fraction of Twitter messages are relevant to the target extraction task. Directly processing the raw unfiltered stream would prohibitively increase computational costs and make learning more difficult due to the noise inherent in the data. To focus our efforts on the promising portion of the stream, we perform two types of filtering. First, we only retain tweets whose authors list some variant of New York as their location in their profile. Second, we employ a MIRA-based binary classifier (Ritter et al., 2010) to predict whether a message mentions a concert event. After training on 2,000 hand-annotated tweets, this classifier achieves an F_1 of 46.9 (precision of 35.0 and recall of 71.0) when tested on 300 messages. While the two-stage filtering does not fully eliminate noise in the input stream, it greatly reduces the presence of irrelevant messages to a manageable 5,800 messages without filtering too many 'signal' tweets.

We also filter our gold record set to include only records in which each field value occurs at least once somewhere in the corpus, as these are the records which are possible to learn given the input. This yields 11 training and 31 testing records.

Training The first weekend of data (2,184 messages and 11 records after preprocessing) is used for training. As mentioned in Section 4, the only learned parameters in our model are those associated with the sequence labeling factor ϕ_{SEQ}. While it is possible to train these parameters via direct annotation of messages with label sequences, we opted instead to use a simple approach where message tokens from the training weekend are labeled via their intersection with gold records, often called "distant supervision" (Mintz et al., 2009b). Concretely, we automatically label message tokens in the training corpus with either the ARTIST or VENUE label if they belonged to a sequence that matched a gold record field, and with NONE otherwise. This is the only use that is made of the gold records throughout training. θ_{SEQ} parameters are trained using this labeling with a standard conditional likelihood objective.

Testing The two weekends of data used for testing totaled 3,662 tweets after preprocessing and 31 gold records for evaluation. The two weekends were tested separately and their results were aggregated across weekends.

Our model assumes a fixed number of records $K = 130$.[7] We rank these records according to a heuristic ranking function that favors the uniqueness of a record's field values across the set and the number of messages in the testing corpus that have

[6] Only considering English tweets and not counting user names (so-called -mentions.)

[7] Chosen based on the training set

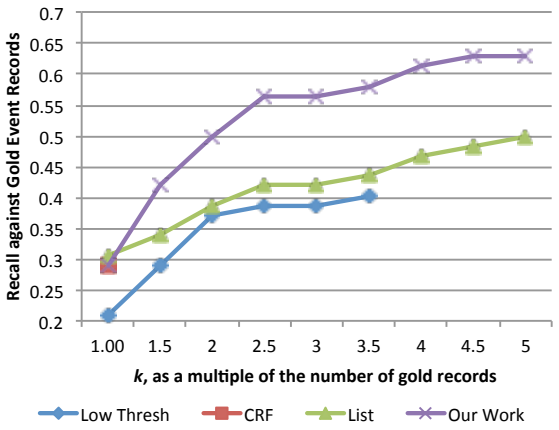


Figure 5: Recall against the gold records. The horizontal axis is the number of records kept from the ranked model output, as a multiple of the number of golds. The CRF lines terminate because of low record yield.

token overlap with these values. This ranking function is intended to push garbage collection records to the bottom of the list. Finally, we retain the top k records, throwing away the rest. Results in Section 7 are reported as a function of this k.

Baseline We compare our system against three baselines that employ a voting methodology similar to Mann and Yarowsky (2005). The baselines label each message and then extract one record for each combination of labeled phrases. Each extraction is considered a vote for that record's existence, and these votes are aggregated across all messages.

Our *List Baseline* labels messages by finding string overlaps against a list of musical artists and venues scraped from web data (the same lists used as features in our CRF component). The *CRF Baseline* is most similar to Mann and Yarowsky (2005)'s CRF Voting method and uses the maximum likelihood CRF labeling of each message. The *Low Threshold Baseline* generates all possible records from labelings with a token-level likelihood greater than $\lambda = 0.1$. The output of these baselines is a set of records ranked by the number of votes cast for each, and we perform our evaluation against the top k of these records.

7 Evaluation

The evaluation of record construction is challenging because many induced music events discussed in Twitter messages are not in our gold data set; our gold records are precise but incomplete. Because of this, we evaluate recall and precision separately. Both evaluations are performed using hard zero-one loss at record level. This is a harsh evaluation criterion, but it is realistic for real-world use.

Recall We evaluate recall, shown in Figure 5, against the gold event records for each weekend. This shows how well our model could do at replacing the a city event guide, providing Twitter users chat about events taking place.

We perform our evaluation by taking the top k records induced, performing a stable marriage matching against the gold records, and then evaluating the resulting matched pairs. Stable marriage matching is a widely used approach that finds a bipartite matching between two groups such that no pairing exists in which both participants would prefer some other pairing (Irving et al., 1987). With our hard loss function and no duplicate gold records, this amounts to the standard recall calculation. We choose this bipartite matching technique because it generalizes nicely to allow for other forms of loss calculation (such as token-level loss).

Precision To evaluate precision we assembled a list of the distinct records produced by all models and then manually determined if each record was correct. This determination was made blind to which model produced the record. We then used this aggregate list of correct records to measure precision for each individual model, shown in Figure 6.

By construction, our baselines incorporate a hard constraint that each relation learned must be expressed in entirety in at least one message. Our model only incorporates a soft version of this constraint via the ϕ_{CON} factor, but this constraint clearly has the ability to boost precision. To show it's effect, we additionally evaluate our model, labeled *Our Work + Con*, with this constraint applied in hard form as an output filter.

The downward trend in precision that can be seen in Figure 6 is the effect of our ranking algorithm, which attempts to push garbage collection records towards the bottom of the record list. As we incorporate these records, precision drops. These lines trend up for two of the baselines because the rank-

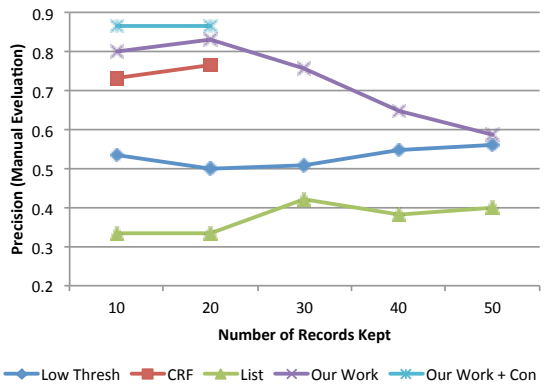


Figure 6: Precision, evaluated manually by cross-referencing model output with event mentions in the input data. The CRF and hard-constrained consensus lines terminate because of low record yield.

ing heuristic is not as effective for them.

These graphs confirm our hypothesis that we gain significant benefit by intertwining constraints on extraction consistency in the learning process, rather than only using this constraint to filter output.

7.1 Analysis

One persistent problem is a popular phrase appearing in many records, such as the value "New York" filling many ARTIST slots. The uniqueness factor θ_{UNQ} helps control this behavior, but it is a relatively blunt instrument. Ideally, our model would learn, for each field ℓ, the degree to which duplicate values are permitted. It is also possible that by learning, rather than hand-tuning, the θ_{CON}, θ_{POP}, and θ_{UNQ} parameters, our model could find a balance that permits the proper level of duplication for a particular domain.

Other errors can be explained by the lack of constituent features in our model, such as the selection of VENUE values that do not correspond to noun phrases. Further, semantic features could help avoid learning syntactically plausible artists like "Screw the Rain" because of the message:

Screw the rain$_{Artist}$! Grab an umbrella and head down to **Webster Hall**$_{Venue}$ for some American rock and roll.

Our model's soft string comparison-based clustering can be seen at work when our model uncovers records that would have been impossible without this approach. One such example is correcting the misspelling of venue names (e.g. *Terminal Five* → *Terminal 5*) even when no message about the event spells the venue correctly.

Still, the clustering can introduce errors by combining messages that provide orthogonal field contributions yet have overlapping tokens (thus escaping the penalty of the consistency factor). An example of two messages participating in this scenario is shown below; the shared term "holiday" in the second message gets relabeled as ARTIST:

Come check out the **holiday cheer**$_{Artist}$ parkside is bursting..
Pls tune in to TV **Guide Network**$_{Venue}$ TONIGHT at 8 pm for 25 Most Hilarious *Holiday* TV Moments...

While our experiments utilized binary relations, we believe our general approach should be useful for n-ary relation recovery in the social media domain. Because short messages are unlikely to express high arity relations completely, tying extraction and clustering seems an intuitive solution. In such a scenario, the record consistency constraints imposed by our model would have to be relaxed, perhaps examining pairwise argument consistency instead.

8 Conclusion

We presented a novel model for record extraction from social media streams such as Twitter. Our model operates on a noisy feed of data and extracts canonical records of events by aggregating information across multiple messages. Despite the noise of irrelevant messages and the relatively colloquial nature of message language, we are able to extract records with relatively high accuracy. There is still much room for improvement using a broader array of features on factors.

9 Acknowledgements

The authors gratefully acknowledge the support of the DARPA Machine Reading Program under AFRL prime contract no. FA8750-09-C-0172. Any opinions, findings, and conclusions expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA, AFRL, or the US government. Thanks also to Tal Wagner for his development assistance and the MIT NLP group for their helpful comments.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of DL*.

Razvan C. Bunescu and Raymond J. Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of the ACL*.

J Eisenstein, B O'Connor, and N Smith.... 2010. A latent variable model for geographic lexical variation. *Proceedings of the 2010 ...*, Jan.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of ACL*.

Robert W. Irving, Paul Leather, and Dan Gusfield. 1987. An efficient algorithm for the optimal stable marriage. *J. ACM*, 34:532–543, July.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of International Conference of Machine Learning (ICML)*, pages 282–289.

P. Liang and D. Klein. 2007. Structured Bayesian nonparametric models with variational inference (tutorial). In *Association for Computational Linguistics (ACL)*.

Gideon S. Mann and David Yarowsky. 2005. Multi-field information extraction and cross-document fusion. In *Proceeding of the ACL*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009a. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL/IJCNLP*.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009b. Distant supervision for relation extraction without labeled data. In *Proceedings of the ACL*, pages 1003–1011.

A Ritter, C Cherry, and B Dolan. 2010. Unsupervised modeling of twitter conversations. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of HLT/NAACL*.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of COLING*.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010a. Collective cross-document relation extraction without labelled data. In *Proceedings of the EMNLP*, pages 1013–1023.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010b. Cross-document relation extraction without labelled data. In *Proceedings of EMNLP*.

Jun Zhu, Zaiqing Nie, Xiaojing Liu, Bo Zhang, and Ji-Rong Wen. 2009. StatSnowball: a statistical approach to extracting entity relationships. In *Proceedings of WWW*.

How do you pronounce your name? Improving G2P with transliterations

Aditya Bhargava and Grzegorz Kondrak
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada, T6G 2E8
{abhargava,kondrak}@cs.ualberta.ca

Abstract

Grapheme-to-phoneme conversion (G2P) of names is an important and challenging problem. The correct pronunciation of a name is often reflected in its transliterations, which are expressed within a different phonological inventory. We investigate the problem of using transliterations to correct errors produced by state-of-the-art G2P systems. We present a novel re-ranking approach that incorporates a variety of score and n-gram features, in order to leverage transliterations from multiple languages. Our experiments demonstrate significant accuracy improvements when re-ranking is applied to n-best lists generated by three different G2P programs.

1 Introduction

Grapheme-to-phoneme conversion (G2P), in which the aim is to convert the orthography of a word to its pronunciation (phonetic transcription), plays an important role in speech synthesis and understanding. Names, which comprise over 75% of unseen words (Black et al., 1998), present a particular challenge to G2P systems because of their high pronunciation variability. Guessing the correct pronunciation of a name is often difficult, especially if they are of foreign origin; this is attested by the *ad hoc* transcriptions which sometimes accompany new names introduced in news articles, especially for international stories with many foreign names.

Transliterations provide a way of disambiguating the pronunciation of names. They are more abundant than phonetic transcriptions, for example when news items of international or global significance are reported in multiple languages. In addition, writing scripts such as Arabic, Korean, or Hindi are more consistent and easier to identify than various phonetic transcription schemes. The process of transliteration, also called *phonetic translation* (Li et al., 2009b), involves "sounding out" a name and then finding the closest possible representation of the sounds in another writing script. Thus, the correct pronunciation of a name is partially encoded in the form of the transliteration. For example, given the ambiguous letter-to-phoneme mapping of the English letter *g*, the initial phoneme of the name *Gershwin* may be predicted by a G2P system to be either /ɡ/ (as in *Gertrude*) or /ʤ/ (as in *Gerald*). The transliterations of the name in other scripts provide support for the former (correct) alternative.

Although it seems evident that transliterations should be helpful in determining the correct pronunciation of a name, designing a system that takes advantage of this insight is not trivial. The main source of the difficulty stems from the differences between the phonologies of distinct languages. The mappings between phonemic inventories are often complex and context-dependent. For example, because Hindi has no /w/ sound, the transliteration of *Gershwin* instead uses a symbol that represents the phoneme /ʋ/, similar to the /v/ phoneme in English. In addition, converting transliterations into phonemes is often non-trivial; although few orthographies are as inconsistent as that of English, this is effectively the G2P task for the particular language in question.

In this paper, we demonstrate that leveraging transliterations can, in fact, improve the grapheme-to-phoneme conversion of names. We propose a novel system based on discriminative re-ranking that is capable of incorporating multiple transliterations. We show that simplistic approaches to the problem

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 399–408,
Portland, Oregon, June 19-24, 2011.

fail to achieve the same goal, and that transliterations from multiple languages are more helpful than from a single language. Our approach can be combined with any G2P system that produces n-best lists instead of single outputs. The experiments that we perform demonstrate significant error reduction for three very different G2P base systems.

2 Improving G2P with transliterations

2.1 Problem definition

In both G2P and machine transliteration, we are interested in learning a function that, given an input sequence x, produces an output sequence y. In the G2P task, x is composed of graphemes and y is composed of phonemes; in transliteration, both sequences consist of graphemes but they represent different writing scripts. Unlike in machine translation, the monotonicity constraint is enforced; i.e., we assume that x and y can be aligned without the alignment links crossing each other (Jiampojamarn and Kondrak, 2010). We assume that we have available a base G2P system that produces an n-best list of outputs with a corresponding list of confidence scores. The goal is to improve the base system's performance by applying existing transliterations of the input x to re-rank the system's n-best output list.

2.2 Similarity-based methods

A simple and intuitive approach to improving G2P with transliterations is to select from the n-best list the output sequence that is most similar to the corresponding transliteration. For example, the Hindi transliteration in Figure 1 is arguably closest in perceptual terms to the phonetic transcription of the second output in the n-best list, as compared to the other outputs. One obvious problem with this method is that it ignores the relative ordering of the n-best lists and their corresponding scores produced by the base system.

A better approach is to combine the similarity score with the output score from the base system, allowing it to contribute an estimate of confidence in its output. For this purpose, we apply a linear combination of the two scores, where a single parameter λ, ranging between zero and one, determines the relative weight of the scores. The exact value of λ can be optimized on a training set. This approach is similar to the method used by Finch and Sumita (2010) to combine the scores of two different machine transliteration systems.

2.3 Measuring similarity

The approaches presented in the previous section crucially depend on a method for computing the similarity between various symbol sequences that represent the same word. If we have a method of converting transliterations to phonetic representations, the similarity between two sequences of phonemes can be computed with a simple method such as normalized edit distance or the longest common subsequence ratio, which take into account the number and position of identical phonemes. Alternatively, we could apply a more complex approach, such as ALINE (Kondrak, 2000), which computes the distance between pairs of phonemes. However, the implementation of a conversion program would require ample training data or language-specific expertise.

A more general approach is to skip the transcription step and compute the similarity between phonemes and graphemes directly. For example, the edit distance function can be learned from a training set of transliterations and their phonetic transcriptions (Ristad and Yianilos, 1998). In this paper, we apply M2M-ALIGNER (Jiampojamarn et al., 2007), an unsupervised aligner, which is a many-to-many generalization of the learned edit distance algorithm. M2M-ALIGNER was originally designed to align graphemes and phonemes, but can be applied to discover the alignment between any sets of symbols (given training data). The logarithm of the probability assigned to the optimal alignment can then be interpreted as a similarity measure between the two sequences.

2.4 Discriminative re-ranking

The methods described in Section 2.2, which are based on the similarity between outputs and transliterations, are difficult to generalize when multiple transliterations of a single name are available. A linear combination is still possible but in this case optimizing the parameters would no longer be straightforward. Also, we are interested in utilizing other features besides sequence similarity.

The SVM re-ranking paradigm offers a solution

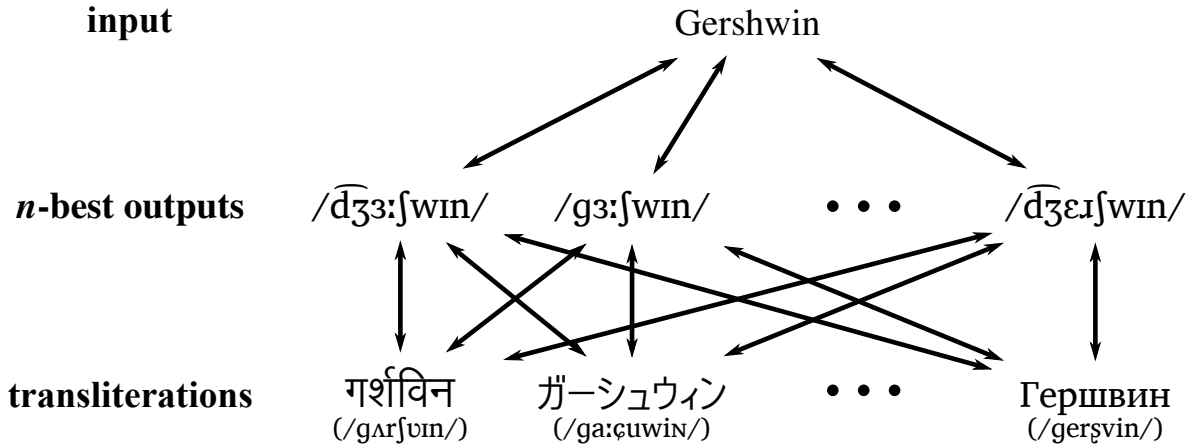


Figure 1: An example name showing the data used for feature construction. Each arrow links a pair used to generate features, including n-gram and score features. The score features use similarity scores for transliteration-transcription pairs and system output scores for input-output pairs. One feature vector is constructed for each system output.

to the problem. Our re-ranking system is informed by a large number of features, which are based on scores and n-grams. The scores are of three types:

1. The scores produced by the base system for each output in the n-best list.
2. The similarity scores between the outputs and each available transliteration.
3. The differences between scores in the n-best lists for both (1) and (2).

Our set of binary n-gram features includes those used for DIRECTL+ (Jiampojamarn et al., 2010). They can be divided into four types:

1. The context features combine output symbols (phonemes) with n-grams of varying sizes in a window of size c centred around a corresponding position on the input side.
2. The transition features are bigrams on the output (phoneme) side.
3. The linear chain features combine the context features with the bigram transition features.
4. The joint n-gram features are n-grams containing both input and output symbols.

We apply the features in a new way: instead of being applied strictly to a given input-output set, we expand their use across many languages and use all of them simultaneously. We apply the n-gram features across all transliteration-transcription pairs in addition to the usual input-output pairs corresponding to the n-best lists. Figure 1 illustrates the set of pairs used for feature generation.

In this paper, we augment the n-gram features by a set of *reverse* features. Unlike a traditional G2P generator, our re-ranker has access to the outputs produced by the base system. By swapping the input and the output side, we can add reverse context and linear-chain features. Since the n-gram features are also applied to transliteration-transcription pairs, the reverse features enable us to include features which bind a variety of n-grams in the transliteration string with a single corresponding phoneme.

The construction of n-gram features presupposes a fixed alignment between the input and output sequences. If the base G2P system does not provide input-output alignments, we use M2M-ALIGNER for this purpose. The transliteration-transcription pairs are also aligned by M2M-ALIGNER, which at the same time produces the corresponding similarity scores. (We set a lower limit of -100 on the M2M-ALIGNER scores.) If M2M-ALIGNER is unable to produce an alignment, we indicate this with a binary feature that is included with the n-gram features.

3 Experiments

We perform several experiments to evaluate our transliteration-informed approaches. We test simple

similarity-based approaches on single-transliteration data, and evaluate our SVM re-ranking approach against this as well. We then test our approach using all available transliterations. Relevant code and scripts required to reproduce our experimental results are available online[1].

3.1 Data & setup

For pronunciation data, we extracted all names from the Combilex corpus (Richmond et al., 2009). We discarded all diacritics, duplicates and multi-word names, which yielded 10,084 unique names. Both the similarity and SVM methods require transliterations for identifying the best candidates in the n-best lists. They are therefore trained and evaluated on the subset of the G2P corpus for which transliterations available. Naturally, allowing transliterations from all languages results in a larger corpus than the one obtained by the intersection with transliterations from a single language.

For our experiments, we split the data into 10% for testing, 10% for development, and 80% for training. The development set was used for initial tests and experiments, and then for our final results the training and development sets were combined into one set for final system training. For SVM re-ranking, during both development and testing we split the training set into 10 folds; this is necessary when training the re-ranker as it must have system output scores that are representative of the scores on unseen data. We ensured that there was never any overlap between the training and testing data for all trained systems.

Our transliteration data come from the shared tasks on transliteration at the 2009 and 2010 Named Entities Workshops (Li et al., 2009a; Li et al., 2010). We use all of the 2010 English-source data plus the English-to-Russian data from 2009, which makes nine languages in total. In cases where the data provide alternative transliterations for a given input, we keep only one; our preliminary experiments indicated that including alternative transliterations did not improve performance. It should be noted that these transliteration corpora are noisy: Jiampojamarn et al. (2009) note a significant increase in English-to-Hindi transliteration performance with a simple cleaning of the data.

Language	Corpus size	Overlap
Bengali	12,785	1,840
Chinese	37,753	4,713
Hindi	12,383	2,179
Japanese	26,206	4,773
Kannada	10,543	1,918
Korean	6,761	3,015
Russian	6,447	487
Tamil	10,646	1,922
Thai	27,023	5,436

Table 1: The number of unique single-word entries in the transliteration corpora for each language and the amount of common data (overlap) with the pronunciation data.

Our tests involving transliterations from multiple languages are performed on the set of names for which we have both the pronunciation and transliteration data. There are 7,423 names in the G2P corpus for which at least one transliteration is available. Table 1 lists the total size of the transliteration corpora as well as the amount of overlap with the G2P data. Note that the base G2P systems are trained using all 10,084 names in the corpus as opposed to only the 7,423 names for which there are transliterations available. This ensures that the G2P systems have more training data to provide the best possible base performance.

For our single-language experiments, we normalize the various scores when tuning the linear combination parameter λ so that we can compare values across different experimental conditions. For SVM re-ranking, we directly implement the method of Joachims (2002) to convert the re-ranking problem into a classification problem, and then use the very fast LIBLINEAR (Fan et al., 2008) to build the SVM models. Optimal hyperparameter values were determined during development.

We evaluate using word accuracy, the percentage of words for which the pronunciations are correctly predicted. This measure marks pronunciations that are even slightly different from the correct one as incorrect, so even a small change in pronunciation that might be acceptable or even unnoticeable to humans would count against the system's performance.

[1] http://www.cs.ualberta.ca/~ab31/g2p-tl-rr

3.2 Base systems

It is important to test multiple base systems in order to ensure that any gain in performance applies to the task in general and not just to a particular system. We use three G2P systems in our tests:

1. FESTIVAL (FEST), a popular speech synthesis package, which implements G2P conversion with CARTs (decision trees) (Black et al., 1998).

2. SEQUITUR (SEQ), a generative system based on the joint n-gram approach (Bisani and Ney, 2008).

3. DIRECTL+ (DTL), the discriminative system on which our n-gram features are based (Jiampojamarn et al., 2010).

All systems are capable of providing n-best output lists along with scores for each output, although for FESTIVAL they had to be constructed from the list of output probabilities for each input character.

We run DIRECTL+ with all of the features described in (Jiampojamarn et al., 2010) (i.e., context features, transition features, linear chain features, and joint n-gram features). System parameters, such as maximum number of iterations, were determined during development. For SEQUITUR, we keep default options except for the enabling of the 10 best outputs and we convert the probabilities assigned to the outputs to log-probabilities. We set SEQUITUR's joint n-gram order to 6 (this was also determined during development).

Note that the three base systems differ slightly in terms of the alignment information that they provide in their outputs. FESTIVAL operates letter-by-letter, so we use the single-letter inputs with the phoneme outputs as the aligned units. DIRECTL+ specifies many-to-many alignments in its output. For SEQUITUR, however, since it provides no information regarding the output structure, we use M2M-ALIGNER to induce alignments for n-gram feature generation.

3.3 Transliterations from a single language

The goal of the first experiment is to compare several similarity-based methods, and to determine how they compare to our re-ranking approach. In order to find the similarity between phonetic transcriptions, we use the two different methods described in Section 2.2: ALINE and M2M-ALIGNER. We further test the use of a linear combination of the similarity scores with the base system's score so that its confidence information can be taken into account; the linear combination weight is determined from the training set. These methods are referred to as ALINE+BASE and M2M+BASE. For these experiments, our training and testing sets are obtained by intersecting our G2P training and testing sets respectively with the Hindi transliteration corpus, yielding 1,950 names for training and 229 names for testing.

Since the similarity-based methods are designed to incorporate homogeneous same-script transliterations, we can only run this experiment on one language at a time. Furthermore, ALINE operates on phoneme sequences, so we first need to convert the transliterations to phonemes. An alternative would be to train a proper G2P system, but this would require a large set of word-pronunciation pairs. For this experiment, we choose Hindi, for which we constructed a rule-based G2P converter. Aside from simple one-to-one mapping (romanization) rules, the converter has about ten rules to adjust for context.

For these experiments, we apply our SVM reranking method in two ways:

1. Using only Hindi transliterations (referred to as SVM-HINDI).

2. Using all available languages (referred to as SVM-ALL).

In both cases, the test set is restricted to the same 229 names, in order to provide a valid comparison.

Table 2 presents the results. Regardless of the choice of the similarity function, the simplest approaches fail in a spectacular manner, significantly reducing the accuracy with respect to the base system. The linear combination methods give mixed results, improving the accuracy for FESTIVAL but not for SEQUITUR or DIRECTL+ (although the differences are not statistically significant). However, they perform much better than the methods based on similarity scores alone as they are able to take advantage of the base system's output scores. If we look at the values of λ that provide the best performance

	Base system		
	FEST	SEQ	DTL
Base	58.1	67.3	71.6
ALINE	28.0	26.6	27.5
M2M	39.3	36.2	36.2
ALINE+BASE	58.5	65.9	71.2
M2M+BASE	58.5	66.4	70.3
SVM-HINDI	63.3	69.0	69.9
SVM-ALL	68.6	72.5	75.6

Table 2: Word accuracy (in percentages) of various methods when only Hindi transliterations are used.

on the training set, we find that they are higher for the stronger base systems, indicating more reliance on the base system output scores. For example, for ALINE+BASE the FESTIVAL-based system has $\lambda = 0.58$ whereas the DIRECTL+-based system has $\lambda = 0.81$. Counter-intuitively, the ALINE+BASE and M2M+BASE methods are unable to improve upon SEQUITUR or DIRECTL+. We would expect to achieve at least the base system's performance, but disparities between the training and testing sets prevent this.

The two SVM-based methods achieve much better results. SVM-ALL produces impressive accuracy gains for all three base systems, while SVM-HINDI yields smaller (but still statistically significant) improvements for FESTIVAL and SEQUITUR. These results suggest that our re-ranking method provides a bigger boost to systems built with different design principles than to DIRECTL+ which utilizes a similar set of features. On the other hand, the results also show that the information obtained by consulting a single transliteration may be insufficient to improve an already high-performing G2P converter.

3.4 Transliterations from multiple languages

Our second experiment expands upon the first; we use all available transliterations instead of being restricted to one language. This rules out the simple similarity-based approaches, but allows us to test our re-ranking approach in a way that fully utilizes the available data. We test three variants of our transliteration-informed SVM re-ranking approach,

	Base system		
	FEST	SEQ	DTL
Base	55.3	66.5	70.8
SVM-SCORE	62.1	68.4	71.0
SVM-N-GRAM	66.2	72.5	73.8
SVM-ALL	67.2	73.4	74.3

Table 3: Word accuracy of the base system versus the re-ranking variants with transliterations from multiple languages.

which differ with respect to the set of included features:

1. SVM-SCORE includes only the three types of score features described in Section 2.4.
2. SVM-N-GRAM uses only the n-gram features.
3. SVM-ALL is the full system that combines the score and n-gram features.

The objective is to determine the degree to which each of the feature classes contributes to the overall results. Because we are using all available transliterations, we achieve much greater coverage over our G2P data than in the previous experiment; in this case, our training set consists of 6,660 names while the test set has 763 names.

Table 3 presents the results. Note that the baseline accuracies are somewhat lower than in Table 2 because of the different test set. We find that, when using all features, the SVM re-ranker can provide a very impressive error reduction over FESTIVAL (26.7%) and SEQUITUR (20.7%) and a smaller but still significant ($p < 0.01$ with the McNemar test) error reduction over DIRECTL+ (12.1%).

When we consider our results using only the score and n-gram features, we can see that, interestingly, the n-gram features are most important. We draw a further conclusion from our results: consider the large disparity in improvements over the base systems. This indicates that FESTIVAL and SEQUITUR are benefiting from the DIRECTL+-style features used in the re-ranking. Without the n-gram features, however, there is still a significant improvement over FESTIVAL, demonstrating that the scores *do* provide useful information. In this case there is

no way for DIRECTL+-style information to make its way into the re-ranking; the process is based purely on the transliterations and their similarities with the transcriptions in the output lists, indicating that the system is capable of extracting useful information directly from transliterations. In the case of DIRECTL+, the transliterations help through the n-gram features rather than the score features; this is probably because the crucial feature that signals the inability of M2M-ALIGNER to align a given transliteration-transcription pair belongs to the set of the n-gram features. Both the n-gram features and score features are dependent on the alignments, but they differ in that the n-gram features allow weights to be learned for local n-gram pairs whereas the score features are based on global information, providing only a single feature for a given transliteration-transcription pair. The two therefore overlap to some degree, although the score features still provide useful information via probabilities learned during the alignment training process.

A closer look at the results provides additional insight into the operation of our re-ranking system. For example, consider the name *Bacchus*, which DIRECTL+ incorrectly converts into /bæktʃəs/. The most likely reason why our re-ranker selects instead the correct pronunciation /bækəs/ is that M2M-ALIGNER fails to align three of the five available transliterations with /bæktʃəs/. Such alignment failures are caused by a lack of evidence for the mapping of the grapheme representing the sound /k/ in the transliteration training data with the phoneme /tʃ/. In addition, the lack of alignments prevents any n-gram features from being enabled.

Considering the difficulty of the task, the top accuracy of almost 75% is quite impressive. In fact, many instances of human transliterations in our corpora are clearly incorrect. For example, the Hindi transliteration of *Bacchus* contains the /tʃ/ consonant instead of the correct /k/. Moreover, our strict evaluation based on word accuracy counts all system outputs that fail to exactly match the dictionary data as errors. The differences are often very minor and may reflect an alternative pronunciation. The *phoneme* accuracy[2] of our best result is 93.1%, which provides some idea of how similar the predicted pronunciation is to the correct one.

[2]The phoneme accuracy is calculated from the minimum edit distance between the predicted and correct pronunciations.

# TL	# Entries	Improvement
≤ 1	111	0.9
≤ 2	266	3.0
≤ 3	398	3.8
≤ 4	536	3.2
≤ 5	619	2.8
≤ 6	685	3.4
≤ 7	732	3.7
≤ 8	762	3.5
≤ 9	763	3.5

Table 4: Absolute improvement in word accuracy (%) over the base system (DIRECTL+) of the SVM re-ranker for various numbers of available transliterations.

3.5 Effect of multiple transliterations

One motivating factor for the use of SVM re-ranking was the ability to incorporate multiple transliteration languages. But how important is it to use more than one language? To examine this question, we look particularly at the sets of names having at most k transliterations available. Table 4 shows the results with DIRECTL+ as the base system. Note that the number of names with more than five transliterations was small. Importantly, we see that the increase in performance when only one transliteration is available is so small as to be insignificant. From this, we can conclude that obtaining improvement on the basis of a single transliteration is difficult in general. This corroborates the results of the experiment described in Section 3.3, where we used only Hindi transliterations.

4 Previous work

There are three lines of research that are relevant to our work: (1) G2P in general; (2) G2P on names; and (3) combining diverse data sources and/or systems.

The two leading approaches to G2P are represented by SEQUITUR (Bisani and Ney, 2008) and DIRECTL+ (Jiampojamarn et al., 2010). Recent comparisons suggests that the former obtains somewhat higher accuracy, especially when it includes joint n-gram features (Jiampojamarn et al., 2010). Systems based on decision trees are far behind. Our

results confirm this ranking.

Names can present a particular challenge to G2P systems. Kienappel and Kneser (2001) reported a higher error rate for German names than for general words, while on the other hand Black et al. (1998) report similar accuracy on names as for other types of English words. Yang et al. (2006) and van den Heuvel et al. (2007) post-process the output of a general G2P system with name-specific phoneme-to-phoneme (P2P) systems. They find significant improvement using this method on data sets consisting of Dutch first names, family names, and geographical names. However, it is unclear whether such an approach would be able to improve the performance of the current state-of-the-art G2P systems. In addition, the P2P approach works only on single outputs, whereas our re-ranking approach is designed to handle n-best output lists.

Although our approach is (to the best of our knowledge) the first to use different *tasks* (G2P and transliteration) to inform each other, this is conceptually similar to model and system combination approaches. In statistical machine translation (SMT), methods that incorporate translations from other languages (Cohn and Lapata, 2007) have proven effective in low-resource situations: when phrase translations are unavailable for a certain language, one can look at other languages where the translation *is* available and then translate from that language. A similar pivoting approach has also been applied to machine transliteration (Zhang et al., 2010). Notably, the focus of these works have been on cases in which there are less data available; they also modify the generation process directly, rather than operating on existing outputs as we do. Ultimately, a combination of the two approaches is likely to give the best results.

Finch and Sumita (2010) combine two very different approaches to transliteration using simple linear interpolation: they use SEQUITUR's n-best outputs and re-rank them using a linear combination of the original SEQUITUR score and the score for that output of a phrased-based SMT system. The linear weights are hand-tuned. We similarly use linear combinations, but with many more scores and other features, necessitating the use of SVMs to determine the weights. Importantly, we combine different *data types* where they combine different *systems*.

5 Conclusions & future work

In this paper, we explored the application of transliterations to G2P. We demonstrated that transliterations have the potential for helping choose between n-best output lists provided by standard G2P systems. Simple approaches based solely on similarity do not work when tested using a single transliteration language (Hindi), necessitating the use of smarter methods that can incorporate multiple transliteration languages. We apply SVM re-ranking to this task, enabling us to use a variety of features based not only on similarity scores but on n-grams as well. Our method shows impressive error reductions over the popular FESTIVAL system and the generative joint n-gram SEQUITUR system. We also find significant error reduction using the state-of-the-art DIRECTL+ system. Our analysis demonstrated that it is essential to provide the re-ranking system with transliterations from multiple languages in order to mitigate the differences between phonological inventories and smooth out noise in the transliterations.

In the future, we plan to generalize our approach so that it can be applied to the task of generating transliterations, and to combine data from distinct G2P dictionaries. The latter task is related to the notion of domain adaptation. We would also like to apply our approach to web data; we have shown that it is possible to use noisy transliteration data, so it may be possible to leverage the noisy *ad hoc* pronunciation data as well. Finally, we plan to investigate earlier integration of such external information into the G2P process for single systems; while we noted that re-ranking provides a general approach applicable to any system that can generate n-best lists, there is a limit as to what re-ranking can do, as it relies on the correct output existing in the n-best list. Modifying existing systems would provide greater potential for improving results even though the changes would be necessarily system-specific.

Acknowledgements

We are grateful to Sittichai Jiampojamarn and Shane Bergsma for the very helpful discussions. This research was supported by the Natural Sciences and Engineering Research Council of Canada.

References

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434–451, May.

Alan W. Black, Kevin Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules. In *The Third ESCA/COCOSDA Workshop (ETRW) on Speech Synthesis*, Jenolan Caves House, Blue Mountains, New South Wales, Australia, November.

Trevor Cohn and Mirella Lapata. 2007. Machine translation by triangulation: Making effective use of multi-parallel corpora. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 728–735, Prague, Czech Republic, June. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Andrew Finch and Eiichiro Sumita. 2010. Transliteration using a phrase-based statistical machine translation system to re-score the output of a joint multigram model. In *Proceedings of the 2010 Named Entities Workshop (NEWS 2010)*, pages 48–52, Uppsala, Sweden, July. Association for Computational Linguistics.

Sittichai Jiampojamarn and Grzegorz Kondrak. 2010. Letter-phoneme alignment: An exploration. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 780–788, Uppsala, Sweden, July. Association for Computational Linguistics.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden Markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379, Rochester, New York, USA, April. Association for Computational Linguistics.

Sittichai Jiampojamarn, Aditya Bhargava, Qing Dou, Kenneth Dwyer, and Grzegorz Kondrak. 2009. DirecTL: a language independent approach to transliteration. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 28–31, Suntec, Singapore, August. Association for Computational Linguistics.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2010. Integrating joint n-gram features into a discriminative training framework. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 697–700, Los Angeles, California, USA, June. Association for Computational Linguistics.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142, Edmonton, Alberta, Canada. Association for Computing Machinery.

Anne K. Kienappel and Reinhard Kneser. 2001. Designing very compact decision trees for grapheme-to-phoneme transcription. In *EUROSPEECH-2001*, pages 1911–1914, Aalborg, Denmark, September.

Grzegorz Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 288–295, Seattle, Washington, USA, April.

Haizhou Li, A Kumaran, Vladimir Pervouchine, and Min Zhang. 2009a. Report of NEWS 2009 machine transliteration shared task. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 1–18, Suntec, Singapore, August. Association for Computational Linguistics.

Haizhou Li, A Kumaran, Min Zhang, and Vladimir Pervouchine. 2009b. Whitepaper of NEWS 2009 machine transliteration shared task. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 19–26, Suntec, Singapore, August. Association for Computational Linguistics.

Haizhou Li, A Kumaran, Min Zhang, and Vladimir Pervouchine. 2010. Report of NEWS 2010 transliteration generation shared task. In *Proceedings of the 2010 Named Entities Workshop (NEWS 2010)*, pages 1–11, Uppsala, Sweden, July. Association for Computational Linguistics.

Korin Richmond, Robert Clark, and Sue Fitt. 2009. Robust LTS rules with the Combilex speech technology lexicon. In *Proceedings of Interspeech*, pages 1295–1298, Brighton, UK, September.

Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string edit distance. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 20(5):522–532, May.

Henk van den Heuvel, Jean-Pierre Martens, and Nanneke Konings. 2007. G2P conversion of names. what can we do (better)? In *Proceedings of Interspeech*, pages 1773–1776, Antwerp, Belgium, August.

Qian Yang, Jean-Pierre Martens, Nanneke Konings, and Henk van den Heuvel. 2006. Development of a phoneme-to-phoneme (p2p) converter to improve the grapheme-to-phoneme (g2p) conversion of names. In

Proceedings of the 2006 International Conference on Language Resources and Evaluation, pages 2570–2573, Genoa, Italy, May.

Min Zhang, Xiangyu Duan, Vladimir Pervouchine, and Haizhou Li. 2010. Machine transliteration: Leveraging on third languages. In *Coling 2010: Posters*, pages 1444–1452, Beijing, China, August. Coling 2010 Organizing Committee.

Unsupervised Word Alignment with Arbitrary Features

Chris Dyer Jonathan Clark Alon Lavie Noah A. Smith
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{cdyer,jhclark,alavie,nasmith}@cs.cmu.edu

Abstract

We introduce a discriminatively trained, globally normalized, log-linear variant of the lexical translation models proposed by Brown et al. (1993). In our model, arbitrary, non-independent features may be freely incorporated, thereby overcoming the inherent limitation of generative models, which require that features be sensitive to the conditional independencies of the generative process. However, unlike previous work on discriminative modeling of word alignment (which also permits the use of arbitrary features), the parameters in our models are learned from unannotated parallel sentences, rather than from supervised word alignments. Using a variety of intrinsic and extrinsic measures, including translation performance, we show our model yields better alignments than generative baselines in a number of language pairs.

1 Introduction

Word alignment is an important subtask in statistical machine translation which is typically solved in one of two ways. The more common approach uses a generative translation model that relates bilingual string pairs using a latent alignment variable to designate which source words (or phrases) generate which target words. The parameters in these models can be learned straightforwardly from parallel sentences using EM, and standard inference techniques can recover most probable alignments (Brown et al., 1993). This approach is attractive because it only requires parallel training data. An alternative to the generative approach uses a discriminatively trained alignment model to predict word alignments in the parallel corpus. Discriminative models are attractive because they can incorporate arbitrary, overlapping features, meaning that errors observed in the predictions made by the model can be addressed by engineering new and better features. Unfortunately, *both* approaches are problematic, but in different ways.

In the case of discriminative alignment models, manual alignment data is required for training, which is problematic for at least three reasons. Manual alignments are notoriously difficult to create and are available only for a handful of language pairs. Second, manual alignments impose a commitment to a particular preprocessing regime; this can be problematic since the optimal segmentation for translation often depends on characteristics of the test set or size of the available training data (Habash and Sadat, 2006) or may be constrained by requirements of other processing components, such parsers. Third, the "correct" alignment annotation for different tasks may vary: for example, relatively denser or sparser alignments may be optimal for different approaches to (downstream) translation model induction (Lopez, 2008; Fraser, 2007).

Generative models have a different limitation: the joint probability of a particular setting of the random variables must factorize according to steps in a process that successively "generates" the values of the variables. At each step, the probability of some value being generated may depend only on the generation history (or a subset thereof), and the possible values a variable will take must form a locally normalized conditional probability distribution (CPD). While these locally normalized CPDs may be pa-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 409–419,
Portland, Oregon, June 19-24, 2011.

rameterized so as to make use of multiple, overlapping features (Berg-Kirkpatrick et al., 2010), the requirement that models factorize according to a particular generative process imposes a considerable restriction on the *kinds* of features that can be incorporated. When Brown et al. (1993) wanted to incorporate a fertility model to create their Models 3 through 5, the generative process used in Models 1 and 2 (where target words were generated one by one from source words independently of each other) had to be abandoned in favor of one in which each source word had to first decide how many targets it would generate.[1]

In this paper, we introduce a discriminatively trained, globally normalized log-linear model of lexical translation that can incorporate arbitrary, overlapping features, and use it to infer word alignments. Our model enjoys the usual benefits of discriminative modeling (e.g., parameter regularization, well-understood learning algorithms), but is trained entirely from parallel sentences without gold-standard word alignments. Thus, it addresses the two limitations of current word alignment approaches.

This paper is structured as follows. We begin by introducing our model (§2), and follow this with a discussion of tractability, parameter estimation, and inference using finite-state techniques (§3). We then describe the specific features we used (§4) and provide experimental evaluation of the model, showing substantial improvements in three diverse language pairs (§5). We conclude with an analysis of related prior work (§6) and a general discussion (§8).

2 Model

In this section, we develop a conditional model $p(\mathbf{t} \mid \mathbf{s})$ that, given a source language sentence $\mathbf{s}$ with length $m = |\mathbf{s}|$, assigns probabilities to a target sentence $\mathbf{t}$ with length n, where each word t_j is an element in the finite target vocabulary Ω. We begin by using the chain rule to factor this probability into two components, a translation model and a length model.

$$p(\mathbf{t} \mid \mathbf{s}) = p(\mathbf{t}, n \mid \mathbf{s}) = \underbrace{p(\mathbf{t} \mid \mathbf{s}, n)}_{\text{translation model}} \times \underbrace{p(n \mid \mathbf{s})}_{\text{length model}}$$

[1] Moore (2005) likewise uses this example to motivate the need for models that support arbitrary, overlapping features.

In the translation model, we then assume that each word t_j is a translation of one source word, or a special null token. We therefore introduce a latent *alignment* variable $\mathbf{a} = \langle a_1, a_2, \ldots, a_n \rangle \in [0, m]^n$, where $a_j = 0$ represents a special null token.

$$p(\mathbf{t} \mid \mathbf{s}, n) = \sum_{\mathbf{a}} p(\mathbf{t}, \mathbf{a} \mid \mathbf{s}, n)$$

So far, our model is identical to that of (Brown et al., 1993); however, we part ways here. Rather than using the chain rule to further decompose this probability and motivate opportunities to make independence assumptions, we use a log-linear model with parameters $\boldsymbol{\theta} \in \mathbb{R}^k$ and feature vector function $\mathbf{H}$ that maps each tuple $\langle \mathbf{a}, \mathbf{s}, \mathbf{t}, n \rangle$ into $\mathbb{R}^k$ to model $p(\mathbf{t}, \mathbf{a} \mid \mathbf{s}, n)$ directly:

$$p_{\boldsymbol{\theta}}(\mathbf{t}, \mathbf{a} \mid \mathbf{s}, n) = \frac{\exp \boldsymbol{\theta}^\top \mathbf{H}(\mathbf{t}, \mathbf{a}, \mathbf{s}, n)}{Z_{\boldsymbol{\theta}}(\mathbf{s}, n)}, \quad \text{where}$$

$$Z_{\boldsymbol{\theta}}(\mathbf{s}, n) = \sum_{\mathbf{t}' \in \Omega^n} \sum_{\mathbf{a}'} \exp \boldsymbol{\theta}^\top \mathbf{H}(\mathbf{t}', \mathbf{a}', \mathbf{s}, n)$$

Under some reasonable assumptions (a finite target vocabulary Ω and that all $\theta_k < \infty$), the partition function $Z_{\boldsymbol{\theta}}(\mathbf{s}, n)$ will always take on finite values, guaranteeing that $p(\mathbf{t}, \mathbf{a} \mid \mathbf{s}, n)$ is a proper probability distribution.

So far, we have said little about the length model. Since our intent here is to use the model for alignment, where both the target length and target string are observed, it will not be necessary to commit to any length model, even during training.

3 Tractability, Learning, and Inference

The model introduced in the previous section is extremely general, and it can incorporate features sensitive to any imaginable aspects of a sentence pair and their alignment, from linguistically inspired (e.g., an indicator feature for whether both the source and target sentences contain a verb), to the mundane (e.g., the probability of the sentence pair and alignment under Model 1), to the absurd (e.g., an indicator if $\mathbf{s}$ and $\mathbf{t}$ are palindromes of each other).

However, while our model can make use of arbitrary, overlapping features, when designing feature functions it is necessary to balance expressiveness and the computational complexity of the inference

algorithms used to reason under models that incorporate these features.[2] To understand this tradeoff, we assume that the random variables being modeled $(\mathbf{t}, \mathbf{a})$ are arranged into an undirected graph $\mathcal{G}$ such that the vertices represent the variables and the edges are specified so that the feature function $\mathbf{H}$ decomposes linearly over all the cliques C in $\mathcal{G}$,

$$\mathbf{H}(\mathbf{t}, \mathbf{a}, \mathbf{s}, n) = \sum_{C} \mathbf{h}(\mathbf{t}_C, \mathbf{a}_C, \mathbf{s}, n) \quad ,$$

where $\mathbf{t}_C$ and $\mathbf{a}_C$ are the components associated with subgraph C and $\mathbf{h}(\cdot)$ is a *local* feature vector function. In general, exact inference is exponential in the width of tree-decomposition of $\mathcal{G}$, but, given a fixed width, they can be solved in polynomial time using dynamic programming. For example, when the graph has a sequential structure, exact inference can be carried out using the familiar forward-backward algorithm (Lafferty et al., 2001). Although our features look at more structure than this, they are designed to keep treewidth low, meaning exact inference is still possible with dynamic programming. Figure 1 gives a graphical representation of our model as well as the more familiar generative (directed) variants. The edge set in the depicted graph is determined by the features that we use (§4).

3.1 Parameter Learning

To learn the parameters of our model, we select the $\boldsymbol{\theta}^*$ that minimizes the ℓ_1 regularized conditional log-likelihood of a set of training data $\mathcal{T}$:

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{\langle \mathbf{s}, \mathbf{t} \rangle \in \mathcal{T}} \log \sum_{\mathbf{a}} p_{\boldsymbol{\theta}}(\mathbf{t}, \mathbf{a} \mid \mathbf{s}, n) + \beta \sum_{k} |\theta_k| \quad .$$

Because of the ℓ_1 penalty, this objective is not everywhere differentiable, but the gradient with respect to the parameters of the log-likelihood term is as follows.

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \sum_{\langle \mathbf{s}, \mathbf{t} \rangle \in \mathcal{T}} \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{a} | \mathbf{s}, \mathbf{t}, n)}[\mathbf{H}(\cdot)] - \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{t}, \mathbf{a} | \mathbf{s}, n)}[\mathbf{H}(\cdot)] \tag{1}$$

To optimize $\mathcal{L}$, we employ an online method that approximates ℓ_1 regularization and only depends on the gradient of the unregularized objective (Tsuruoka et al., 2009). This method is quite attractive since it is only necessary to represent the active features, meaning impractically large feature spaces can be searched provided the regularization strength is sufficiently high. Additionally, not only has this technique been shown to be very effective for optimizing convex objectives, but evidence suggests that the stochasticity of online algorithms often results in better solutions than batch optimizers for non-convex objectives (Liang and Klein, 2009). On account of the latent alignment variable in our model, $\mathcal{L}$ is non-convex (as is the likelihood objective of the generative variant).

To choose the regularization strength β and the initial learning rate η_0,[3] we trained several models on a 10,000-sentence-pair subset of the French-English Hansards, and chose values that minimized the alignment error rate, as evaluated on a 447 sentence set of manually created alignments (Mihalcea and Pedersen, 2003). For the remainder of the experiments, we use the values we obtained, $\beta = 0.4$ and $\eta_0 = 0.3$.

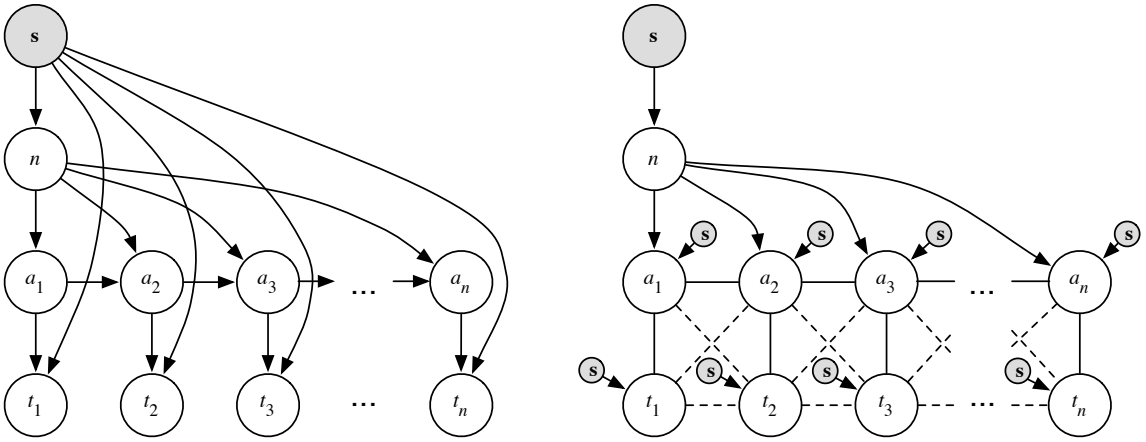
3.2 Inference with WFSAs

We now describe how to use weighted finite-state automata (WFSAs) to compute the quantities necessary for training. We begin by describing the ideal WFSA representing the full translation search space, which we call the *discriminative neighborhood*, and then discuss strategies for reducing its size in the next section, since the full model is prohibitively large, even with small data sets.

For each training instance $\langle \mathbf{s}, \mathbf{t} \rangle$, the contribution to the gradient (Equation 1) is the difference in two vectors of expectations. The first term is the expected value of $\mathbf{H}(\cdot)$ when observing $\langle \mathbf{s}, n, \mathbf{t} \rangle$ and letting $\mathbf{a}$ range over all possible alignments. The second is the expectation of the same function, but observing only $\langle \mathbf{s}, n \rangle$ and letting $\mathbf{t}'$ and $\mathbf{a}$ take on any possible values (i.e., all possible translations of length n and all their possible alignments to $\mathbf{s}$). To compute these expectations, we can construct a WFSA representing the discriminative neighborhood, the set $\Omega^n \times [0, m]^n$, such that every path from the start state to goal yields a pair $\langle \mathbf{t}', \mathbf{a} \rangle$ with weight

[2] One way to understand expressiveness is in terms of independence assumptions, of course. Research in graphical models has done much to relate independence assumptions to the complexity of inference algorithms (Koller and Friedman, 2009).

[3] For the other free parameters of the algorithm, we use the default values recommended by Tsuruoka et al. (2009).



Fully directed model (Brown et al., 1993; Vogel et al., 1996; Berg-Kirkpatrick et al., 2010)

Our model

Figure 1: A graphical representation of a conventional generative lexical translation model (left) and our model with an undirected translation model. For clarity, the observed node **s** (representing the full source sentence) is drawn in multiple locations. The dashed lines indicate a dependency on a deterministic mapping of t_j (not its complete value).

$\mathbf{H}(\mathbf{t}', \mathbf{a}, \mathbf{s}, n)$. With our feature set (§4), number of states in this WFSA is $O(m \times n)$ since at each target index j, there is a different state for each possible index of the source word translated at position $j - 1$.[4]

Once the WFSA representing the discriminative neighborhood is built, we use the forward-backward algorithm to compute the *second* expectation term. We then intersect the WFSA with an unweighted FSA representing the target sentence **t** (because of the restricted structure of our WFSA, this amounts to removing edges), and finally run the forward-backward algorithm on the resulting WFSA to compute the first expectation.

3.3 Shrinking the Discriminative Neighborhood

The WFSA we constructed requires $m \times |\Omega|$ transitions between all adjacent states, which is impractically large. We can reduce the number of edges by restricting the set of words that each source word can translate into. Thus, the model will not discriminate among all candidate target strings in Ω^n, but rather in $\Omega^n_{\mathbf{s}}$, where $\Omega_{\mathbf{s}} = \bigcup_{i=1}^{m} \Omega_{s_i}$, and where Ω_s is the set of target words that s may translate into.[5]

We consider four different definitions of Ω_s: (1) the baseline of the full target vocabulary, (2) the set of all target words that co-occur in sentence pairs containing s, (3) the most probable words under IBM Model 1 that are above a threshold, and (4) the same Model 1, except we add a sparse symmetric Dirichlet prior ($\alpha = 0.01$) on the translation distributions and use the empirical Bayes (EB) method to infer a point estimate, using variational inference.

Table 1: Comparison of alternative definitions Ω_s (arrows indicate whether higher or lower is better).

Ω_s	time (s) ↓	$\sum_s \lvert\Omega_s\rvert$ ↓	AER ↓
$= \Omega$	22.4	86.0M	**0.0**
co-occ.	8.9	0.68M	**0.0**
Model 1	**0.2**	0.38M	6.2
EB-Model 1	1.0	**0.15M**	2.9

Table 1 compares the average per-sentence time required to run the inference algorithm described

[4] States contain a bit more information than the index of the previous source word, for example, there is some additional information about the previous translation decision that is passed forward. However, the concept of splitting states to guarantee distinct paths for different values of non-local features is well understood by NLP and machine translation researchers, and the necessary state structure should be obvious from the feature description.

[5] Future work will explore alternative formulations of the discriminative neighborhood with the goal of further improving inference efficiency. Smith and Eisner (2005) show that good performance on unsupervised syntax learning is possible even when learning from very small discriminative neighborhoods, and we posit that the same holds here.

above under these four different definitions of Ω_s on a 10,000 sentence subset of the Hansards French-English corpus that includes manual word alignments. While our constructions guarantee that all references are reachable even in the reduced neighborhoods, not all *alignments* between source and target are possible. The last column is the *oracle* AER. Although EB variant of Model 1 neighborhood is slightly more expensive to do inference with than regular Model 1, we use it because it has a lower oracle AER.[6]

During alignment prediction (rather than during training) for a sentence pair $\langle \mathbf{s}, \mathbf{t} \rangle$, it is possible to further restrict $\Omega_\mathbf{s}$ to be just the set of words occurring in $\mathbf{t}$, making extremely fast inference possible (comparable to that of the generative HMM alignment model).

4 Features

Feature engineering lets us encode knowledge about what aspects of a translation derivation are useful in predicting whether it is good or not. In this section we discuss the features we used in our model. Many of these were taken from the discriminative alignment modeling literature, but we also note that our features can be much more fine-grained than those used in supervised alignment modeling, since we learn our models from a large amount of parallel data, rather than a small number of manual alignments.

Word association features. Word association features are at the heart of all lexical translation models, whether generative or discriminative. In addition to fine-grained boolean indicator features $\langle s_{a_j}, t_j \rangle$ for pair types, we have several orthographic features: identity, prefix identity, and an orthographic similarity measure designed to be informative for predicting the translation of named entities in languages that use similar alphabets.[7] It has the property that source-target pairs of *long* words that are similar are given a higher score than word pairs that are *short* and similar (dissimilar pairs have a score near zero, regardless of length). We also include "global" association scores that are precomputed by looking at the full training data: Dice's coefficient (discretized), which we use to measure association strength between pairs of source and target word types across sentence pairs (Dice, 1945), IBM Model 1 forward and reverse probabilities, and the geometric mean of the Model 1 forward and reverse probabilities. Finally, we also cluster the source and target vocabularies (Och, 1999) and include class pair indicator features, which can learn generalizations that, e.g., "nouns tend to translate into nouns but not modal verbs."

Positional features. Following Blunsom and Cohn (2006), we include features indicating closeness to the alignment matrix diagonal, $h(a_j, j, m, n) = \left| \frac{a_j}{m} - \frac{j}{n} \right|$. We also conjoin this feature with the source word class type indicator to enable the model to learn that certain word types are more or less likely to favor a location on the diagonal (e.g. Urdu's sentence-final verbs).

Source features. Some words are functional elements that fulfill purely grammatical roles and should not be the "source" of a translation. For example, Romance languages require a preposition in the formation of what could be a noun-noun compound in English, thus, it may be useful to learn *not* to translate certain words (i.e. they should not participate in alignment links), or to have a bias to translate others. To capture this intuition we include an indicator feature that fires each time a source vocabulary item (and source word class) participates in an alignment link.

Source path features. One class of particularly useful features assesses the goodness of the alignment 'path' through the source sentence (Vogel et al., 1996). Although assessing the predicted path requires using nonlocal features, since each $a_j \in [0, m]$ and m is relatively small, features can be sensitive to a wider context than is often practical.

We use many overlapping source path features, some of which are sensitive to the distance and direction of the jump between a_{j-1} and a_j, and others which are sensitive to the word pair these two points define, and others that combine all three elements. The features we use include a discretized

[6] We included all translations whose probability was within a factor of 10^{-4} of the highest probability translation.

[7] In experiments with Urdu, which uses an Arabic-derived script, the orthographic feature was computed after first applying a heuristic Romanization, which made the orthographic forms somewhat comparable.

jump distance, the discretized jump conjoined with an indicator feature for the target length n, the discretized jump feature conjoined with the class of s_{a_j}, and the discretized jump feature conjoined with the class of s_{a_j} and $s_{a_{j-1}}$. To discretize the features we take a log transform (base 1.3) of the jump width and let an indicator feature fire for the closest integer. In addition to these distance-dependent features, we also include indicator features that fire on bigrams $\langle s_{a_{j-1}}, s_{a_j} \rangle$ and their word classes. Thus, this feature can capture our intuition that, e.g., adjectives are more likely to come before or after a noun in different languages.

Target string features. Features sensitive to multiple values in the predicted target string or latent alignment variable must be handled carefully for the sake of computational tractability. While features that look at multiple source words can be computed linearly in the number of source words considered (since the source string is always observable), features that look at multiple target words require exponential time and space![8] However, by grouping the t_j's into coarse equivalence classes and looking at small numbers of variables, it is possible to incorporate such features. We include a feature that fires when a word translates as itself (for example, a name or a date, which occurs in languages that share the same alphabet) in position j, but then is translated again (as something else) in position $j-1$ or $j+1$.

5 Experiments

We now turn to an empirical assessment of our model. Using various datasets, we evaluate the performance of the models' intrinsic quality and theirtheir alignments' contribution to a standard machine translation system. We make use of parallel corpora from languages with very different typologies: a small (0.8M words) Chinese-English corpus from the tourism and travel domain (Takezawa et al., 2002), a corpus of Czech-English news commentary (3.1M words),[9] and an Urdu-English corpus (2M words) provided by NIST for the 2009 Open MT Evaluation. These pairs were selected since each poses different alignment challenges (word order in Chinese and Urdu, morphological complexity in Czech, and a non-alphabetic writing system in Chinese), and confining ourselves to these relatively small corpora reduced the engineering overhead of getting an implementation up and running. Future work will explore the scalability characteristics and limits of the model.

5.1 Methodology

For each language pair, we train two log-linear translation models as described above (§3), once with English as the source and once with English as the target language. For a baseline, we use the Giza++ toolkit (Och and Ney, 2003) to learn Model 4, again in both directions. We symmetrize the alignments from both model types using the `grow-diag-final-and` heuristic (Koehn et al., 2003) producing, in total, six alignment sets. We evaluate them both intrinsically and in terms of their performance in a translation system.

Since we only have gold alignments for Czech-English (Bojar and Prokopová, 2006), we can report alignment error rate (AER; Och and Ney, 2003) only for this pair. However, we offer two further measures that we believe are suggestive and that do not require gold alignments. One is the average alignment "fertility" of source words that occur only a single time in the training data (so-called *hapax legomena*). This assesses the impact of a typical alignment problem observed in generative models trained to maximize likelihood: infrequent source words act as "garbage collectors", with many target words aligned to them (the word *dislike* in the Model 4 alignment in Figure 2 is an example). Thus, we expect lower values of this measure to correlate with better alignments. The second measure is the number of rule types learned in the grammar induction process used for translation that match the translation test sets.[10] While neither a decrease in the average singleton fertility nor an increase in the number of rules induced guarantees better alignment quality, we believe it is reasonable to assume that they are positively correlated.

For the translation experiments in each language pair, we make use of the `cdec` decoder (Dyer et al.,

[8] This is of course what makes history-based language model integration an inference challenge in translation.

[9] `http://statmt.org/wmt10`

[10] This measure does not assess whether the rule types are good or bad, but it does suggest that the system's coverage is greater.

2010), inducing a hierarchical phrase based translation grammar from two sets of symmetrized alignments using the method described by Chiang (2007). Additionally, recent work that has demonstrated that extracting rules from n-best alignments has value (Liu et al., 2009; Venugopal et al., 2008). We therefore define a third condition where rules are extracted from the corpus under *both* the Model 4 and discriminative alignments and merged to form a single grammar. We incorporate a 3-gram language model learned from the target side of the training data as well as 50M supplemental words of monolingual training data consisting of sentences randomly sampled from the English Gigaword, version 4. In the small Chinese-English travel domain experiment, we just use the LM estimated from the bitext. The parameters of the translation model were tuned using "hypergraph" minimum error rate training (MERT) to maximize BLEU on a held-out development set (Kumar et al., 2009). Results are reported using case-insensitive BLEU (Papineni et al., 2002), METEOR[11] (Lavie and Denkowski, 2009), and TER (Snover et al., 2006), with the number of references varying by task. Since MERT is a non-deterministic optimization algorithm and results can vary considerably between runs, we follow Clark et al. (2011) and report the *average* score and standard deviation of 5 independent runs, 30 in the case of Chinese-English, since observed variance was higher.

5.2 Experimental Results

Czech-English. Czech-English poses problems for word alignment models since, unlike English, Czech words have a complex inflectional morphology, and the syntax permits relatively free word order. For this language pair, we evaluate alignment error rate using the manual alignment corpus described by Bojar and Prokopová (2006). Table 2 summarizes the results.

Chinese-English. Chinese-English poses a different set of problems for alignment. While Chinese words have rather simple morphology, the Chinese writing system renders our orthographic features useless. Despite these challenges, the Chinese results in Table 3 show the same pattern of results as seen in Czech-English.

Table 2: Czech-English experimental results. $\tilde{\phi}_{sing.}$ is the average fertility of singleton source words.

		AER ↓	$\tilde{\phi}_{sing.}$ ↓	# rules ↑
Model 4	**e** | **f**	24.8	4.1	
	f | **e**	33.6	6.6	
	sym.	23.4	2.7	993,953
Our model	**e** | **f**	21.9	2.3	
	f | **e**	29.3	3.8	
	sym.	**20.5**	**1.6**	**1,146,677**

Alignment	BLEU ↑	METEOR ↑	TER ↓
Model 4	16.3±0.2	46.1±0.1	67.4±0.3
Our model	16.5±0.1	46.8±0.1	67.0±0.2
Both	**17.4**±0.1	**47.7**±0.1	**66.3**±0.5

Table 3: Chinese-English experimental results.

		$\tilde{\phi}_{sing.}$ ↓	# rules ↑
Model 4	**e** | **f**	4.4	
	f | **e**	3.9	
	sym.	3.6	52,323
Our model	**e** | **f**	3.5	
	f | **e**	2.6	
	sym.	**3.1**	**54,077**

Alignment	BLEU ↑	METEOR ↑	TER ↓
Model 4	56.5±0.3	73.0±0.4	29.1±0.3
Our model	57.2±0.8	73.8±0.4	29.3±1.1
Both	**59.1**±0.6	**74.8**±0.7	**27.6**±0.5

Urdu-English. Urdu-English is a more challenging language pair for word alignment than the previous two we have considered. The parallel data is drawn from numerous genres, and much of it was acquired automatically, making it quite noisy. So our models must not only predict good translations, they must cope with bad ones as well. Second, there has been no previous work on discriminative modeling of Urdu, since, to our knowledge, no manual alignments have been created. Finally, unlike English, Urdu is a head-final language: not only does it have SOV word order, but rather than prepositions, it has post-positions, which follow the nouns they modify, meaning its large scale word order is substantially

[11] Meteor 1.0 with exact, stem, synonymy, and paraphrase modules and HTER parameters.

different from that of English. Table 4 demonstrates the same pattern of improving results with our alignment model.

Table 4: Urdu-English experimental results.

		$\tilde{\phi}_{sing.} \downarrow$	# rules ↑
Model 4	**e** | **f**	6.5	
	f | **e**	8.0	
	sym.	3.2	244,570
Our model	**e** | **f**	4.8	
	f | **e**	8.3	
	sym.	**2.3**	**260,953**

Alignment	BLEU ↑	METEOR ↑	TER ↓
Model 4	23.3±0.2	49.3±0.2	68.8±0.8
Our model	23.4±0.2	49.7±0.1	67.7±0.2
Both	**24.1**±0.2	**50.6**±0.1	**66.8**±0.5

5.3 Analysis

The quantitative results presented in this section strongly suggest that our modeling approach produces better alignments. In this section, we try to characterize how the model is doing what it does and what it has learned. Because of the ℓ_1 regularization, the number of active (non-zero) features in the inferred models is small, relative to the number of features considered during training. The number of active features ranged from about 300k for the small Chinese-English corpus to 800k for Urdu-English, which is less than one tenth of the available features in both cases. In all models, the coarse features (Model 1 probabilities, Dice coefficient, coarse positional features, etc.) typically received weights with large magnitudes, but finer features also played an important role.

Language pair differences manifested themselves in many ways in the models that were learned. For example, orthographic features were (unsurprisingly) more valuable in Czech-English, with their largely overlapping alphabets, than in Chinese or Urdu. Examining the more fine-grained features is also illuminating. Table 5 shows the most highly weighted source path bigram features on the three models where English was the source language, and in each, we may observe some interesting characteristics of the target language. Left-most is English-Czech. At first it may be surprising that words like *since* and *that* have a highly weighted feature for transitioning to themselves. However, Czech punctuation rules require that relative clauses and subordinating conjunctions be preceded by a comma (which is only optional or outright forbidden in English), therefore our model translates these words twice, once to produce the comma, and a second time to produce the lexical item. The middle column is the English-Chinese model. In the training data, many of the sentences are questions directed to a second person, *you*. However, Chinese questions do not invert and the subject remains in the canonical first position, thus the transition from the start of sentence to *you* is highly weighted. Finally, Figure 2 illustrates how Model 4 (left) and our discriminative model (right) align an English-Urdu sentence pair (the English side is being conditioned on in both models). A reflex of Urdu's head-final word order is seen in the list of most highly weighted bigrams, where a path through the English source where verbs that transition to end-of-sentence periods are predictive of good translations into Urdu.

Table 5: The most highly weighted source path bigram features in the English-Czech, -Chinese, and -Urdu models.

Bigram	θ_k	Bigram	θ_k	Bigram	θ_k
._⟨/s⟩	3.08	._⟨/s⟩	2.67	._⟨/s⟩	1.87
like_like	1.19	?_?	2.25	⟨s⟩_this	1.24
one_of	1.06	⟨s⟩_please	2.01	**will_.**	**1.17**
"_.	0.95	much_?	1.61	**are_.**	**1.16**
that_that	**0.92**	⟨s⟩_if	1.58	**is_.**	**1.09**
is_but	0.92	thank_you	1.47	is_that	1.00
since_since	**0.84**	⟨s⟩_sorry	1.46	**have_.**	**0.97**
⟨s⟩_when	0.83	**⟨s⟩_you**	**1.45**	**has_.**	**0.96**
,_how	0.83	please_like	1.24	**was_.**	**0.91**
,_not	0.83	⟨s⟩_this	1.19	**will_⟨/s⟩**	**0.88**

6 Related Work

The literature contains numerous descriptions of discriminative approaches to word alignment motivated by the desire to be able to incorporate multiple, overlapping knowledge sources (Ayan et al., 2005; Moore, 2005; Taskar et al., 2005; Blunsom and Cohn, 2006; Haghighi et al., 2009; Liu et al., 2010; DeNero and Klein, 2010; Setiawan et al., 2010). This body of work has been an invaluable source of useful features. Several authors have dealt with the problem training log-linear models in an unsu-

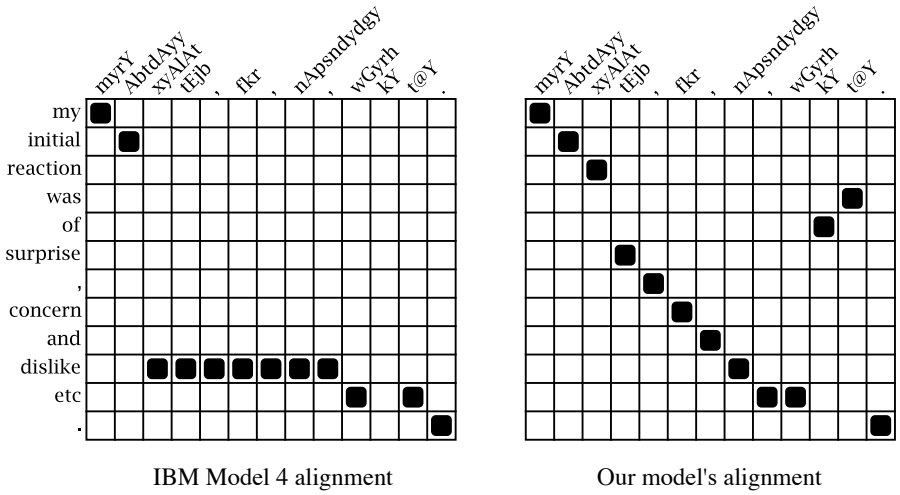


Figure 2: Example English-Urdu alignment under IBM Model 4 (left) and our discriminative model (right). Model 4 displays two characteristic errors: garbage collection and an overly-strong monotonicity bias. Whereas our model does not exhibit these problems, and in fact, makes no mistakes in the alignment.

pervised setting. The contrastive estimation technique proposed by Smith and Eisner (2005) is globally normalized (and thus capable of dealing with arbitrary features), and closely related to the model we developed; however, they do not discuss the problem of word alignment. Berg-Kirkpatrick et al. (2010) learn locally normalized log-linear models in a generative setting. Globally normalized discriminative models with latent variables (Quattoni et al., 2004) have been used for a number of language processing problems, including MT (Dyer and Resnik, 2010; Blunsom et al., 2008a). However, this previous work relied on translation grammars constructed using standard generative word alignment processes.

7 Future Work

While we have demonstrated that this model can be substantially useful, it is limited in some important ways which are being addressed in ongoing work. First, training is expensive, and we are exploring alternatives to the conditional likelihood objective that is currently used, such as contrastive neighborhoods advocated by (Smith and Eisner, 2005). Additionally, there is much evidence that non-local features like the source word fertility are (*cf.* IBM Model 3) useful for translation and alignment modeling. To be truly general, it must be possible to utilize such features. Unfortunately, features like this that depend on global properties of the alignment vector, **a**, make the inference problem NP-hard, and approximations are necessary. Fortunately, there is much recent work on approximate inference techniques for incorporating nonlocal features (Blunsom et al., 2008b; Gimpel and Smith, 2009; Cromières and Kurohashi, 2009; Weiss and Taskar, 2010), suggesting that this problem too can be solved using established techniques.

8 Conclusion

We have introduced a globally normalized, log-linear lexical translation model that can be trained discriminatively using only parallel sentences, which we apply to the problem of word alignment. Our approach addresses two important shortcomings of previous work: (1) that local normalization of generative models constrains the features that can be used, and (2) that previous discriminatively trained word alignment models required supervised alignments. According to a variety of measures in a variety of translation tasks, this model produces superior alignments to generative approaches. Furthermore, the features learned by our model reveal interesting characteristics of the language pairs being modeled.

Acknowledgments

This work was supported in part by the DARPA GALE program; the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant num-

ber W911NF-10-1-0533; and the National Science Foundation through grants IIS-0844507, IIS-0915187, IIS-0713402, and IIS-0915327 and through TeraGrid resources provided by the Pittsburgh Supercomputing Center under grant number TG-DBS110003. We thank Ondřej Bojar for providing the Czech-English alignment data, and three anonymous reviewers for their detailed suggestions and comments on an earlier draft of this paper.

References

N. F. Ayan, B. J. Dorr, and C. Monz. 2005. NeurAlign: combining word alignments using neural networks. In *Proc. of HLT-EMNLP*.

T. Berg-Kirkpatrick, A. Bouchard-Côté, J. DeNero, and D. Klein. 2010. Painless unsupervised learning with features. In *Proc. of NAACL*.

P. Blunsom and T. Cohn. 2006. Discriminative word alignment with conditional random fields. In *Proc. of ACL*.

P. Blunsom, T. Cohn, and M. Osborne. 2008a. A discriminative latent variable model for statistical machine translation. In *Proc. of ACL-HLT*.

P. Blunsom, T. Cohn, and M. Osborne. 2008b. Probabilistic inference for machine translation. In *Proc. of EMNLP 2008*.

O. Bojar and M. Prokopová. 2006. Czech-English word alignment. In *Proc. of LREC*.

P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

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

J. Clark, C. Dyer, A. Lavie, and N. A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proc. of ACL*.

F. Cromières and S. Kurohashi. 2009. An alignment algorithm using belief propagation and a structure-based distortion model. In *Proc. of EACL*.

J. DeNero and D. Klein. 2010. Discriminative modeling of extraction sets for machine translation. In *Proc. of ACL*.

L. R. Dice. 1945. Measures of the amount of ecologic association between species. *Journal of Ecology*, 26:297–302.

C. Dyer and P. Resnik. 2010. Context-free reordering, finite-state translation. In *Proc. of NAACL*.

C. Dyer, A. Lopez, J. Ganitkevitch, J. Weese, F. Ture, P. Blunsom, H. Setiawan, V. Eidelman, and P. Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proc. of ACL (demonstration session)*.

A. Fraser. 2007. *Improved Word Alignments for Statistical Machine Translation*. Ph.D. thesis, University of Southern California.

K. Gimpel and N. A. Smith. 2009. Cube summing, approximate inference with non-local features, and dynamic programming without semirings. In *Proc. of EACL*.

N. Habash and F. Sadat. 2006. Arabic preprocessing schemes for statistical machine translation. In *Proc. of NAACL*, New York.

A. Haghighi, J. Blitzer, J. DeNero, and D. Klein. 2009. Better word alignments with supervised ITG models. In *Proc. of ACL-IJCNLP*.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. of NAACL*.

D. Koller and N. Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

S. Kumar, W. Macherey, C. Dyer, and F. Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proc. of ACL-IJCNLP*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

A. Lavie and M. Denkowski. 2009. The METEOR metric for automatic evaluation of machine translation. *Machine Translation Journal*, 23(2–3):105–115.

P. Liang and D. Klein. 2009. Online EM for unsupervised models. In *Proc. of NAACL*.

Y. Liu, T. Xia, X. Xiao, and Q. Liu. 2009. Weighted alignment matrices for statistical machine translation. In *Proc. of EMNLP*.

Y. Liu, Q. Liu, and S. Lin. 2010. Discriminative word alignment by linear modeling. *Computational Linguistics*, 36(3):303–339.

A. Lopez. 2008. Tera-scale translation models via pattern matching. In *Proc. of COLING*.

R. Mihalcea and T. Pedersen. 2003. An evaluation exercise for word alignment. In *Proc. of the Workshop on Building and Using Parallel Texts*.

R. C. Moore. 2005. A discriminative framework for bilingual word alignment. In *Proc. of HLT-EMNLP*.

F. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

F. Och. 1999. An efficient method for determining bilingual word classes. In *Proc. of EACL*.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

A. Quattoni, M. Collins, and T. Darrell. 2004. Conditional random fields for object recognition. In *NIPS 17*.

H. Setiawan, C. Dyer, and P. Resnik. 2010. Discriminative word alignment with a function word reordering model. In *Proc. of EMNLP*.

N. A. Smith and J. Eisner. 2005. Contrastive estimation: training log-linear models on unlabeled data. In *Proc. of ACL*.

M. Snover, B. J. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proc. of AMTA*.

T. Takezawa, E. Sumita, F. Sugaya, H. Yamamoto, and S. Yamamoto. 2002. Toward a broad-coverage bilingual corpus for speech translation of travel conversations in the real world. In *Proc. of LREC*.

B. Taskar, S. Lacoste-Julien, and D. Klein. 2005. A discriminative matching approach to word alignment. In *Proc. of HLT-EMNLP*.

Y. Tsuruoka, J. Tsujii, and S. Ananiadou. 2009. Stochastic gradient descent training for l_1-regularized log-linear models with cumulative penalty. In *Proc. of ACL-IJCNLP*.

A. Venugopal, A. Zollmann, N. A. Smith, and S. Vogel. 2008. Wider pipelines: n-best alignments and parses in MT training. In *Proc. of AMTA*.

S. Vogel, H. Ney, and C. Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proc. of COLING*.

D. Weiss and B. Taskar. 2010. Structured prediction cascades. In *Proc. of AISTATS*.

Model-Based Aligner Combination Using Dual Decomposition

John DeNero
Google Research
denero@google.com

Klaus Macherey
Google Research
kmach@google.com

Abstract

Unsupervised word alignment is most often modeled as a Markov process that generates a sentence $\boldsymbol{f}$ conditioned on its translation $\boldsymbol{e}$. A similar model generating $\boldsymbol{e}$ from $\boldsymbol{f}$ will make different alignment predictions. Statistical machine translation systems combine the predictions of two directional models, typically using heuristic combination procedures like *grow-diag-final*. This paper presents a graphical model that embeds two directional aligners into a single model. Inference can be performed via dual decomposition, which reuses the efficient inference algorithms of the directional models. Our bidirectional model enforces a one-to-one phrase constraint while accounting for the uncertainty in the underlying directional models. The resulting alignments improve upon baseline combination heuristics in word-level and phrase-level evaluations.

1 Introduction

Word alignment is the task of identifying corresponding words in sentence pairs. The standard approach to word alignment employs directional Markov models that align the words of a sentence $\boldsymbol{f}$ to those of its translation $\boldsymbol{e}$, such as IBM Model 4 (Brown et al., 1993) or the HMM-based alignment model (Vogel et al., 1996).

Machine translation systems typically combine the predictions of two directional models, one which aligns $\boldsymbol{f}$ to $\boldsymbol{e}$ and the other $\boldsymbol{e}$ to $\boldsymbol{f}$ (Och et al., 1999). Combination can reduce errors and relax the one-to-many structural restriction of directional models. Common combination methods include the union or intersection of directional alignments, as well as heuristic interpolations between the union and intersection like *grow-diag-final* (Koehn et al., 2003). This paper presents a model-based alternative to aligner combination. Inference in a probabilistic model resolves the conflicting predictions of two directional models, while taking into account each model's uncertainty over its output.

This result is achieved by embedding two directional HMM-based alignment models into a larger bidirectional graphical model. The full model structure and potentials allow the two embedded directional models to disagree to some extent, but reward agreement. Moreover, the bidirectional model enforces a one-to-one phrase alignment structure, similar to the output of phrase alignment models (Marcu and Wong, 2002; DeNero et al., 2008), unsupervised inversion transduction grammar (ITG) models (Blunsom et al., 2009), and supervised ITG models (Haghighi et al., 2009; DeNero and Klein, 2010).

Inference in our combined model is not tractable because of numerous edge cycles in the model graph. However, we can employ dual decomposition as an approximate inference technique (Rush et al., 2010). In this approach, we iteratively apply the same efficient sequence algorithms for the underlying directional models, and thereby optimize a dual bound on the model objective. In cases where our algorithm converges, we have a certificate of optimality under the full model. Early stopping before convergence still yields useful outputs.

Our model-based approach to aligner combination yields improvements in alignment quality and phrase extraction quality in Chinese-English experiments, relative to typical heuristic combinations methods applied to the predictions of independent directional models.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 420–429,
Portland, Oregon, June 19-24, 2011.

2 Model Definition

Our bidirectional model $\mathcal{G} = (\mathcal{V}, \mathcal{D})$ is a globally normalized, undirected graphical model of the word alignment for a fixed sentence pair $(\boldsymbol{e}, \boldsymbol{f})$. Each vertex in the vertex set $\mathcal{V}$ corresponds to a model variable V_i, and each undirected edge in the edge set $\mathcal{D}$ corresponds to a pair of variables (V_i, V_j). Each vertex has an associated potential function $\omega_i(v_i)$ that assigns a real-valued potential to each possible value v_i of V_i.[1] Likewise, each edge has an associated potential function $\mu_{ij}(v_i, v_j)$ that scores pairs of values. The probability under the model of any full assignment $\boldsymbol{v}$ to the model variables, indexed by $\mathcal{V}$, factors over vertex and edge potentials.

$$\mathrm{P}(\boldsymbol{v}) \propto \prod_{v_i \in \mathcal{V}} \omega_i(v_i) \cdot \prod_{(v_i, v_j) \in \mathcal{D}} \mu_{ij}(v_i, v_j)$$

Our model contains two directional hidden Markov alignment models, which we review in Section 2.1, along with additional structure that that we introduce in Section 2.2.

2.1 HMM-Based Alignment Model

This section describes the classic hidden Markov model (HMM) based alignment model (Vogel et al., 1996). The model generates a sequence of words $\boldsymbol{f}$ conditioned on a word sequence $\boldsymbol{e}$. We conventionally index the words of $\boldsymbol{e}$ by i and $\boldsymbol{f}$ by j. $\mathrm{P}(\boldsymbol{f}|\boldsymbol{e})$ is defined in terms of a latent alignment vector $\mathbf{a}$, where $a_j = i$ indicates that word position i of $\boldsymbol{e}$ aligns to word position j of $\boldsymbol{f}$.

$$\mathrm{P}(\boldsymbol{f}|\boldsymbol{e}) = \sum_{\mathbf{a}} \mathrm{P}(\boldsymbol{f}, \mathbf{a}|\boldsymbol{e})$$

$$\mathrm{P}(\boldsymbol{f}, \mathbf{a}|\boldsymbol{e}) = \prod_{j=1}^{|\boldsymbol{f}|} \mathrm{D}(a_j|a_{j-1}) \mathrm{M}(f_j|e_{a_j}) \; . \quad (1)$$

In Equation 1 above, the emission model M is a learned multinomial distribution over word types. The transition model D is a multinomial over transition distances, which treats null alignments as a special case.

$$\mathrm{D}(a_j = 0|a_{j-1} = i) = \mathrm{p}_o$$
$$\mathrm{D}(a_j = i' \neq 0|a_{j-1} = i) = (1 - \mathrm{p}_o) \cdot c(i' - i) \; ,$$

where $c(i' - i)$ is a learned distribution over signed distances, normalized over the possible transitions from i. The parameters of the conditional multinomial M and the transition model c can be learned from a sentence aligned corpus via the expectation maximization algorithm. The null parameter p_o is typically fixed.[2]

The highest probability word alignment vector under the model for a given sentence pair $(\boldsymbol{e}, \boldsymbol{f})$ can be computed exactly using the standard Viterbi algorithm for HMMs in $O(|\boldsymbol{e}|^2 \cdot |\boldsymbol{f}|)$ time.

An alignment vector $\mathbf{a}$ can be converted trivially into a set of word alignment links $\mathcal{A}$:

$$\mathcal{A}_{\mathbf{a}} = \{(i, j) : a_j = i, i \neq 0\} \; .$$

$\mathcal{A}_{\mathbf{a}}$ is constrained to be many-to-one from $\boldsymbol{f}$ to $\boldsymbol{e}$; many positions j can align to the same i, but each j appears at most once.

We have defined a directional model that generates $\boldsymbol{f}$ from $\boldsymbol{e}$. An identically structured model can be defined that generates $\boldsymbol{e}$ from $\boldsymbol{f}$. Let $\mathbf{b}$ be a vector of alignments where $b_i = j$ indicates that word position j of $\boldsymbol{f}$ aligns to word position i of $\boldsymbol{e}$. Then, $\mathrm{P}(\boldsymbol{e}, \mathbf{b}|\boldsymbol{f})$ is defined similarly to Equation 1, but with $\boldsymbol{e}$ and $\boldsymbol{f}$ swapped. We can distinguish the transition and emission distributions of the two models by subscripting them with their generative direction.

$$\mathrm{P}(\boldsymbol{e}, \mathbf{b}|\boldsymbol{f}) = \prod_{j=1}^{|\boldsymbol{e}|} \mathrm{D}_{\boldsymbol{f} \to \boldsymbol{e}}(b_i|b_{i-1}) \mathrm{M}_{\boldsymbol{f} \to \boldsymbol{e}}(e_i|f_{b_i}) \; .$$

The vector $\mathbf{b}$ can be interpreted as a set of alignment links that is one-to-many: each value i appears at most once in the set.

$$\mathcal{A}_{\mathbf{b}} = \{(i, j) : b_i = j, j \neq 0\} \; .$$

2.2 A Bidirectional Alignment Model

We can combine two HMM-based directional alignment models by embedding them in a larger model

[1] Potentials in an undirected model play the same role as conditional probabilities in a directed model, but do not need to be locally normalized.

[2] In experiments, we set $\mathrm{p}_o = 10^{-6}$. Transitions from a null-aligned state $a_{j-1} = 0$ are also drawn from a fixed distribution, where $\mathrm{D}(a_j = 0|a_{j-1} = 0) = 10^{-4}$ and for $i' \geq 1$,

$$\mathrm{D}(a_j = i'|a_{j-1} = 0) \propto 0.8^{\left(-\left|i' \cdot \frac{|\boldsymbol{f}|}{|\boldsymbol{e}|} - j\right|\right)} \; .$$

With small p_o, the shape of this distribution has little effect on the alignment outcome.

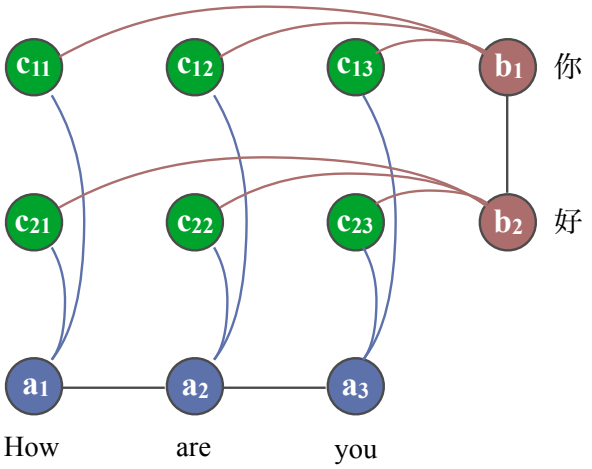


Figure 1: The structure of our graphical model for a simple sentence pair. The variables $\mathbf{a}$ are blue, $\mathbf{b}$ are red, and $\mathbf{c}$ are green.

that includes all of the random variables of two directional models, along with additional structure that promotes agreement and resolves discrepancies.

The original directional models include observed word sequences $\boldsymbol{e}$ and $\boldsymbol{f}$, along with the two latent alignment vectors $\mathbf{a}$ and $\mathbf{b}$ defined in Section 2.1. Because the word types and lengths of $\boldsymbol{e}$ and $\boldsymbol{f}$ are always fixed by the observed sentence pair, we can define our model only over $\mathbf{a}$ and $\mathbf{b}$, where the edge potentials between any a_j, f_j, and $\boldsymbol{e}$ are compiled into a vertex potential function $\omega_j^{(\mathbf{a})}$ on a_j, defined in terms of $\boldsymbol{f}$ and $\boldsymbol{e}$, and likewise for any b_i.

$$\omega_j^{(\mathbf{a})}(i) = \mathrm{M}_{\boldsymbol{e}\to\boldsymbol{f}}(f_j|e_i)$$
$$\omega_i^{(\mathbf{b})}(j) = \mathrm{M}_{\boldsymbol{f}\to\boldsymbol{e}}(e_i|f_j)$$

The edge potentials between $\mathbf{a}$ and $\mathbf{b}$ encode the transition model in Equation 1.

$$\mu_{j-1,j}^{(\mathbf{a})}(i,i') = \mathrm{D}_{\boldsymbol{e}\to\boldsymbol{f}}(a_j = i'|a_{j-1} = i)$$
$$\mu_{i-1,i}^{(\mathbf{b})}(j,j') = \mathrm{D}_{\boldsymbol{f}\to\boldsymbol{e}}(b_i = j'|b_{i-1} = j)$$

In addition, we include in our model a latent boolean matrix $\mathbf{c}$ that encodes the output of the combined aligners:

$$\mathbf{c} \in \{0,1\}^{|\boldsymbol{e}|\times|\boldsymbol{f}|} .$$

This matrix encodes the alignment links proposed by the bidirectional model:

$$\mathcal{A}_{\mathbf{c}} = \{(i,j) : c_{ij} = 1\} .$$

Each model node for an element $c_{ij} \in \{0,1\}$ is connected to a_j and b_i via *coherence edges*. These edges allow the model to ensure that the three sets of variables, $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$, together encode a coherent alignment analysis of the sentence pair. Figure 1 depicts the graph structure of the model.

2.3 Coherence Potentials

The potentials on coherence edges are not learned and do not express any patterns in the data. Instead, they are fixed functions that promote consistency between the integer-valued directional alignment vectors $\mathbf{a}$ and $\mathbf{b}$ and the boolean-valued matrix $\mathbf{c}$.

Consider the assignment $a_j = i$, where $i = 0$ indicates that word f_j is null-aligned, and $i \geq 1$ indicates that f_j aligns to e_i. The coherence potential ensures the following relationship between the variable assignment $a_j = i$ and the variables $c_{i'j}$, for any $i' \in [1, |\boldsymbol{e}|]$.

- If $i = 0$ (null-aligned), then all $c_{i'j} = 0$.
- If $i > 0$, then $c_{ij} = 1$.
- $c_{i'j} = 1$ only if $i' \in \{i-1,\ i,\ i+1\}$.
- Assigning $c_{i'j} = 1$ for $i' \neq i$ incurs a cost $e^{-\alpha}$.

Collectively, the list of cases above enforce an intuitive correspondence: an alignment $a_j = i$ ensures that c_{ij} must be 1, adjacent neighbors may be 1 but incur a cost, and all other elements are 0.

This pattern of effects can be encoded in a potential function $\mu^{(\mathbf{c})}$ for each coherence edge. These edge potential functions takes an integer value i for some variable a_j and a binary value k for some $c_{i'j}$.

$$\mu_{(a_j,c_{i'j})}^{(\mathbf{c})}(i,k) = \begin{cases} 1 & i = 0 \wedge k = 0 \\ 0 & i = 0 \wedge k = 1 \\ \hline 1 & i = i' \wedge k = 1 \\ 0 & i = i' \wedge k = 0 \\ \hline 1 & i \neq i' \wedge k = 0 \\ e^{-\alpha} & |i - i'| = 1 \wedge k = 1 \\ 0 & |i - i'| > 1 \wedge k = 1 \end{cases} \quad (2)$$

Above, potentials of 0 effectively disallow certain cases because a full assignment to $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ is scored by the product of all model potentials. The potential function $\mu_{(b_i,c_{ij'})}^{(\mathbf{c})}(j,k)$ for a coherence edge between $\mathbf{b}$ and $\mathbf{c}$ is defined similarly.

2.4 Model Properties

We interpret $\mathbf{c}$ as the final alignment produced by the model, ignoring $\mathbf{a}$ and $\mathbf{b}$. In this way, we relax the one-to-many constraints of the directional models. However, all of the information about how words align is expressed by the vertex and edge potentials on $\mathbf{a}$ and $\mathbf{b}$. The coherence edges and the link matrix $\mathbf{c}$ only serve to resolve conflicts between the directional models and communicate information between them.

Because directional alignments are preserved intact as components of our model, extensions or refinements to the underlying directional Markov alignment model could be integrated cleanly into our model as well, including lexicalized transition models (He, 2007), extended conditioning contexts (Brunning et al., 2009), and external information (Shindo et al., 2010).

For any assignment to $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ with non-zero probability, $\mathbf{c}$ must encode a one-to-one phrase alignment with a maximum phrase length of 3. That is, any word in either sentence can align to at most three words in the opposite sentence, and those words must be contiguous. This restriction is directly enforced by the edge potential in Equation 2.

3 Model Inference

In general, graphical models admit efficient, exact inference algorithms if they do not contain cycles. Unfortunately, our model contains numerous cycles. For every pair of indices (i, j) and (i', j'), the following cycle exists in the graph:

$$c_{ij} \rightarrow b_i \rightarrow c_{ij'} \rightarrow a_{j'} \rightarrow c_{i'j'} \rightarrow b_{i'} \rightarrow c_{i'j} \rightarrow a_j \rightarrow c_{ij}$$

Additional cycles also exist in the graph through the edges between a_{j-1} and a_j and between b_{i-1} and b_i. The general phrase alignment problem under an arbitrary model is known to be NP-hard (DeNero and Klein, 2008).

3.1 Dual Decomposition

While the entire graphical model has loops, there are two overlapping subgraphs that are cycle-free. One subgraph $\mathcal{G}_\mathbf{a}$ includes all of the vertices corresponding to variables $\mathbf{a}$ and $\mathbf{c}$. The other subgraph $\mathcal{G}_\mathbf{b}$ includes vertices for variables $\mathbf{b}$ and $\mathbf{c}$. Every edge in the graph belongs to exactly one of these two subgraphs.

The dual decomposition inference approach allows us to exploit this sub-graph structure (Rush et al., 2010). In particular, we can iteratively apply exact inference to the subgraph problems, adjusting their potentials to reflect the constraints of the full problem. The technique of dual decomposition has recently been shown to yield state-of-the-art performance in dependency parsing (Koo et al., 2010).

3.2 Dual Problem Formulation

To describe a dual decomposition inference procedure for our model, we first restate the inference problem under our graphical model in terms of the two overlapping subgraphs that admit tractable inference. Let $\mathbf{c}^{(\mathbf{a})}$ be a copy of $\mathbf{c}$ associated with $\mathcal{G}_\mathbf{a}$, and $\mathbf{c}^{(\mathbf{b})}$ with $\mathcal{G}_\mathbf{b}$. Also, let $f(\mathbf{a}, \mathbf{c}^{(\mathbf{a})})$ be the unnormalized log-probability of an assignment to $\mathcal{G}_\mathbf{a}$ and $g(\mathbf{b}, \mathbf{c}^{(\mathbf{b})})$ be the unnormalized log-probability of an assignment to $\mathcal{G}_\mathbf{b}$. Finally, let $\mathcal{I}$ be the index set of all (i, j) for $\mathbf{c}$. Then, the maximum likelihood assignment to our original model can be found by optimizing

$$\max_{\mathbf{a},\mathbf{b},\mathbf{c}^{(\mathbf{a})},\mathbf{c}^{(\mathbf{b})}} f(\mathbf{a}, \mathbf{c}^{(\mathbf{a})}) + g(\mathbf{b}, \mathbf{c}^{(\mathbf{b})}) \qquad (3)$$

$$\textit{such that: } c^{(\mathbf{a})}_{ij} = c^{(\mathbf{b})}_{ij} \;\; \forall\, (i,j) \in \mathcal{I}\,.$$

The Lagrangian relaxation of this optimization problem is $L(\mathbf{a}, \mathbf{b}, \mathbf{c}^{(\mathbf{a})}, \mathbf{c}^{(\mathbf{b})}, \mathbf{u}) =$

$$f(\mathbf{a}, \mathbf{c}^{(\mathbf{a})}) + g(\mathbf{b}, \mathbf{c}^{(\mathbf{b})}) + \sum_{(i,j)\in\mathcal{I}} u(i,j)(\mathbf{c}^{(\mathbf{a})}_{i,j} - \mathbf{c}^{(\mathbf{b})}_{i,j})\,.$$

Hence, we can rewrite the original problem as

$$\max_{\mathbf{a},\mathbf{b},\mathbf{c}^{(\mathbf{a})},\mathbf{c}^{(\mathbf{b})}} \min_{\mathbf{u}} L(\mathbf{a}, \mathbf{b}, \mathbf{c}^{(\mathbf{a})}, \mathbf{c}^{(\mathbf{b})}, \mathbf{u})\,.$$

We can form a dual problem that is an upper bound on the original optimization problem by swapping the order of $\min$ and $\max$. In this case, the dual problem decomposes into two terms that are each local to an acyclic subgraph.

$$\min_{\mathbf{u}} \left(\max_{\mathbf{a},\mathbf{c}^{(\mathbf{a})}} \left[f(\mathbf{a}, \mathbf{c}^{(\mathbf{a})}) + \sum_{i,j} u(i,j) c^{(\mathbf{a})}_{ij} \right] + \max_{\mathbf{b},\mathbf{c}^{(\mathbf{b})}} \left[g(\mathbf{b}, \mathbf{c}^{(\mathbf{b})}) - \sum_{i,j} u(i,j) c^{(\mathbf{b})}_{ij} \right] \right) \qquad (4)$$

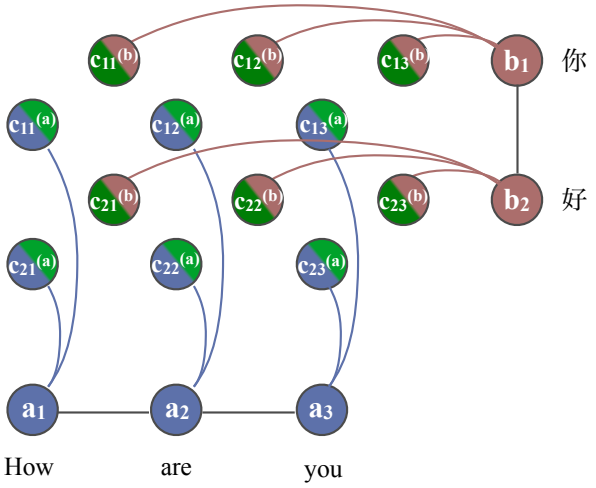


Figure 2: Our combined model decomposes into two acyclic models that each contain a copy of $\mathbf{c}$.

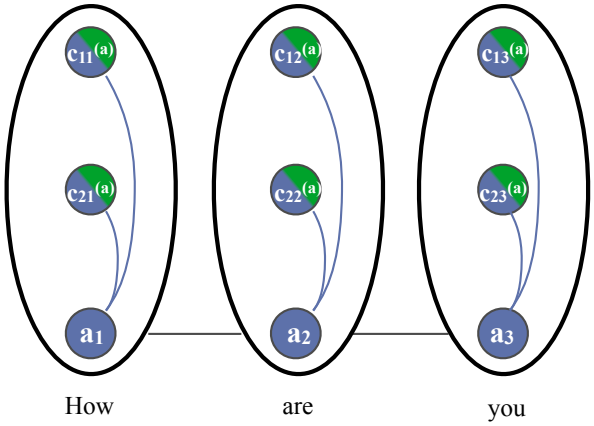


Figure 3: The tree-structured subgraph $\mathcal{G}_{\mathbf{a}}$ can be mapped to an equivalent chain-structured model by optimizing over $c_{i'j}$ for $a_j = i$.

The decomposed model is depicted in Figure 2. As in previous work, we solve for the dual variable $\mathbf{u}$ by repeatedly performing inference in the two decoupled maximization problems.

3.3 Sub-Graph Inference

We now address the problem of evaluating Equation 4 for fixed $\mathbf{u}$. Consider the first line of Equation 4, which includes variables $\mathbf{a}$ and $\mathbf{c}^{(\mathbf{a})}$.

$$\max_{\mathbf{a},\mathbf{c}^{(\mathbf{a})}} \left[f(\mathbf{a}, \mathbf{c}^{(\mathbf{a})}) + \sum_{i,j} u(i,j) c_{ij}^{(\mathbf{a})} \right] \quad (5)$$

Because the graph $\mathcal{G}_{\mathbf{a}}$ is tree-structured, Equation 5 can be evaluated in polynomial time. In fact, we can make a stronger claim: we can reuse the Viterbi inference algorithm for linear chain graphical models that applies to the embedded directional HMM models. That is, we can cast the optimization of Equation 5 as

$$\max_{\mathbf{a}} \left[\prod_{j=1}^{|\boldsymbol{f}|} \mathrm{D}_{\boldsymbol{e} \rightarrow \boldsymbol{f}}(a_j | a_{j-1}) \cdot \mathrm{M}'_j(a_j = i) \right] .$$

In the original HMM-based aligner, the vertex potentials correspond to bilexical probabilities. Those quantities appear in $f(\mathbf{a}, \mathbf{c}^{(\mathbf{a})})$, and therefore will be a part of $\mathrm{M}'_j(\cdot)$ above. The additional terms of Equation 5 can also be factored into the vertex potentials of this linear chain model, because the optimal choice of each c_{ij} can be determined from a_j and the model parameters. If $a_j = i$, then $c_{ij} = 1$ according to our edge potential defined in Equation 2. Hence, setting $a_j = i$ requires the inclusion of the corresponding vertex potential $\omega_j^{(\mathbf{a})}(i)$, as well as $u(i,j)$. For $i' \neq i$, either $c_{i'j} = 0$, which contributes nothing to Equation 5, or $c_{i'j} = 1$, which contributes $u(i',j) - \alpha$, according to our edge potential between a_j and $c_{i'j}$.

Thus, we can capture the net effect of assigning a_j and then optimally assigning all $c_{i'j}$ in a single potential $\mathrm{M}'_j(a_j = i) =$

$$\omega_j^{(\mathbf{a})}(i) + \exp \left[u(i,j) + \sum_{j' : |j'-j|=1} \max(0, u(i,j') - \alpha) \right]$$

Note that Equation 5 and f are sums of terms in log space, while Viterbi inference for linear chains assumes a product of terms in probability space, which introduces the exponentiation above.

Defining this potential allows us to collapse the source-side sub-graph inference problem defined by Equation 5, into a simple linear chain model that only includes potential functions M'_j and $\mu^{(\mathbf{a})}$. Hence, we can use a highly optimized linear chain inference implementation rather than a solver for general tree-structured graphical models. Figure 3 depicts this transformation.

An equivalent approach allows us to evaluate the

Algorithm 1 Dual decomposition inference algorithm for the bidirectional model

for $t = 1$ to max iterations **do**
  $r \leftarrow \frac{1}{t}$ ▷ *Learning rate*
  $\mathbf{c}^{(\mathbf{a})} \leftarrow \arg\max f(\mathbf{a}, \mathbf{c}^{(\mathbf{a})}) + \sum_{i,j} u(i,j) c_{ij}^{(\mathbf{a})}$
  $\mathbf{c}^{(\mathbf{b})} \leftarrow \arg\max g(\mathbf{b}, \mathbf{c}^{(\mathbf{b})}) - \sum_{i,j} u(i,j) c_{ij}^{(\mathbf{b})}$
  if $\mathbf{c}^{(\mathbf{a})} = \mathbf{c}^{(\mathbf{b})}$ **then**
    return $\mathbf{c}^{(\mathbf{a})}$ ▷ *Converged*
  $\mathbf{u} \leftarrow \mathbf{u} + r \cdot (\mathbf{c}^{(\mathbf{b})} - \mathbf{c}^{(\mathbf{a})})$ ▷ *Dual update*
return combine$(\mathbf{c}^{(\mathbf{a})}, \mathbf{c}^{(\mathbf{b})})$ ▷ *Stop early*

second line of Equation 4 for fixed $\mathbf{u}$:

$$\max_{\mathbf{b}, \mathbf{c}^{(\mathbf{b})}} \left[g(\mathbf{b}, \mathbf{c}^{(\mathbf{b})}) + \sum_{i,j} u(i,j) c_{ij}^{(\mathbf{b})} \right] . \quad (6)$$

3.4 Dual Decomposition Algorithm

Now that we have the means to efficiently evaluate Equation 4 for fixed $\mathbf{u}$, we can define the full dual decomposition algorithm for our model, which searches for a $\mathbf{u}$ that optimizes Equation 4. We can iteratively search for such a $\mathbf{u}$ via sub-gradient descent. We use a learning rate $\frac{1}{t}$ that decays with the number of iterations t. The full dual decomposition optimization procedure appears in Algorithm 1.

If Algorithm 1 converges, then we have found a $\mathbf{u}$ such that the value of $\mathbf{c}^{(\mathbf{a})}$ that optimizes Equation 5 is identical to the value of $\mathbf{c}^{(\mathbf{b})}$ that optimizes Equation 6. Hence, it is also a solution to our original optimization problem: Equation 3. Since the dual problem is an upper bound on the original problem, this solution must be optimal for Equation 3.

3.5 Convergence and Early Stopping

Our dual decomposition algorithm provides an inference method that is exact upon convergence.[3] When Algorithm 1 does not converge, the two alignments $\mathbf{c}^{(\mathbf{a})}$ and $\mathbf{c}^{(\mathbf{b})}$ can still be used. While these alignments may differ, they will likely be more similar than the alignments of independent aligners.

These alignments will still need to be combined procedurally (e.g., taking their union), but because they are more similar, the importance of the combination procedure is reduced. We analyze the behavior of early stopping experimentally in Section 5.

[3]This certificate of optimality is not provided by other approximate inference algorithms, such as belief propagation, sampling, or simulated annealing.

3.6 Inference Properties

Because we set a maximum number of iterations n in the dual decomposition algorithm, and each iteration only involves optimization in a sequence model, our entire inference procedure is only a constant multiple n more computationally expensive than evaluating the original directional aligners.

Moreover, the value of $\mathbf{u}$ is specific to a sentence pair. Therefore, our approach does not require any additional communication overhead relative to the independent directional models in a distributed aligner implementation. Memory requirements are virtually identical to the baseline: only $\mathbf{u}$ must be stored for each sentence pair as it is being processed, but can then be immediately discarded once alignments are inferred.

Other approaches to generating one-to-one phrase alignments are generally more expensive. In particular, an ITG model requires $O(|\boldsymbol{e}|^3 \cdot |\boldsymbol{f}|^3)$ time, whereas our algorithm requires only

$$O(n \cdot (|\boldsymbol{f}||\boldsymbol{e}|^2 + |\boldsymbol{e}||\boldsymbol{f}|^2)) .$$

Moreover, our approach allows Markov distortion potentials, while standard ITG models are restricted to only hierarchical distortion.

4 Related Work

Alignment combination normally involves selecting some $\mathcal{A}$ from the output of two directional models. Common approaches include forming the union or intersection of the directional sets.

$$\mathcal{A}_{\cup} = \mathcal{A}_{\mathbf{a}} \cup \mathcal{A}_{\mathbf{b}}$$
$$\mathcal{A}_{\cap} = \mathcal{A}_{\mathbf{a}} \cap \mathcal{A}_{\mathbf{b}} .$$

More complex combiners, such as the *grow-diag-final* heuristic (Koehn et al., 2003), produce alignment link sets that include all of $\mathcal{A}_{\cap}$ and some subset of $\mathcal{A}_{\cup}$ based on the relationship of multiple links (Och et al., 1999).

In addition, supervised word alignment models often use the output of directional unsupervised aligners as features or pruning signals. In the case

that a supervised model is restricted to proposing alignment links that appear in the output of a directional aligner, these models can be interpreted as a combination technique (Deng and Zhou, 2009). Such a model-based approach differs from ours in that it requires a supervised dataset and treats the directional aligners' output as fixed.

Combination is also related to agreement-based learning (Liang et al., 2006). This approach to jointly learning two directional alignment models yields state-of-the-art unsupervised performance. Our method is complementary to agreement-based learning, as it applies to Viterbi inference under the model rather than computing expectations. In fact, we employ agreement-based training to estimate the parameters of the directional aligners in our experiments.

A parallel idea that closely relates to our bidirectional model is posterior regularization, which has also been applied to the word alignment problem (Graça et al., 2008). One form of posterior regularization stipulates that the posterior probability of alignments from two models must agree, and enforces this agreement through an iterative procedure similar to Algorithm 1. This approach also yields state-of-the-art unsupervised alignment performance on some datasets, along with improvements in end-to-end translation quality (Ganchev et al., 2008).

Our method differs from this posterior regularization work in two ways. First, we iterate over Viterbi predictions rather than posteriors. More importantly, we have changed the output space of the model to be a one-to-one phrase alignment via the coherence edge potential functions.

Another similar line of work applies belief propagation to factor graphs that enforce a one-to-one word alignment (Cromières and Kurohashi, 2009). The details of our models differ: we employ distance-based distortion, while they add structural correspondence terms based on syntactic parse trees. Also, our model training is identical to the HMM-based baseline training, while they employ belief propagation for both training and Viterbi inference. Although differing in both model and inference, our work and theirs both find improvements from defining graphical models for alignment that do not admit exact polynomial-time inference algorithms.

Aligner Model	Intersection $\lvert\mathcal{A}_{\cap}\rvert$	Union $\lvert\mathcal{A}_{\cup}\rvert$	Agreement $\lvert\mathcal{A}_{\cap}\rvert/\lvert\mathcal{A}_{\cup}\rvert$
Baseline	5,554	10,998	50.5%
Bidirectional	7,620	10,262	74.3%

Table 1: The bidirectional model's dual decomposition algorithm substantially increases the overlap between the predictions of the directional models, measured by the number of links in their intersection.

5 Experimental Results

We evaluated our bidirectional model by comparing its output to the annotations of a hand-aligned corpus. In this way, we can show that the bidirectional model improves alignment quality and enables the extraction of more correct phrase pairs.

5.1 Data Conditions

We evaluated alignment quality on a hand-aligned portion of the NIST 2002 Chinese-English test set (Ayan and Dorr, 2006). We trained the model on a portion of FBIS data that has been used previously for alignment model evaluation (Ayan and Dorr, 2006; Haghighi et al., 2009; DeNero and Klein, 2010). We conducted our evaluation on the first 150 sentences of the dataset, following previous work. This portion of the dataset is commonly used to train supervised models.

We trained the parameters of the directional models using the agreement training variant of the expectation maximization algorithm (Liang et al., 2006). Agreement-trained IBM Model 1 was used to initialize the parameters of the HMM-based alignment models (Brown et al., 1993). Both IBM Model 1 and the HMM alignment models were trained for 5 iterations on a 6.2 million word parallel corpus of FBIS newswire. This training regimen on this data set has provided state-of-the-art unsupervised results that outperform IBM Model 4 (Haghighi et al., 2009).

5.2 Convergence Analysis

With $n = 250$ maximum iterations, our dual decomposition inference algorithm only converges 6.2% of the time, perhaps largely due to the fact that the two directional models have different one-to-many structural constraints. However, the dual decompo-

Model	Combiner	Prec	Rec	AER
Baseline	union	57.6	80.0	33.4
	intersect	**86.2**	62.7	27.2
	grow-diag	60.1	78.8	32.1
Bidirectional	union	63.3	**81.5**	29.1
	intersect	77.5	75.1	**23.6**
	grow-diag	65.6	80.6	28.0

Table 2: Alignment error rate results for the bidirectional model versus the baseline directional models. "grow-diag" denotes the grow-diag-final heuristic.

Model	Combiner	Prec	Rec	F1
Baseline	union	**75.1**	33.5	46.3
	intersect	64.3	43.4	51.8
	grow-diag	68.3	37.5	48.4
Bidirectional	union	63.2	44.9	52.5
	intersect	57.1	**53.6**	**55.3**
	grow-diag	60.2	47.4	53.0

Table 3: Phrase pair extraction accuracy for phrase pairs up to length 5. "grow-diag" denotes the grow-diag-final heuristic.

sition algorithm does promote agreement between the two models. We can measure the agreement between models as the fraction of alignment links in the union $\mathcal{A}_\cup$ that also appear in the intersection $\mathcal{A}_\cap$ of the two directional models. Table 1 shows a 47% relative increase in the fraction of links that both models agree on by running dual decomposition (bidirectional), relative to independent directional inference (baseline). Improving convergence rates represents an important area of future work.

5.3 Alignment Error Evaluation

To evaluate alignment error of the baseline directional aligners, we must apply a combination procedure such as union or intersection to $\mathcal{A}_\mathbf{a}$ and $\mathcal{A}_\mathbf{b}$. Likewise, in order to evaluate alignment error for our combined model in cases where the inference algorithm does not converge, we must apply combination to $\mathbf{c}^{(\mathbf{a})}$ and $\mathbf{c}^{(\mathbf{b})}$. In cases where the algorithm does converge, $\mathbf{c}^{(\mathbf{a})} = \mathbf{c}^{(\mathbf{b})}$ and so no further combination is necessary.

We evaluate alignments relative to hand-aligned data using two metrics. First, we measure alignment error rate (AER), which compares the proposed alignment set $\mathcal{A}$ to the sure set $\mathcal{S}$ and possible set $\mathcal{P}$ in the annotation, where $\mathcal{S} \subseteq \mathcal{P}$.

$$\text{Prec}(\mathcal{A}, \mathcal{P}) = \frac{|\mathcal{A} \cap \mathcal{P}|}{|\mathcal{A}|}$$

$$\text{Rec}(\mathcal{A}, \mathcal{S}) = \frac{|\mathcal{A} \cap \mathcal{S}|}{|\mathcal{S}|}$$

$$\text{AER}(\mathcal{A}, \mathcal{S}, \mathcal{P}) = 1 - \frac{|\mathcal{A} \cap \mathcal{S}| + |\mathcal{A} \cap \mathcal{P}|}{|\mathcal{A}| + |\mathcal{S}|}$$

AER results for Chinese-English are reported in Table 2. The bidirectional model improves both precision and recall relative to all heuristic combination techniques, including *grow-diag-final* (Koehn et al., 2003). Intersected alignments, which are one-to-one phrase alignments, achieve the best AER.

Second, we measure phrase extraction accuracy. Extraction-based evaluations of alignment better coincide with the role of word aligners in machine translation systems (Ayan and Dorr, 2006). Let $\text{R}_5(\mathcal{S}, \mathcal{P})$ be the set of phrases up to length 5 extracted from the sure link set $\mathcal{S}$ and possible link set $\mathcal{P}$. Possible links are both included and excluded from phrase pairs during extraction, as in DeNero and Klein (2010). Null aligned words are never included in phrase pairs for evaluation. Phrase extraction precision, recall, and F1 for $\text{R}_5(\mathcal{A}, \mathcal{A})$ are reported in Table 3. Correct phrase pair recall increases from 43.4% to 53.6% (a 23.5% relative increase) for the bidirectional model, relative to the best baseline.

Finally, we evaluated our bidirectional model in a large-scale end-to-end phrase-based machine translation system from Chinese to English, based on the alignment template approach (Och and Ney, 2004). The translation model weights were tuned for both the baseline and bidirectional alignments using lattice-based minimum error rate training (Kumar et al., 2009). In both cases, *union* alignments outperformed other combination heuristics. Bidirectional alignments yielded a modest improvement of 0.2% BLEU[4] on a single-reference evaluation set of sentences sampled from the web (Papineni et al., 2002).

[4]BLEU improved from 29.59% to 29.82% after training IBM Model 1 for 3 iterations and training the HMM-based alignment model for 3 iterations. During training, link posteriors were symmetrized by pointwise linear interpolation.

As our model only provides small improvements in alignment precision and recall for the *union* combiner, the magnitude of the BLEU improvement is not surprising.

6 Conclusion

We have presented a graphical model that combines two classical HMM-based alignment models. Our bidirectional model, which requires no additional learning and no supervised data, can be applied using dual decomposition with only a constant factor additional computation relative to independent directional inference. The resulting predictions improve the precision and recall of both alignment links and extraced phrase pairs in Chinese-English experiments. The best results follow from combination via *intersection*.

Because our technique is defined declaratively in terms of a graphical model, it can be extended in a straightforward manner, for instance with additional potentials on $\mathbf{c}$ or improvements to the component directional models. We also look forward to discovering the best way to take advantage of these new alignments in downstream applications like machine translation, supervised word alignment, bilingual parsing (Burkett et al., 2010), part-of-speech tag induction (Naseem et al., 2009), or cross-lingual model projection (Smith and Eisner, 2009; Das and Petrov, 2011).

References

Necip Fazil Ayan and Bonnie J. Dorr. 2006. Going beyond AER: An extensive analysis of word alignments and their impact on MT. In *Proceedings of the Association for Computational Linguistics*.

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A Gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the Association for Computational Linguistics*.

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

Jamie Brunning, Adria de Gispert, and William Byrne. 2009. Context-dependent alignment models for statistical machine translation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

David Burkett, John Blitzer, and Dan Klein. 2010. Joint parsing and alignment with weakly synchronized grammars. In *Proceedings of the North American Association for Computational Linguistics and IJCNLP*.

Fabien Cromières and Sadao Kurohashi. 2009. An alignment algorithm using belief propagation and a structure-based distortion model. In *Proceedings of the European Chapter of the Association for Computational Linguistics and IJCNLP*.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the Association for Computational Linguistics*.

John DeNero and Dan Klein. 2008. The complexity of phrase alignment problems. In *Proceedings of the Association for Computational Linguistics*.

John DeNero and Dan Klein. 2010. Discriminative modeling of extraction sets for machine translation. In *Proceedings of the Association for Computational Linguistics*.

John DeNero, Alexandre Bouchard-Côté, and Dan Klein. 2008. Sampling alignment structure under a Bayesian translation model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Yonggang Deng and Bowen Zhou. 2009. Optimizing word alignment combination for phrase table training. In *Proceedings of the Association for Computational Linguistics*.

Kuzman Ganchev, Joao Graça, and Ben Taskar. 2008. Better alignments = better translations? In *Proceedings of the Association for Computational Linguistics*.

Joao Graça, Kuzman Ganchev, and Ben Taskar. 2008. Expectation maximization and posterior constraints. In *Proceedings of Neural Information Processing Systems*.

Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised ITG models. In *Proceedings of the Association for Computational Linguistics*.

Xiaodong He. 2007. Using word-dependent transition models in HMM-based word alignment for statistical machine. In *ACL Workshop on Statistical Machine Translation*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Josef Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of the Association for Computational Linguistics*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Tahira Naseem, Benjamin Snyder, Jacob Eisenstein, and Regina Barzilay. 2009. Multilingual part-of-speech tagging: Two unsupervised approaches. *Journal of Artificial Intelligence Research*.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*.

Franz Josef Och, Christopher Tillman, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the Association for Computational Linguistics*.

Alexander M. Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Hiroyuki Shindo, Akinori Fujino, and Masaaki Nagata. 2010. Word alignment with synonym regularization. In *Proceedings of the Association for Computational Linguistics*.

David A. Smith and Jason Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the Conference on Computational linguistics*.

An Algorithm for Unsupervised Transliteration Mining with an Application to Word Alignment

Hassan Sajjad **Alexander Fraser** **Helmut Schmid**
Institute for Natural Language Processing
University of Stuttgart
{sajjad,fraser,schmid}@ims.uni-stuttgart.de

Abstract

We propose a language-independent method for the automatic extraction of transliteration pairs from parallel corpora. In contrast to previous work, our method uses no form of supervision, and does not require linguistically informed preprocessing. We conduct experiments on data sets from the NEWS 2010 shared task on transliteration mining and achieve an F-measure of up to 92%, outperforming most of the semi-supervised systems that were submitted. We also apply our method to English/Hindi and English/Arabic parallel corpora and compare the results with manually built gold standards which mark transliterated word pairs. Finally, we integrate the transliteration module into the GIZA++ word aligner and evaluate it on two word alignment tasks achieving improvements in both precision and recall measured against gold standard word alignments.

1 Introduction

Most previous methods for building transliteration systems were supervised, requiring either handcrafted rules or a clean list of transliteration pairs, both of which are expensive to create. Such resources are also not applicable to other language pairs.

In this paper, we show that it is possible to extract transliteration pairs from a parallel corpus using an unsupervised method. We first align a bilingual corpus at the word level using GIZA++ and create a list of word pairs containing a mix of non-transliterations and transliterations. We train a statistical transliterator on the list of word pairs. We then filter out a few word pairs (those which have the lowest transliteration probabilities according to the trained transliteration system) which are likely to be non-transliterations. We retrain the transliterator on the filtered data set. This process is iterated, filtering out more and more non-transliteration pairs until a nearly clean list of transliteration word pairs is left. The optimal number of iterations is automatically determined by a novel stopping criterion.

We compare our unsupervised transliteration mining method with the semi-supervised systems presented at the NEWS 2010 shared task on transliteration mining (Kumaran et al., 2010) using four language pairs. We refer to this task as NEWS10. These systems used a manually labelled set of data for initial supervised training, which means that they are semi-supervised systems. In contrast, our system is fully unsupervised. We achieve an F-measure of up to 92% outperforming most of the semi-supervised systems.

The NEWS10 data sets are extracted Wikipedia InterLanguage Links (WIL) which consist of parallel phrases, whereas a parallel corpus consists of parallel sentences. Transliteration mining on the WIL data sets is easier due to a higher percentage of transliterations than in parallel corpora. We also do experiments on parallel corpora for two language pairs. To this end, we created gold standards in which sampled word pairs are annotated as either transliterations or non-transliterations. These gold standards have been submitted with the paper as supplementary material as they are available to the research community.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 430–439,
Portland, Oregon, June 19-24, 2011.

Finally we integrate a transliteration module into the GIZA++ word aligner and show that it improves word alignment quality. The transliteration module is trained on the transliteration pairs which our mining method extracts from the parallel corpora. We evaluate our word alignment system on two language pairs using gold standard word alignments and achieve improvements of 10% and 13.5% in precision and 3.5% and 13.5% in recall.

The rest of the paper is organized as follows. In section 2, we describe the filtering model and the transliteration model. In section 3, we present our iterative transliteration mining algorithm and an algorithm which computes a stopping criterion for the mining algorithm. Section 4 describes the evaluation of our mining method through both gold standard evaluation and through using it to improve word alignment quality. In section 5, we present previous work and we conclude in section 6.

2 Models

Our algorithms use two different models. The first model is a joint character sequence model which we apply to transliteration mining. We use the grapheme-to-phoneme converter g2p to implement this model. The other model is a standard phrase-based MT model which we apply to transliteration (as opposed to transliteration mining). We build it using the Moses toolkit.

2.1 Joint Sequence Model Using g2p

Here, we briefly describe g2p using notation from Bisani and Ney (2008). The details of the model, its parameters and the utilized smoothing techniques can be found in Bisani and Ney (2008).

The training data is a list of word pairs (a source word and its presumed transliteration) extracted from a word-aligned parallel corpus. g2p builds a joint sequence model on the character sequences of the word pairs and infers m-to-n alignments between source and target characters with Expectation Maximization (EM) training. The m-to-n character alignment units are referred to as "multigrams".

The model built on multigrams consisting of source and target character sequences greater than one learns too much noise (non-transliteration information) from the training data and performs poorly. In our experiments, we use multigrams with a maximum of one character on the source and one character on the target side (i.e., 0,1-to-0,1 character alignment units).

The N-gram approximation of the joint probability can be defined in terms of multigrams q_i as:

$$p(q_1^k) \approx \prod_{j=1}^{k+1} p(q_j | q_{j-N+1}^{j-1}) \quad (1)$$

where q_0, q_{k+1} are set to a special boundary symbol.

N-gram models of order > 1 did not work well because these models tended to learn noise (information from non-transliteration pairs) in the training data. For our experiments, we only trained g2p with the unigram model.

In test mode, we look for the best sequence of multigrams given a fixed source and target string and return the probability of this sequence.

For the mining process, we trained g2p on lists containing both transliteration pairs and non-transliteration pairs.

2.2 Statistical Machine Transliteration System

We build a phrase-based MT system for transliteration using the Moses toolkit (Koehn et al., 2003). We also tried using g2p for implementing the transliteration decoder but found Moses to perform better. Moses has the advantage of using Minimum Error Rate Training (MERT) which optimizes transliteration accuracy rather than the likelihood of the training data as g2p does. The training data contains more non-transliteration pairs than transliteration pairs. We don't want to maximize the likelihood of the non-transliteration pairs. Instead we want to optimize the transliteration performance for test data. Secondly, it is easy to use a large language model (LM) with Moses. We build the LM on the target word types in the data to be filtered.

For training Moses as a transliteration system, we treat each word pair as if it were a parallel sentence, by putting spaces between the characters of each word. The model is built with the default settings of the Moses toolkit. The distortion limit "d" is set to zero (no reordering). The LM is implemented as a five-gram model using the SRILM-Toolkit (Stolcke, 2002), with Add-1 smoothing for unigrams and Kneser-Ney smoothing for higher n-grams.

3 Extraction of Transliteration Pairs

Training of a supervised transliteration system requires a list of transliteration pairs which is expensive to create. Such lists are usually either built manually or extracted using a classifier trained on manually labelled data and using other language dependent information. In this section, we present an iterative method for the extraction of transliteration pairs from parallel corpora which is fully unsupervised and language pair independent.

Initially, we extract a list of word pairs from a word-aligned parallel corpus using GIZA++. The extracted word pairs are either transliterations, other kinds of translations, or misalignments. In each iteration, we first train g2p on the list of word pairs. Then we delete those 5% of the (remaining) training data which are least likely to be transliterations according to g2p.[1] We determine the best iteration according to our stopping criterion and return the filtered data set from this iteration. The stopping criterion uses unlabelled held-out data to predict the optimal stopping point. The following sections describe the transliteration mining method in detail.

3.1 Methodology

We will first describe the iterative filtering algorithm (Algorithm 1) and then the algorithm for the stopping criterion (Algorithm 2). In practice, we first run Algorithm 2 for 100 iterations to determine the best number of iterations. Then, we run Algorithm 1 for that many iterations.

Initially, the parallel corpus is word-aligned using GIZA++ (Och and Ney, 2003), and the alignments are refined using the grow-diag-final-and heuristic (Koehn et al., 2003). We extract all word pairs which occur as 1-to-1 alignments in the word-aligned corpus. We ignore non-1-to-1 alignments because they are less likely to be transliterations for most language pairs. The extracted set of word pairs will be called "*list of word pairs*" later on. We use the list of word pairs as the training data for Algorithm 1.

Algorithm 1 builds a joint sequence model using g2p on the training data and computes the joint probability of all word pairs according to g2p. We normalize the probabilities by taking the nth square root where n is the average length of the source and the target string. The training data contains mostly non-transliteration pairs and a few transliteration pairs. Therefore the training data is initially very noisy and the joint sequence model is not very accurate. However it can successfully be used to eliminate a few word pairs which are very unlikely to be transliterations.

Algorithm 1 Mining of transliteration pairs

1: training data ←list of word pairs
2: I ← 0
3: **repeat**
4: Build a joint source channel model on the training data using g2p and compute the joint probability of every word pair.
5: Remove the 5% word pairs with the lowest length-normalized probability from the training data. {and repeat the process with the filtered training data}
6: I ← I+1
7: **until** I = Stopping iteration from Algorithm 2

On the filtered training data, we can train a model which is slightly better than the previous model. Using this improved model, we can eliminate further non-transliterations.

Our results show that at the iteration determined by our stopping criterion, the filtered set mostly contains transliterations and only a small number of transliterations have been mistakenly eliminated (see section 4.2).

Algorithm 2 automatically determines the best stopping point of the iterative transliteration mining process. It is an extension of Algorithm 1. It runs the iterative process of Algorithm 1 on half of the list of word pairs (training data) for 100 iterations. For every iteration, it builds a transliteration system on the filtered data. The transliteration system is tested on the source side of the other half of the list of word pairs (held-out). The output of the transliteration system is matched against the target side of the held-out data. (These target words are either transliterations, translations or misalignments.) We match the target side of the held-out data under the assumption that all matches are transliterations. The iteration where the output of the transliteration system best matches the held-out data is chosen as the stopping iteration of Algorithm 1.

[1] Since we delete 5% from the filtered data, the number of deleted data items decreases in each iteration.

Algorithm 2 Selection of the stopping iteration for the transliteration mining algorithm

1: Create clusters of word pairs from the list of word pairs which have a common prefix of length 2 both on the source and target language side.
2: Randomly add each cluster either to the training data or to the held-out data.
3: I $\leftarrow$ 0
4: **while** I $<$ 100 **do**
5: Build a joint sequence model on the training data using g2p and compute the length-normalized joint probability of every word pair in the training data.
6: Remove the 5% word pairs with the lowest probability from the training data. {The training data will be reduced by 5% of the rest in each iteration}
7: Build a transliteration system on the filtered training data and test it using the source side of the held-out and match the output against the target side of the held-out.
8: I $\leftarrow$ I+1
9: **end while**
10: Collect statistics of the matching results and take the median from 9 consecutive iterations (median9).
11: Choose the iteration with the best median9 score for the transliteration mining process.

We will now describe Algorithm 2 in detail. Algorithm 2 initially splits the word pairs into training and held-out data. This could be done randomly, but it turns out that this does not work well for some tasks. The reason is that the parallel corpus contains inflectional variants of the same word. If two variants are distributed over training and held-out data, then the one in the training data may cause the transliteration system to produce a correct translation (but not transliteration) of its variant in the held-out data. This problem is further discussed in section 4.2.2. Instead of randomly splitting the data, we first create clusters of word pairs which have a common prefix of length 2 both on the source and target language side. We randomly add each cluster either to the training data or to the held-out data.

We repeat the mining process (described in Algorithm 1) to eliminate non-transliteration pairs from the training data. For each iteration of Algorithm 2, i.e., steps 4 to 9, we build a transliteration system on the filtered training data and test it on the source side of the held-out. We collect statistics on how well the output of the system matches the target side of the held-out. The matching scores on the held-out data often make large jumps from iteration to iteration. We take the median of the results from 9 consecutive iterations (the 4 iterations before, the current and the 4 iterations after the current iteration) to smooth the scores. We call this median9. We choose the iteration with the best smoothed score as the stopping point for the filtering process. In our tests, the median9 heuristic indicated an iteration close to the optimal iteration.

Sometimes several nearby iterations have the same maximal smoothed score. In that case, we choose the one with the highest unsmoothed score. Section 4.2 explains the median9 heuristic in more detail and presents experimental results showing that it works well.

4 Experiments

We evaluate our transliteration mining algorithm on three tasks: transliteration mining from Wikipedia InterLanguage Links, transliteration mining from parallel corpora, and word alignment using a word aligner with a transliteration component. On the WIL data sets, we compare our fully unsupervised system with the semi-supervised systems presented at the NEWS10 (Kumaran et al., 2010). In the evaluation on parallel corpora, we compare our mining results with a manually built gold standard in which each word pair is either marked as a transliteration or as a non-transliteration. In the word alignment experiment, we integrate a transliteration module which is trained on the transliterations pairs extracted by our method into a word aligner and show a significant improvement. The following sections describe the experiments in detail.

4.1 Experiments Using Parallel Phrases of Wikipedia InterLanguage Links

We conduct transliteration mining experiments on the English/Arabic, English/Hindi, English/Tamil and English/Russian Wikipedia InterLanguage Links (WIL) used in the NEWS10.[2] All data sets

[2]We do not evaluate on the English/Chinese data because the Chinese data requires word segmentation which is beyond the scope of our work. Another problem is that our extraction method was developed for alphabetic languages and probably needs to be adapted before it is applicable to logographic languages such as Chinese.

	Our	S-Best	S-Worst	Systems	Rank
EA	87.4	91.5	70.2	16	3
ET	90.1	91.4	57.5	14	3
EH	92.2	94.4	71.4	14	3

Table 1: Summary of results on NEWS10 data sets where "EA" is English/Arabic, "ET" is English/Tamil and "EH" is English/Hindi. "Our" shows the F-measure of our filtered data against the gold standard using the supplied evaluation tool, "Systems" is the total number of participants in the subtask, and "Rank" is the rank we would have obtained if our system had participated.

contain training data, seed data and reference data. We make no use of the seed data since our system is fully unsupervised. We calculate the F-measure of our filtered transliteration pairs against the supplied gold standard using the supplied evaluation tool.

For English/Arabic, English/Hindi and English/Tamil, our system is better than most of the semi-supervised systems presented at the NEWS 2010 shared task for transliteration mining. Table 1 summarizes the F-scores on these data sets.

On the English/Russian data set, our system achieves 76% F-measure which is not good compared with the systems that participated in the shared task. The English/Russian corpus contains many cognates which – according to the NEWS10 definition – are not transliterations of each other. Our system learns the cognates in the training data and extracts them as transliterations (see Table 2).

The two best teams on the English/Russian task presented various extraction methods (Jiampojamarn et al., 2010; Darwish, 2010). Their systems behave differently on English/Russian than on other language pairs. Their best systems for English/Russian are only trained on the seed data and the use of unlabelled data does not help the performance. Since our system is fully unsupervised, and the unlabelled data is not useful, we perform badly.

4.2 Experiments Using Parallel Corpora

The Wikipedia InterLanguage Links shared task data contains a much larger proportion of transliterations than a parallel corpus. In order to examine how well our method performs on parallel corpora, we apply it to parallel corpora of English/Hindi and English/Arabic, and compare the transliteration mining results with a gold standard.

English	Russian	English	Russian
Studio	Студия/Studiya	Catalonia	Каталонии/Katalonii
Estonia	Эстонии/Estonii	Monastery	Монастырь/Monastyr
Tartary	Тартария/Tartariya	Geography	География/Geografiya

Table 2: Cognates from English/Russian corpus extracted by our system as transliteration pairs. None of them are correct transliteration pairs according to the gold standard.

We use the English/Hindi corpus from the shared task on word alignment, organized as part of the ACL 2005 Workshop on Building and Using Parallel Texts (WA05) (Martin et al., 2005). For English/Arabic, we use a freely available parallel corpus from the United Nations (UN) (Eisele and Chen, 2010). We randomly take 200,000 parallel sentences from the UN corpus of the year 2000. We create gold standards for both language pairs by randomly selecting a few thousand word pairs from the lists of word pairs extracted from the two corpora. We manually tag them as either transliterations or non-transliterations. The English/Hindi gold standard contains 180 transliteration pairs and 2084 non-transliteration pairs and the English/Arabic gold standard contains 288 transliteration pairs and 6639 non-transliteration pairs. We have submitted these gold standards with the paper. They are available to the research community.

In the following sections, we describe the median9 heuristic and the splitting method of Algorithm 2. The splitting method is used to avoid early peaks in the held-out statistics, and the median9 heuristic smooths the held-out statistics in order to obtain a single peak.[3]

4.2.1 Motivation for Median9 Heuristic

Algorithm 2 collects statistics from the held-out data (step 10) and selects the stopping iteration. Due to the noise in the held-out data, the transliteration accuracy on the held-out data often jumps from iteration to iteration. The dotted line in figure 1 (right) shows the held-out prediction accuracy for the En-

[3]We do not use the seed data in our system. However, to check the correctness of the stopping point, we tested the transliteration system on the seed data (available with NEWS10) for every iteration of Algorithm 2. We verified that the median9 held-out statistics and accuracy on the seed data have their peaks at the same iteration.

glish/Hindi parallel corpus. The curve is very noisy and has two peaks. It is difficult to see the effect of the filtering. We take the median of the results from 9 consecutive iterations to smooth the scores. The solid line in figure 1 (right) shows a smoothed curve built using the median9 held-out scores. A comparison with the gold standard (section 4.2.3) shows that the stopping point (peak) reached using the median9 heuristic is better than the stopping point obtained with unsmoothed scores.

4.2.2 Motivation for Splitting Method

Algorithm 2 initially splits the list of word pairs into training and held-out data. A random split worked well for the WIL data, but failed on the parallel corpora. The reason is that parallel corpora contain inflectional variants of the same word. If these variants are randomly distributed over training and held-out data, then a non-transliteration word pair such as the English-Hindi pair "change – badlao" may end up in the training data and the related pair "changes – badlao" in the held-out data. The Moses system used for transliteration will learn to "transliterate" (or actually translate) "change" to "badlao". From other examples, it will learn that a final "s" can be dropped. As a consequence, the Moses transliterator may produce the non-transliteration "badlao" for the English word "changes" in the held-out data. Such matching predictions of the transliterator which are actually translations lead to an overestimate of the transliteration accuracy and may cause Algorithm 2 to predict a stopping iteration which is too early.

By splitting the list of word pairs in such a way that inflectional variants of a word are placed either in the training data, or in the held-out, but not in both, this problem can be solved.[4]

The left graph in Figure 1 shows that the median9 held-out statistics obtained after a random data split of a Hindi/English corpus contains two peaks which occur too early. These peaks disappear in the right graph of Figure 1 which shows the results obtained after a split with the clustering method.

The overall trend of the smoothed curve in figure 1 (right) is very clear. We start by filtering out non-transliteration pairs from the data, so the results

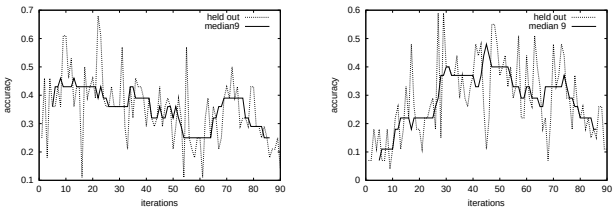


Figure 1: Statistics of held-out prediction of English/Hindi data using modified Algorithm 2 with random division of the list of word pairs (left) and using Algorithm 2 (right). The dotted line shows unsmoothed held-out scores and solid line shows median9 held-out scores

of the transliteration system go up. When no more non-transliteration pairs are left, we start filtering out transliteration pairs and the results of the system go down. We use this stopping criterion for all language pairs and achieve consistently good results.

4.2.3 Results on Parallel Corpora

According to the gold standard, the English/Hindi and English/Arabic data sets contain 8% and 4% transliteration pairs respectively. We repeat the same mining procedure – run Algorithm 2 up to 100 iterations and return the stopping iteration. Then, we run Algorithm 1 up to the stopping iteration returned by Algorithm 2 and obtain the filtered data.

	TP	FN	TN	FP
EH Filtered	170	10	2039	45
EA Filtered	197	91	6580	59

Table 3: Transliteration mining results using the parallel corpus of English/Hindi (EH) and English/Arabic (EA) against the gold standard

Table 3 shows the mining results on the English/Hindi and English/Arabic corpora. The gold standard is a subset of the data sets. The English/Hindi gold standard contains 180 transliteration pairs and 2084 non-transliteration pairs. The English/Arabic gold standard contains 288 transliteration pairs and 6639 non-transliteration pairs. From the English/Hindi data, the mining system has mined 170 transliteration pairs out of 180 transliteration pairs. The English/Arabic mined data contains 197 transliteration pairs out of 288 transliteration pairs. The mining system has wrongly identified a few non-transliteration pairs as transliterations (see

[4]This solution is appropriate for all of the language pairs used in our experiments, but should be revisited if there is inflection realized as prefixes, etc.

table 3, last column). Most of these word pairs are close transliterations and differ by only one or two characters from perfect transliteration pairs. The close transliteration pairs provide many valid multigrams which may be helpful for the mining system.

4.3 Integration into Word Alignment Model

In the previous section, we presented a method for the extraction of transliteration pairs from a parallel corpus. In this section, we will explain how to build a transliteration module on the extracted transliteration pairs and how to integrate it into MGIZA++ (Gao and Vogel, 2008) by interpolating it with the t-table probabilities of the IBM models and the HMM model. MGIZA++ is an extension of GIZA++. It has the ability to resume training from any model rather than starting with Model1.

4.3.1 Modified EM Training of the Word Alignment Models

GIZA++ applies the IBM models (Brown et al., 1993) and the HMM model (Vogel et al., 1996) in both directions, i.e., source to target and target to source. The alignments are refined using the grow-diag-final-and heuristic (Koehn et al., 2003). GIZA++ generates a list of translation pairs with alignment probabilities, which is called the t-table. In this section, we propose a method to modify the translation probabilities of the t-table by interpolating the translation counts with transliteration counts. The interpolation is done in both directions. In the following, we will only consider the **e**-to-**f** direction. The transliteration module which is used to calculate the conditional transliteration probability is described in Algorithm 3.

We build a transliteration system by training Moses on the filtered transliteration corpus (using Algorithm 1) and apply it to the **e** side of the list of word pairs. For every source word, we generate the list of 10-best transliterations $nbestTI(e)$. Then, we extract every **f** that cooccurs with **e** in a parallel sentence and add it to $nbestTI(e)$ which gives us the list of candidate transliteration pairs $candidateTI(e)$. We use the sum of transliteration probabilities $\sum_{f' \in CandidateTI(e)} p_{moses}(f', e)$ as an approximation for the prior probability $p_{moses}(e) = \sum_{f'} p_{moses}(f', e)$ which is needed to convert the joint transliteration probability into a conditional probability. We use the constraint decoding option of Moses to compute the joint probability of **e** and **f**. It computes the probability by dividing the translation score of the best target sentence given a source sentence by the normalization factor.

Algorithm 3 Estimation of transliteration probabilities, **e**-to-**f** direction

```
unfiltered data ←list of word pairs
filtered data ←transliteration pairs extracted using Algorithm 1
Train a transliteration system on the filtered data
for all e do
    nbestTI(e) ← 10 best transliterations for e according to the transliteration system
    cooc(e) ← set of all f that cooccur with e in a parallel sentence
    candidateTI(e) ← cooc(e) ∪ nbestTI(e)
end for
for all f do
    p_moses(f, e) ← joint transliteration probability of e and f according to the transliterator
    p_ti(f|e) ← p_moses(f,e) / Σ_{f'∈CandidateTI(e)} p_moses(f',e)
end for
```

We combine the transliteration probabilities with the translation probabilities of the IBM models and the HMM model. The normal translation probability $p_{ta}(f|e)$ of the word alignment models is computed with relative frequency estimates.

We smooth the alignment frequencies by adding the transliteration probabilities weighted by the factor λ and get the following modified translation probabilities

$$\hat{p}(f|e) = \frac{f_{ta}(f, e) + \lambda p_{ti}(f|e)}{f_{ta}(e) + \lambda} \qquad (2)$$

where $f_{ta}(f, e) = p_{ta}(f|e)f(e)$. $p_{ta}(f|e)$ is obtained from the original t-table of the alignment model. $f(e)$ is the total corpus frequency of **e**. λ is the transliteration weight which is optimized for every language pair (see section 4.3.2). Apart from the definition of the weight λ, our smoothing method is equivalent to Witten-Bell smoothing.

We smooth after every iteration of the IBM models and the HMM model except the last iteration of each model. Algorithm 4 shows the smoothing for IBM Model4. IBM Model1 and the HMM model are smoothed in the same way. We also apply Algorithm 3 and Algorithm 4 in the alignment direction

Algorithm 4 Interpolation with the IBM Model4, **e**-to-**f** direction

1: {We want to run four iterations of Model4}
2: $f(e) \leftarrow$ total frequency of e in the corpus
3: Run MGIZA++ for one iteration of Model4
4: $I \leftarrow 1$
5: **while** $I < 4$ **do**
6: Look up $p_{ta}(f|e)$ in the t-table of Model4
7: $f_{ta}(f,e) \leftarrow p_{ta}(f|e)f(e)$ for all (f,e)
8: $\hat{p}(f|e) \leftarrow \frac{f_{ta}(f,e)+\lambda p_{ti}(f|e)}{f_{ta}(e)+\lambda}$ for all (f,e)
9: Resume MGIZA++ training for 1 iteration using the modified t-table probabilities $\hat{p}(f|e)$
10: $I \leftarrow I + 1$
11: **end while**

f to **e**. The final alignments are generated using the grow-diag-final-and heuristic (Koehn et al., 2003).

4.3.2 Evaluation

The English/Hindi corpus available from WA05 consists of training, development and test data. As development and test data for English/Arabic, we use manually created gold standard word alignments for 155 sentences extracted from the Hansards corpus released by LDC. We use 50 sentences for development and 105 sentences for test.

Baseline: We align the data sets using GIZA++ (Och and Ney, 2003) and refine the alignments using the grow-diag-final-and heuristic (Koehn et al., 2003). We obtain the baseline F-measure by comparing the alignments of the test corpus with the gold standard alignments.

Experiments We use GIZA++ with 5 iterations of Model1, 4 iterations of HMM and 4 iterations of Model4. We interpolate translation and transliteration probabilities at different iterations (and different combinations of iterations) of the three models and always observe an improvement in alignment quality. For the final experiments, we interpolate at every iteration of the IBM models and the HMM model except the last iteration of every model where we could not interpolate for technical reasons.[5] Algorithm 4 shows the interpolation of the transliteration probabilities with IBM Model4. We used the same procedure with IBM Model1 and the HMM model.

The parameter λ is optimized on development data for every language pair. The word alignment system is not very sensitive to λ. Any λ in the range between 50 and 100 works fine for all language pairs. The optimization helps to maximize the improvement in word alignment quality. For our experiments, we use $\lambda = 80$.

On test data, we achieve an improvement of approximately 10% and 13.5% in precision and 3.5% and 13.5% in recall on English/Hindi and English/Arabic word alignment, respectively. Table 4 shows the scores of the baseline and our word alignment model.

Lang	P_b	R_b	F_b	P_{ti}	R_{ti}	F_{ti}
EH	49.1	48.5	51.2	59.1	52.1	55.4
EA	50.8	49.9	50.4	64.4	63.6	64

Table 4: Word alignment results on the test data of English/Hindi (EH) and English/Arabic (EA) where P_b is the precision of baseline GIZA++ and P_{ti} is the precision of our word alignment system

We compared our word alignment results with the systems presented at WA05. Three systems, one limited and two un-limited, participated in the English/Hindi task. We outperform the limited system and one un-limited system.

5 Previous Research

Previous work on transliteration mining uses a manually labelled set of training data to extract transliteration pairs from a parallel corpus or comparable corpora. The training data may contain a few hundred randomly selected transliteration pairs from a transliteration dictionary (Yoon et al., 2007; Sproat et al., 2006; Lee and Chang, 2003) or just a few carefully selected transliteration pairs (Sherif and Kondrak, 2007; Klementiev and Roth, 2006). Our work is more challenging as we extract transliteration pairs without using transliteration dictionaries or gold standard transliteration pairs.

Klementiev and Roth (2006) initialize their transliteration model with a list of 20 transliteration

[5] We had problems in resuming MGIZA++ training when training was supposed to continue from a different model, such as if we stopped after the 5th iteration of Model1 and then tried to resume MGIZA++ from the first iteration of the HMM model. In this case, we ran the 5th iteration of Model1, then the first iteration of the HMM and only then stopped for interpolation; so we did not interpolate in just those iterations of training where we were transitioning from one model to the next.

pairs. Their model makes use of temporal scoring to rank the candidate transliterations. A lot of work has been done on discovering and learning transliterations from comparable corpora by using temporal and phonetic information (Tao et al., 2006; Klementiev and Roth, 2006; Sproat et al., 2006). We do not have access to this information.

Sherif and Kondrak (2007) train a probabilistic transducer on 14 manually constructed transliteration pairs of English/Arabic. They iteratively extract transliteration pairs from the test data and add them to the training data. Our method is different from the method of Sherif and Kondrak (2007) as our method is fully unsupervised, and because in each iteration, they add the most probable transliteration pairs to the training data, while we filter out the least probable transliteration pairs from the training data.

The transliteration mining systems of the four NEWS10 participants are either based on discriminative or on generative methods. All systems use manually labelled (seed) data for the initial training. The system based on the edit distance method submitted by Jiampojamarn et al. (2010) performs best for the English/Russian task. Jiampojamarn et al. (2010) submitted another system based on a standard n-gram kernel which ranked first for the English/Hindi and English/Tamil tasks.[6] For the English/Arabic task, the transliteration mining system of Noeman and Madkour (2010) was best. They normalize the English and Arabic characters in the training data which increases the recall.[7]

Our transliteration extraction method differs in that we extract transliteration pairs from a parallel corpus without supervision. The results of the NEWS10 experiments (Kumaran et al., 2010) show that no single system performs well on all language pairs. Our unsupervised method seems robust as its performance is similar to or better than many of the semi-supervised systems on three language pairs.

We are only aware of one previous work which uses transliteration information for word alignment. Hermjakob (2009) proposed a linguistically focused word alignment system which uses many features including hand-crafted transliteration rules for Arabic/English alignment. His evaluation did not explicitly examine the effect of transliteration (alone) on word alignment. We show that the integration of a transliteration system based on unsupervised transliteration mining increases the word alignment quality for the two language pairs we tested.

[6] They use the seed data as positive examples. In order to obtain also negative examples, they generate all possible word pairs from the source and target words in the seed data and extract the ones which are not transliterations but have a common substring of some minimal length.

[7] They use the phrase table of Moses to build a mapping table between source and target characters. The mapping table is then used to construct a finite state transducer.

6 Conclusion

We proposed a method to automatically extract transliteration pairs from parallel corpora without supervision or linguistic knowledge. We evaluated it against the semi-supervised systems of NEWS10 and achieved high F-measure and performed better than most of the semi-supervised systems. We also evaluated our method on parallel corpora and achieved high F-measure. We integrated the transliteration extraction module into the GIZA++ word aligner and showed gains in alignment quality. We will release our transliteration mining system and word alignment system in the near future.

Acknowledgments

The authors wish to thank the anonymous reviewers for their comments. We would like to thank Christina Lioma for her valuable feedback on an earlier draft of this paper. Hassan Sajjad was funded by the Higher Education Commission (HEC) of Pakistan. Alexander Fraser was funded by Deutsche Forschungsgemeinschaft grant Models of Morphosyntax for Statistical Machine Translation. Helmut Schmid was supported by Deutsche Forschungsgemeinschaft grant SFB 732.

References

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5).

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

Kareem Darwish. 2010. Transliteration mining with phonetic conflation and iterative training. In *Proceedings of the 2010 Named Entities Workshop*, Uppsala, Sweden. Association for Computational Linguistics.

Andreas Eisele and Yu Chen. 2010. MultiUN: A multilingual corpus from United Nation documents. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta.

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, Columbus, Ohio, June. Association for Computational Linguistics.

Ulf Hermjakob. 2009. Improved word alignment with statistics and linguistic heuristics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, Morristown, NJ, USA. Association for Computational Linguistics.

Sittichai Jiampojamarn, Kenneth Dwyer, Shane Bergsma, Aditya Bhargava, Qing Dou, Mi-Young Kim, and Grzegorz Kondrak. 2010. Transliteration generation and mining with limited training resources. In *Proceedings of the 2010 Named Entities Workshop*, Uppsala, Sweden. Association for Computational Linguistics.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, Morristown, NJ, USA.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference*, pages 127–133, Edmonton, Canada.

A Kumaran, Mitesh M. Khapra, and Haizhou Li. 2010. Whitepaper of news 2010 shared task on transliteration mining. In *Proceedings of the 2010 Named Entities Workshop the 48th Annual Meeting of the ACL*, Uppsala, Sweden.

Chun-Jen Lee and Jason S. Chang. 2003. Acquisition of English-Chinese transliterated word pairs from parallel-aligned texts using a statistical machine transliteration model. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and using parallel texts*, Morristown, NJ, USA. ACL.

Joel Martin, Rada Mihalcea, and Ted Pedersen. 2005. Word alignment for languages with scarce resources. In *ParaText '05: Proceedings of the ACL Workshop on Building and Using Parallel Texts*, Morristown, NJ, USA. Association for Computational Linguistics.

Sara Noeman and Amgad Madkour. 2010. Language independent transliteration mining system using finite state automata framework. In *Proceedings of the 2010 Named Entities Workshop*, Uppsala, Sweden. Association for Computational Linguistics.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Tarek Sherif and Grzegorz Kondrak. 2007. Bootstrapping a stochastic transducer for Arabic-English transliteration extraction. In *ACL*, Prague, Czech Republic.

Richard Sproat, Tao Tao, and ChengXiang Zhai. 2006. Named entity transliteration with comparable corpora. In *ACL*.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Intl. Conf. Spoken Language Processing*, Denver, Colorado.

Tao Tao, Su-Yoon Yoon, Andrew Fister, Richard Sproat, and ChengXiang Zhai. 2006. Unsupervised named entity transliteration using temporal and phonetic correlation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Sydney.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *16th International Conference on Computational Linguistics*, pages 836–841, Copenhagen, Denmark.

Su-Youn Yoon, Kyoung-Young Kim, and Richard Sproat. 2007. Multilingual transliteration using feature based phonetic method. In *Proceedings of the 45th Annual Meeting of the ACL*, Prague, Czech Republic.

Beam-Width Prediction for Efficient Context-Free Parsing

Nathan Bodenstab[†] **Aaron Dunlop**[†] **Keith Hall**[‡] **and Brian Roark**[†]
[†] Center for Spoken Language Understanding, Oregon Health & Science University, Portland, OR
[‡]Google, Inc., Zurich, Switzerland
{bodensta,dunlopa,roark}@cslu.ogi.edu kbhall@google.com

Abstract

Efficient decoding for syntactic parsing has become a necessary research area as statistical grammars grow in accuracy and size and as more NLP applications leverage syntactic analyses. We review prior methods for pruning and then present a new framework that unifies their strengths into a single approach. Using a log linear model, we learn the optimal beam-search pruning parameters for each CYK chart cell, effectively predicting the most promising areas of the model space to explore. We demonstrate that our method is faster than coarse-to-fine pruning, exemplified in both the Charniak and Berkeley parsers, by empirically comparing our parser to the Berkeley parser using the same grammar and under identical operating conditions.

1 Introduction

Statistical constituent parsers have gradually increased in accuracy over the past ten years. This accuracy increase has opened the door to automatically derived syntactic information within a number of NLP tasks. Prior work incorporating parse structure into machine translation (Chiang, 2010) and Semantic Role Labeling (Tsai et al., 2005; Punyakanok et al., 2008) indicate that such hierarchical structure can have great benefit over shallow labeling techniques like chunking and part-of-speech tagging.

Although syntax is becoming increasingly important for large-scale NLP applications, constituent parsing is slow — too slow to scale to the size of many potential consumer applications. The exhaustive CYK algorithm has computational complexity $O(n^3|G|)$ where n is the length of the sentence and $|G|$ is the number of grammar productions, a non-negligible constant. Increases in accuracy have primarily been accomplished through an increase in the size of the grammar, allowing individual grammar rules to be more sensitive to their surrounding context, at a considerable cost in efficiency. Grammar transformation techniques such as linguistically inspired non-terminal annotations (Johnson, 1998; Klein and Manning, 2003b) and latent variable grammars (Matsuzaki et al., 2005; Petrov et al., 2006) have increased the grammar size $|G|$ from a few thousand rules to several million in an explicitly enumerable grammar, or even more in an implicit grammar. Exhaustive search for the maximum likelihood parse tree with a state-of-the-art grammar can require over a minute of processing for a single sentence of 25 words, an unacceptable amount of time for real-time applications or when processing millions of sentences. Deterministic algorithms for dependency parsing exist that can extract syntactic dependency structure very quickly (Nivre, 2008), but this approach is often undesirable as constituent parsers are more accurate and more adaptable to new domains (Petrov et al., 2010).

The most accurate constituent parsers, e.g., Charniak (2000), Petrov and Klein (2007a), make use of approximate inference, limiting their search to a fraction of the total search space and achieving speeds of between one and four newspaper sentences per second. The paradigm for building state-of-the-art parsing models is to first design a model structure that can achieve high accuracy and then, after the model has been built, design effective approximate inference methods around that particular model; e.g., coarse-to-fine non-terminal hierarchies for a given model, or agenda-based methods

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 440–449,
Portland, Oregon, June 19-24, 2011.

that are empirically tuned to achieve acceptable efficiency/accuracy operating points. While both of the above mentioned papers use the CYK dynamic programming algorithm to search through possible solutions, their particular methods of approximate inference are quite distinct.

In this paper, we examine a general approach to approximate inference in constituent parsing that learns cell-specific thresholds for arbitrary grammars. For each cell in the CYK chart, we sort all potential constituents in a local agenda, ordered by an estimate of their posterior probability. Given features extracted from the chart cell context – e.g., span width; POS-tags and words surrounding the boundary of the cell – we train a log linear model to predict how many constituents should be popped from the local agenda and added to the chart. As a special case of this approach, we simply predict whether the number to add should be zero or greater than zero, in which case the method can be seen as a cell-by-cell generalization of Roark and Hollingshead's (2008; 2009) tagger-derived Chart Constraints. More generally, instead of a binary classification decision, we can also use this method to predict the desired cell population directly and get cell closure for free when the classifier predicts a beam-width of zero. In addition, we use a non-symmetric loss function during optimization to account for the imbalance between over-predicting or under-predicting the beam-width.

A key feature of our approach is that it does not rely upon reference syntactic annotations when learning to search. Rather, the beam-width prediction model is trained to learn the rank of constituents in the maximum likelihood trees.[1] We will illustrate this by presenting results using a latent-variable grammar, for which there is no "true" reference latent variable parse. We simply parse sections 2-21 of the WSJ treebank and train our search models from the output of these trees, with no prior knowledge of the non-terminal set or other grammar characteristics to guide the process. Hence, this approach is broadly applicable to a wide range of scenarios, including tuning the search to new domains where domain mismatch may yield very different efficiency/accuracy operating points.

[1]Note that we do not call this method "unsupervised" because all grammars used in this paper are induced from supervised data, although our framework can also accommodate unsupervised grammars. We emphasize that we are learning to search using only maximum likelihood trees, not that we are doing unsupervised parsing.

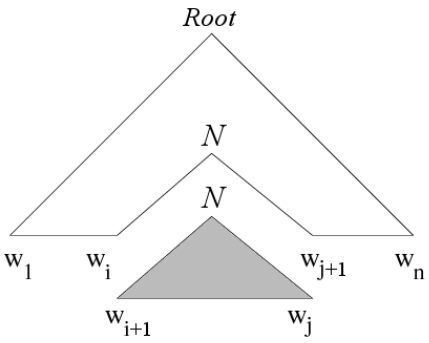


Figure 1: Inside (grey) and outside (white) representations of an example chart edge $N_{i,j}$.

In the next section, we present prior work on approximate inference in parsing, and discuss how our method to learn optimal beam-search parameters unite many of their strengths into a single framework. We then explore using our approach to open or close cells in the chart as an alternative to Roark and Hollingshead (2008; 2009). Finally, we present results which combine cell closure and adaptive beam-width prediction to achieve the most efficient parser.

2 Background

2.1 Preliminaries and notation

Let $S = w_1 \dots w_{|S|}$ represent an input string of $|S|$ words. Let $w_{i,j}$ denote the substring from word w_{i+1} to w_j; i.e., $S = w_{0,|S|}$. We use the term chart *edge* to refer to a non-terminal spanning a specific substring of the input sentence. Let $N_{i,j}$ denote the edge labeled with non-terminal N spanning $w_{i,j}$, for example $\text{NP}_{3,7}$. We define an edge's figure-of-merit (FOM) as an estimate of the product of its inside (β) and outside (α) scores, conceptually the relative merit the edge has to participate in the final parse tree (see Figure 1). More formally:

$$\begin{aligned} \alpha(N_{i,j}) &= P(w_{0,i}, N_{i,j}, w_{j,n}) \\ \beta(N_{i,j}) &= P(w_{i,j}|N) \\ \text{FOM}(N_{i,j}) &= \hat{\alpha}(N_{i,j})\hat{\beta}(N_{i,j}) \end{aligned}$$

With bottom-up parsing, the true inside probability is accumulated and $\beta(N_{i,j})$ does not need to be estimated, improving the FOMs ability to represent the true inside/outside distribution.

In this paper, we use a modified version of the Caraballo and Charniak **Boundary FOM** (1998) for local edge comparison, which computes $\hat{\alpha}(N_{i,j})$ using POS forward-backward scores and POS-to-nonterminal constituent boundary transition probabilities. Details can be found in (**?**).

We also note that in this paper we only use the FOM scoring function to rank constituents in a local agenda. Alternative approaches to ranking competitors are also possible, such as Learning as Search Optimization (Daumé and Marcu, 2005). The method we present in this paper to learn the optimal beam-search parameters is applicable to any ranking function, and we demonstrate this by computing results with both the Boundary FOM and only the inside probability in Section 6.

2.2 Agenda-based parsing

Agenda-based parsers maintain a global agenda of edges, ranked by FOM score. At each iteration, the highest-scoring edge is popped off of the agenda, added to the chart, and combined with other edges already in the chart. The agenda-based approach includes best-first parsing (Bobrow, 1990) and A* parsing (Klein and Manning, 2003a), which differ in whether an admissible FOM estimate $\hat{\alpha}(N_{i,j})$ is required. A* uses an admissible FOM, and thus guarantees finding the maximum likelihood parse, whereas an inadmissible heuristic (best-first) may require less exploration of the search space. Much work has been pursued in both admissible and inadmissible heuristics for agenda parsing (Caraballo and Charniak, 1998; Klein and Manning, 2003a; Pauls et al., 2010).

In this paper, we also make use of agendas, but at a local rather than a global level. We maintain an agenda for each cell, which has two significant benefits: 1) Competing edges can be compared directly, avoiding the difficulty inherent in agenda-based approaches of comparing edges of radically different span lengths and characteristics; and 2) Since the agendas are very small, the overhead of agenda maintenance — a large component of agenda-based parse time — is minimal.

2.3 Beam-search parsing

CYK parsing with a beam-search is a local pruning strategy, comparing edges within the same chart cell. The beam-width can be defined in terms of a threshold in the number of edges allowed, or in terms of a threshold on the difference in probability relative to the highest scoring edge (Collins, 1999; Zhang et al., 2010). For the current paper, we use both kinds of thresholds, avoiding pathological cases that each individual criteria is prone to encounter. Further, unlike most beam-search approaches we will make use of a FOM estimate of the posterior probability of an edge, defined above, as our ranking function. Finally, we will learn log linear models to assign cell-specific thresholds, rather than relying on a single search parameter.

2.4 Coarse-to-Fine Parsing

Coarse-to-fine parsing, also known as multiple pass parsing (Goodman, 1997; Charniak, 2000; Charniak and Johnson, 2005), first parses the input sentence with a simplified (coarse) version of the target (fine) grammar in which multiple non-terminals are merged into a single state. Since the coarse grammar is quite small, parsing is much faster than with the fine grammar, and can quickly yield an estimate of the outside probability $\alpha(\cdot)$ for use in subsequent agenda or beam-search parsing with the fine grammar. This approach can also be used iteratively with grammars of increasing complexity (Petrov and Klein, 2007a).

Building a coarse grammar from a fine grammar is a non-trivial problem, and most often approached with detailed knowledge of the fine grammar being used. For example, Goodman (1997) suggests using a coarse grammar consisting of regular non-terminals, such as NP and VP, and then non-terminals augmented with head-word information for the more accurate second-pass grammar. Such an approach is followed by Charniak (2000) as well. Petrov and Klein (2007a) derive coarse grammars in a more statistically principled way, although the technique is closely tied to their latent variable grammar representation.

To the extent that our cell-specific threshold classifier predicts that a chart cell should contain zero edges or more than zero edges, it is making coarse

predictions about the unlabeled constituent structure of the target parse tree. This aspect of our work is can be viewed as a coarse-to-fine process, though without considering specific grammatical categories or rule productions.

2.5 Chart Constraints

Roark and Hollingshead (2008; 2009) introduced a pruning technique that ignores entire chart cells based on lexical and POS features of the input sentence. They train two finite-state binary taggers: one that allows multi-word constituents to start at a word, and one that allows constituents to end at a word. Given these tags, it is straightforward to completely skip many chart cells during processing.

In this paper, instead of tagging word positions to infer valid constituent spans, we classify chart cells directly. We further generalize this cell classification to predict the beam-width of the chart cell, where a beam-width of zero indicates that the cell is completely closed. We discuss this in detail in the next section.

3 Open/Closed Cell Classification

3.1 Constituent Closure

We first look at the binary classification of chart cells as either open or closed to full constituents, and predict this value from the input sentence alone. This is the same problem that Roark and Hollingshead (2008; 2009) solve with Chart Constraints; however, where they classify lexical items as either beginning or ending a constituent, we classify individual chart cells as open or closed, an approach we call *Constituent Closure*. Although the number of classifications scales quadratically with our approach, the total parse time is still dominated by the $O(n^3|G|)$ parsing complexity and we find that the added level of specificity reduces the search space significantly.

To learn to classify a chart cell spanning words $w_{i+1} \dots w_j$ of a sentence S as open or closed to full constituents, we first map cells in the training corpus to tuples:

$$\Phi(S, i, j) = (\boldsymbol{x}, y) \tag{1}$$

where $\boldsymbol{x}$ is a feature-vector representation of the chart cell and y is the target class 1 if the cell contains an edge from the maximum likelihood parse tree, 0 otherwise. The feature vector $\boldsymbol{x}$ is encoded with the chart cell's absolute and relative span width, as well as unigram and bigram lexical and part-of-speech tag items from $w_{i-1} \dots w_{j+2}$.

Given feature/target tuples $(\boldsymbol{x}, y)$ for every chart cell in every sentence of a training corpus τ, we train a weight vector θ using the averaged perceptron algorithm (Collins, 2002) to learn an open/closed binary decision boundary:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{(\boldsymbol{x},y)\in\Phi(\tau)} L_\lambda(H(\theta \cdot \boldsymbol{x}), y) \tag{2}$$

where $H(\cdot)$ is the unit step function: 1 if the inner product $\theta \cdot \boldsymbol{x} > 0$, and 0 otherwise; and $L_\lambda(\cdot, \cdot)$ is an asymmetric loss function, defined below.

When predicting cell closure, all misclassifications are not equal. If we leave open a cell which contains no edges in the maximum likelihood (ML) parse, we incur the cost of additional processing, but are still able to recover the ML tree. However, if we close a chart cell which contains an ML edge, search errors occur. To deal with this imbalance, we introduce an asymmetric loss function $L_\lambda(\cdot, \cdot)$ to penalize false-negatives more severely during training.

$$L_\lambda(h, y) = \begin{cases} 0 & \text{if } h = y \\ 1 & \text{if } h > y \\ \lambda & \text{if } h < y \end{cases} \tag{3}$$

We found the value $\lambda = 10^2$ to give the best performance on our development set, and we use this value in all of our experiments.

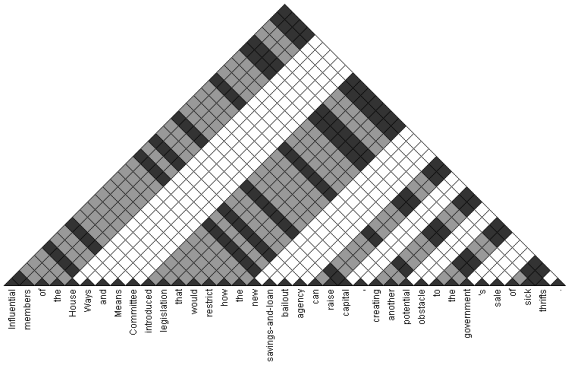
Figures 2a and 2b compare the pruned charts of Chart Constraints and Constituent Closure for a single sentence in the development set. Note that both of these methods are predicting where a complete constituent may be located in the chart, not partial constituents headed by factored nonterminals within a binarized grammar. Depending on the grammar factorization (right or left) we can infer chart cells that are restricted to only edges with a factored left-hand-side non-terminal. In Figure 2 these chart cells are colored gray. Note that Constituent Closure reduces the number of completely open cells considerably vs. Chart Constraints, and the number of cells open to factored categories somewhat.

3.2 Complete Closure

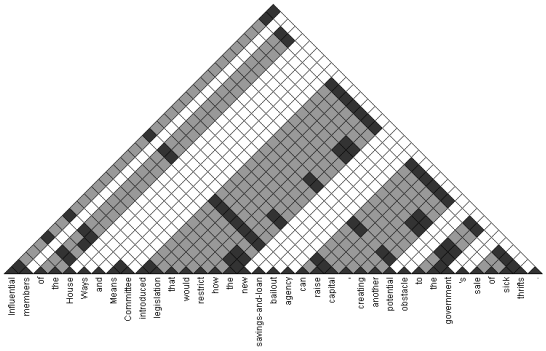
Alternatively, we can predict whether a chart cell contains any edge, either a partial or a full constituent, an approach we call *Complete Closure*. This is a more difficult classification problem as partial constituents occur in a variety of contexts. Nevertheless, learning this directly allows us to remove a large number of internal chart cells from consideration, since no additional cells need to be left open to partial constituents. The learning algorithm is identical to Equation 2, but training examples are now assigned a positive label if the chart cell contains any edge from the binarized maximum likelihood tree. Figure 2c gives a visual representation of Complete Closure for the same sentence; the number of completely open cells increases somewhat, but the total number of open cells (including those open to factored categories) is greatly reduced.

We compare the effectiveness of Constituent Closure, Complete Closure, and Chart Constraints, by decreasing the percentage of chart cells closed until accuracy over all sentences in our development set start to decline. For Constituent and Complete Closure, we also vary the loss function, adjusting the relative penalty between a false-negative (closing off a chart cell that contains a maximum likelihood edge) and a false-positive. Results show that using Chart Constrains as a baseline, we prune (skip) 33% of the total chart cells. Constituent Closure improves on this baseline only slightly (36%), but we see our biggest gains with Complete Closure, which prunes 56% of all chart cells in the development set.

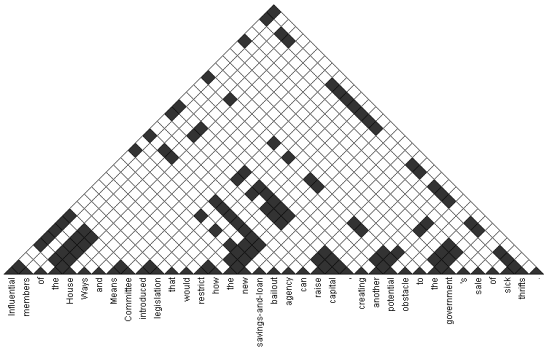
All of these open/closed cell classification methods can improve the efficiency of the exhaustive CYK algorithm, or any of the approximate inference methods mentioned in Section 2. We empirically evaluate them when applied to CYK parsing and beam-search parsing in Section 6.



(a) Chart Constraints (Roark and Hollingshead, 2009)



(b) Constituent Closure (this paper)



(c) Complete Closure (this paper)

Figure 2: Comparison of Chart Constraints (Roark and Hollingshead, 2009) to Constituent and Complete Closure for a single example sentence. Black cells are open to all edges while grey cells only allow factored edges (incomplete constituents).

4 Beam-Width Prediction

The cell-closing approaches discussed in Section 3 make binary decisions to either allow or completely block all edges in each cell. This all-on/all-off tactic ignores the characteristics of the local cell population, which, given a large statistical grammar, may contain hundred of edges, even if very improbable. Retaining all of these partial derivations forces the search in larger spans to continue down improbable paths, adversely affecting efficiency. We can further improve parsing speed in these open cells by leveraging local pruning methods, such as beam-search.

When parsing with a beam-search, finding the optimal beam-width threshold(s) to balance speed and accuracy is a necessary step. As mentioned in Sec-

tion 2.3, two variations of the beam-width are often considered: a fixed number of allowed edges, or a relative probability difference from the highest scoring local edge. For the remainder of this paper we fix the relative probability threshold for all experiments and focus on adapting the number of allowed edges per cell. We will refer to this number-of-allowed-edges value as the beam-width, notated by b, and leave adaptation of the relative probability difference to future work.

The standard way to tune the beam-width is a simple sweep over possible values until accuracy on a heldout data set starts to decline. The optimal point will necessarily be very conservative, allowing outliers (sentences or sub-phrases with above average ambiguity) to stay within the beam and produce valid parse trees. The majority of chart cells will require much fewer than b entries to find the maximum likelihood (ML) edge, yet, constrained by a constant beam-width, the cell will continue to be filled with unfruitful edges, exponentially increasing downstream computation.

For example, when parsing with the Berkeley latent-variable grammar and Boundary FOM, we find we can reduce the global beam-width b to 15 edges in each cell before accuracy starts to decline. However we find that 73% of the ML edges are ranked *first* in their cell and 96% are ranked in the top three. Thus, in 24 of every 25 cells, 80% of the edges are unnecessary (12 of the top 15). Clearly, it would be advantageous to adapt the beam-width such that it is restrictive when we are confident in the FOM ranking and more forgiving in ambiguous contexts.

To address this problem, we learn the optimal beam-width for each chart cell directly. We define $R_{i,j}$ as the rank of the ML edge in the chart cell spanning $w_{i+1} \dots w_j$. If no ML edge exists in the cell, then $R_{i,j} = 0$. Given a global maximum beam-width b, we train b different binary classifiers, each using separate mapping functions Φ_k, where the target value y produced by Φ_k is 1 if $R_{i,j} > k$ and 0 otherwise.

The same asymmetry noted in Section 3 applies in this task as well. When in doubt, we prefer to over-predict the beam-width and risk an increase in processing time opposed to under-predicting at the expense of accuracy. Thus we use the same loss function L_λ, this time training several classifiers:

$$\hat{\theta}_k = \operatorname*{argmin}_{\theta} \sum_{(\boldsymbol{x},y)\in\Phi_k(\tau)} L_\lambda(H(\theta \cdot \boldsymbol{x}), y) \qquad (4)$$

Note that in Equation 4 when $k = 0$, we recover the open/closed cell classification of Equation 2, since a beam width of 0 indicates that the chart cell is completely closed.

During decoding, we assign the beam-width for chart cell spanning $w_{i+1} \dots w_j$ given models $\theta_0, \theta_1, ...\theta_{b-1}$ by finding the lowest value k such that the binary classifier θ_k classifies $R_{i,j} \leq k$. If no such k exists, $\hat{R}_{i,j}$ is set to the maximum beam-width value b:

$$\hat{R}_{i,j} = \operatorname*{argmin}_{k} \theta_k \cdot x_i \leq 0 \qquad (5)$$

In Equation 5 we assume there are b unique classifiers, one for each possible beam-width value between 0 and $b-1$, but this level of granularity is not required. Choosing the number of classification bins to minimize total parsing time is dependent on the FOM function and how it ranks ML edges. With the Boundary FOM we use in this paper, 97.8% of ML edges have a local rank less than five and we find that the added cost of computing b decision boundaries for each cell is not worth the added specificity. We searched over possible classification bins and found that training four classifiers with beam-width decision boundaries at 0, 1, 2, and 4 is faster than 15 individual classifiers and more memory efficient, since each model θ_k has over 800,000 parameters. All beam-width prediction results reported in this paper use these settings.

Figure 3 is a visual representation of beam-width prediction on a single sentence of the development set using the Berkeley latent-variable grammar and Boundary FOM. In this figure, the gray scale represents the relative size of the beam-width, black being the maximum beam-width value, b, and the lightest gray being a beam-width of size one. We can see from this figure that very few chart cells are classified as needing the full 15 edges, apart from span-1 cells which we do not classify.

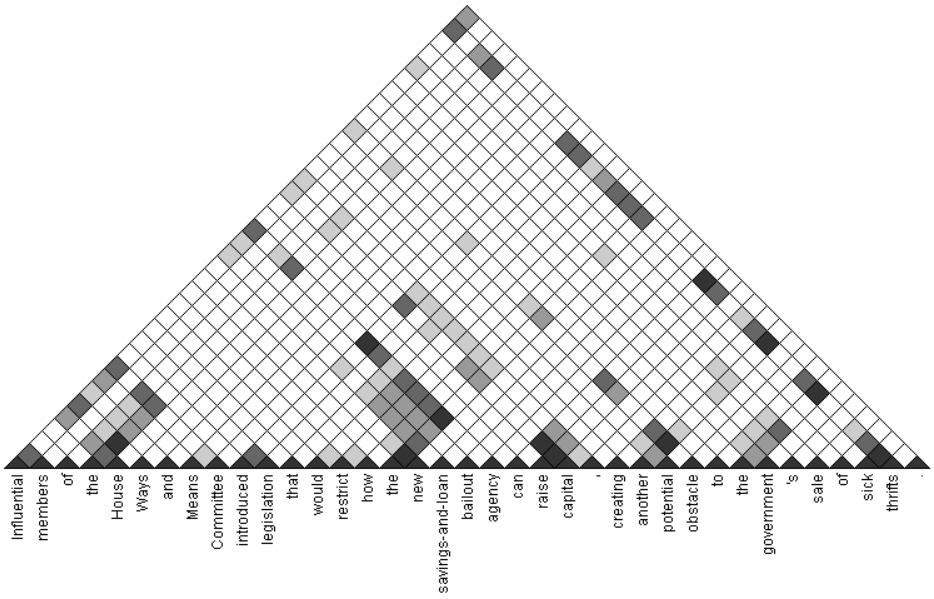


Figure 3: Visualization of Beam-Width Prediction for a single example sentence. The grey scale represents the size of the predicted beam-width: white is 0 (cell is skipped) and black is the maximum value b (b=15 in this example).

5 Experimental Setup

We run all experiments on the WSJ treebank (Marcus et al., 1999) using the standard splits: section 2-21 for training, section 22 for development, and section 23 for testing. We preprocess the treebank by removing empty nodes, temporal labels, and spurious unary productions ($X \rightarrow X$), as is standard in published works on syntactic parsing.

The pruning methods we present in this paper can be used to parse with any grammar. To achieve state-of-the-art accuracy levels, we parse with the Berkeley SM6 latent-variable grammar (Petrov and Klein, 2007b) where the original treebank non-terminals are automatically split into subclasses to optimize parsing accuracy. This is an explicit grammar consisting of 4.3 million productions, 2.4 million of which are lexical productions. Exhaustive CYK parsing with the grammar takes more than a minute per sentence.

Accuracy is computed from the 1-best Viterbi (max) tree extracted from the chart. Alternative decoding methods, such as marginalizing over the latent variables in the grammar or MaxRule decoding (Petrov and Klein, 2007a) are certainly possible in our framework, but it is unknown how effective these methods will be given the heavily pruned nature of the chart. We leave investigation of this to future work. We compute the precision and recall of constituents from the 1-best Viterbi trees using the standard EVALB script (**?**), which ignores punctuation and the root symbol. Accuracy results are reported as F-measure (F_1), the harmonic mean between precision and recall.

We ran all timing tests on an Intel 3.00GHz processor with 6MB of cache and 16GB of memory. Our parser is written in Java and publicly available at http://nlp.csee.ogi.edu.

6 Results

We empirically demonstrate the advantages of our pruning methods by comparing the total parse time of each system, including FOM initialization, chart cell classification, and beam-width prediction. The parse times reported for Chart Constraints do not include tagging times as we were provided with this pre-tagged data, but tagging all of Section 22 takes less than three seconds and we choose to ignore this contribution for simplicity.

Figure 4 contains a timing comparison of the three components of our final parser: Boundary FOM initialization (which includes the forward-backward algorithm over ambiguous part-of-speech tags), beam-

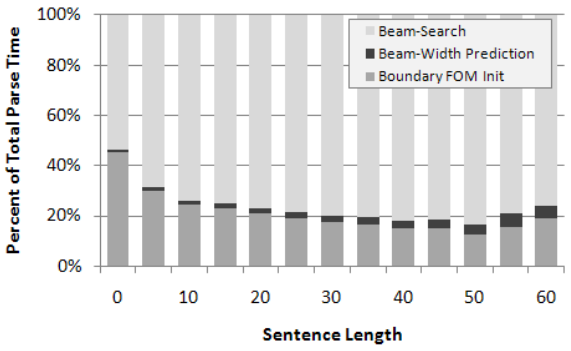


Figure 4: Timing breakdown by sentence length for major components of our parser.

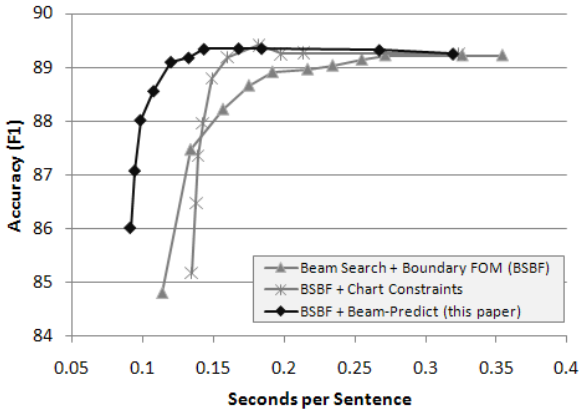


Figure 5: Time vs. accuracy curves comparing beam-width prediction (Beam-Predict) and Chart Constraints.

width prediction, and the final beam-search, including 1-best extraction. We bin these relative times with respect to sentence length to see how each component scales with the number of input words. As expected, the $O(n^3|G|)$ beam-search begins to dominate as the sentence length grows, but Boundary FOM initialization is not cheap, and absorbs, on average, 20% of the total parse time. Beam-width prediction, on the other hand, is almost negligible in terms of processing time even though it scales quadratically with the length of the sentence.

We compare the accuracy degradation of beam-width prediction and Chart Constraints in Figure 5 as we incrementally tighten their respective pruning parameters. We also include the baseline beam-search parser with Boundary FOM in this figure to demonstrate the accuracy/speed trade-off of adjusting a global beam-width alone. In this figure we see that the knee of the beam-width prediction curve (Beam-Predict) extends substantially further to the left before accuracy declines, indicating that our pruning method is intelligently removing a significant portion of the search space that remains unpruned with Chart Constraints.

In Table 1 we present the accuracy and parse time for three baseline parsers on the development set: exhaustive CYK parsing, beam-search parsing using only the inside score $\beta(\cdot)$, and beam-search parsing using the Boundary FOM. We then apply our two cell-closing methods, Constituent Closure and Complete Closure, to all three baselines. As expected, the relative speedup of these methods across the various baselines is similar since the open/closed cell classification does not change across parsers. We also see that Complete Closure is between 22% and 31% faster than Constituent Closure, indicating that the greater number of cells closed translates directly into a reduction in parse time. We can further apply beam-width prediction to the two beam-search baseline parsers in Table 1. Dynamically adjusting the beam-width for the remaining open cells decreases parse time by an additional 25% when using the Inside FOM, and 28% with the boundary FOM.

We apply our best model to the test set and report results in Table 2. Beam-width prediction, again, outperforms the baseline of a constant beam-width by 65% and the open/closed classification of Chart Constraints by 49%. We also compare beam-width prediction to the Berkeley Coarse-to-Fine parser. Both our parser and the Berkeley parser are written in Java, both are run with Viterbi decoding, and both parse with the same grammar, so a direct comparison of speed and accuracy is fair.[2]

7 Conclusion and Future Work

We have introduced three new pruning methods, the best of which unites figure-of-merit estimation from agenda-based parsing, local pruning from beam-search parsing, and unlabeled constituent structure

[2]We run the Berkeley parser with the default search parameterization to achieve the fastest possible parsing time. We note that 3 of 2416 sentences fail to parse under these settings. Using the '-accurate' option provides a valid parse for all sentences, but increases parsing time of section 23 to 0.293 seconds per sentence with no increase in F-score. We assume a back-off strategy for failed parses could be implemented to parse all sentences with a parsing time close to the default parameterization.

Parser	Sec/Sent	F_1
CYK	70.383	89.4
CYK + Constituent Closure	47.870	89.3
CYK + Complete Closure	32.619	89.3
Beam + Inside FOM (BI)	3.977	89.2
BI + Constituent Closure	2.033	89.2
BI + Complete Closure	1.575	89.3
BI + Beam-Predict	1.180	89.3
Beam + Boundary FOM (BB)	0.326	89.2
BB + Constituent Closure	0.279	89.2
BB + Complete Closure	0.199	89.3
BB + Beam-Predict	0.143	89.3

Table 1: Section 22 development set results for CYK and Beam-Search (Beam) parsing using the Berkeley latent-variable grammar.

Parser	Sec/Sent	F_1
CYK	64.610	88.7
Berkeley CTF MaxRule	0.213	90.2
Berkeley CTF Viterbi	0.208	88.8
Beam + Boundary FOM (BB)	0.334	88.6
BB + Chart Constraints	0.244	88.7
BB + Beam-Predict (this paper)	0.125	88.7

Table 2: Section 23 test set results for multiple parsers using the Berkeley latent-variable grammar.

prediction from coarse-to-fine parsing and Chart Constraints. Furthermore, our pruning method is trained using only maximum likelihood trees, allowing it to be tuned to specific domains without labeled data. Using this framework, we have shown that we can decrease parsing time by 65% over a standard beam-search without any loss in accuracy, and parse significantly faster than both the Berkeley parser and Chart Constraints.

We plan to explore a number of remaining questions in future work. First, we will try combining our approach with constituent-level Coarse-to-Fine pruning. The two methods prune the search space in very different ways and may prove to be complementary. On the other hand, our parser currently spends 20% of the total parse time initializing the FOM, and adding additional preprocessing costs, such as parsing with a coarse grammar, may not outweigh the benefits gained in the final search.

Second, as with Chart Constraints we do not prune lexical or unary edges in the span-1 chart cells (i.e., chart cells that span a single word). We expect pruning entries in these cells would notably reduce parse time since they cause exponentially many chart edges to be built in larger spans. Initial work constraining span-1 chart cells has promising results (Bodenstab et al., 2011) and we hope to investigate its interaction with beam-width prediction even further.

Finally, the size and structure of the grammar is the single largest contributor to parse efficiency. In contrast to the current paradigm, we plan to investigate new algorithms that jointly optimize accuracy and efficiency during grammar induction, leading to more efficient decoding.

Acknowledgments

We would like to thank Kristy Hollingshead for her valuable discussions, as well as the anonymous reviewers who gave very helpful feedback. This research was supported in part by NSF Grants #IIS-0447214, #IIS-0811745 and DARPA grant #HR0011-09-1-0041. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF or DARPA.

References

Robert J. Bobrow. 1990. Statistical agenda parsing. In *DARPA Speech and Language Workshop*, pages 222–224.

Nathan Bodenstab, Kristy Hollingshead, and Brian Roark. 2011. Unary constraints for efficient context-free parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, Portland, Oregon.

Sharon A Caraballo and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24:275–298.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180, Ann Arbor, Michigan.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American*

chapter of the Association for Computational Linguistics conference, pages 132–139, Seattle, Washington.

David Chiang. 2010. Learning to translate with source and target syntax. In *Proceedings of the 48rd Annual Meeting on Association for Computational Linguistics*, pages 1443–1452.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. PhD dissertation, University of Pennsylvania.

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical Methods in Natural Language Processing*, volume 10, pages 1–8, Philadelphia.

Hal Daumé, III and Daniel Marcu. 2005. Learning as search optimization: approximate large margin methods for structured prediction. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 169–176, New York, NY, USA.

Joshua Goodman. 1997. Global thresholding and Multiple-Pass parsing. *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 11–25.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Dan Klein and Christopher D. Manning. 2003a. A* parsing. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL '03)*, pages 40–47, Edmonton, Canada.

Dan Klein and Christopher D. Manning. 2003b. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pages 423–430, Sapporo, Japan.

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

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL '05*, pages 75–82, Ann Arbor, Michigan.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Comput. Linguist.*, 34:513–553.

Adam Pauls, Dan Klein, and Chris Quirk. 2010. Top-down k-best a* parsing. In *In proceedings of the Annual Meeting on Association for Computational Linguistics Short Papers*, ACLShort '10, pages 200–204, Morristown, NJ, USA.

Slav Petrov and Dan Klein. 2007a. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York.

Slav Petrov and Dan Klein. 2007b. Learning and inference for hierarchically split PCFGs. In *AAAI 2007 (Nectar Track)*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia.

Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 705–713, Cambridge, MA, October.

Vasin Punyakanok, Dan Roth, and Wen tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Brian Roark and Kristy Hollingshead. 2008. Classifying chart cells for quadratic complexity context-free inference. In Donia Scott and Hans Uszkoreit, editors, *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 745–752, Manchester, UK.

Brian Roark and Kristy Hollingshead. 2009. Linear complexity Context-Free parsing pipelines via chart constraints. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 647–655, Boulder, Colorado.

Tzong-Han Tsai, Chia-Wei Wu, Yu-Chun Lin, and Wen-Lian Hsu. 2005. Exploiting full parsing information to label semantic roles using an ensemble of ME and SVM via integer linear programming. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 233–236, Morristown, NJ, USA.

Yue Zhang, Byung gyu Ahn, Stephen Clark, Curt Van Wyk, James R. Curran, and Laura Rimell. 2010. Chart pruning for fast Lexicalised-Grammar parsing. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1472–1479, Beijing, China.

Optimal Head-Driven Parsing Complexity for Linear Context-Free Rewriting Systems

Pierluigi Crescenzi
Dip. di Sistemi e Informatica
Università di Firenze

Daniel Gildea
Computer Science Dept.
University of Rochester

Andrea Marino
Dip. di Sistemi e Informatica
Università di Firenze

Gianluca Rossi
Dip. di Matematica
Università di Roma *Tor Vergata*

Giorgio Satta
Dip. di Ingegneria dell'Informazione
Università di Padova

Abstract

We study the problem of finding the best *head-driven* parsing strategy for Linear Context-Free Rewriting System productions. A head-driven strategy must begin with a specified righthand-side nonterminal (the head) and add the remaining nonterminals one at a time in any order. We show that it is NP-hard to find the best head-driven strategy in terms of either the time or space complexity of parsing.

1 Introduction

Linear Context-Free Rewriting Systems (LCFRSs) (Vijay-Shankar et al., 1987) constitute a very general grammatical formalism which subsumes context-free grammars (CFGs) and tree adjoining grammars (TAGs), as well as the synchronous context-free grammars (SCFGs) and synchronous tree adjoining grammars (STAGs) used as models in machine translation.[1] LCFRSs retain the fundamental property of CFGs that grammar nonterminals rewrite independently, but allow nonterminals to generate discontinuous phrases, that is, to generate more than one span in the string being produced. This important feature has been recently exploited by Maier and Søgaard (2008) and Kallmeyer and Maier (2010) for modeling phrase structure treebanks with discontinuous constituents, and by Kuhlmann and Satta (2009) for modeling non-projective dependency treebanks.

The rules of a LCFRS can be analyzed in terms of the properties of *rank* and *fan-out*. Rank is the number of nonterminals on the right-hand side (rhs) of a rule, while fan-out is the number of spans of the string generated by the nonterminal in the left-hand side (lhs) of the rule. CFGs are equivalent to LCFRSs with fan-out one, while TAGs are one type of LCFRSs with fan-out two. Rambow and Satta (1999) show that rank and fan-out induce an infinite, two-dimensional hierarchy in terms of generative power; while CFGs can always be reduced to rank two (Chomsky Normal Form), this is not the case for LCFRSs with any fan-out greater than one.

General algorithms for parsing LCFRSs build a dynamic programming chart of recognized nonterminals bottom-up, in a manner analogous to the CKY algorithm for CFGs (Hopcroft and Ullman, 1979), but with time and space complexity that are dependent on the rank and fan-out of the grammar rules. Whenever it is possible, binarization of LCFRS rules, or reduction of rank to two, is therefore important for parsing, as it reduces the time complexity needed for dynamic programming. This has lead to a number of binarization algorithms for LCFRSs, as well as *factorization* algorithms that factor rules into new rules with smaller rank, without necessarily reducing rank all the way to two. Kuhlmann and Satta (2009) present an algorithm for binarizing certain LCFRS rules without increasing their fan-out, and Sagot and Satta (2010) show how to reduce rank to the lowest value possible for LCFRS rules of fan-out two, again without increasing fan-out. Gómez-Rodríguez et al. (2010) show how to factorize *well-nested* LCFRS rules of arbitrary fan-out for efficient parsing.

In general there may be a trade-off required between rank and fan-out, and a few recent papers have investigated this trade-off taking gen-

[1] To be more precise, SCFGs and STAGs generate languages composed by pair of strings, while LCFRSs generate string languages. We can abstract away from this difference by assuming concatenation of components in a string pair.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 450–459,
Portland, Oregon, June 19-24, 2011.

eral LCFRS rules as input. Gómez-Rodríguez et al. (2009) present an algorithm for binarization of LCFRSs while keeping fan-out as small as possible. The algorithm is exponential in the resulting fan-out, and Gómez-Rodríguez et al. (2009) mention as an important open question whether polynomial-time algorithms to minimize fan-out are possible. Gildea (2010) presents a related method for binarizing rules while keeping the time complexity of parsing as small as possible. Binarization turns out to be possible with no penalty in time complexity, but, again, the factorization algorithm is exponential in the resulting time complexity. Gildea (2011) shows that a polynomial time algorithm for factorizing LCFRSs in order to minimize time complexity would imply an improved approximation algorithm for the well-studied graph-theoretic property known as treewidth. However, whether the problem of factorizing LCFRSs in order to minimize time complexity is NP-hard is still an open question in the above works.

Similar questions have arisen in the context of machine translation, as the SCFGs used to model translation are also instances of LCFRSs, as already mentioned. For SCFG, Satta and Peserico (2005) showed that the exponent in the time complexity of parsing algorithms must grow at least as fast as the square root of the rule rank, and Gildea and Štefankovič (2007) tightened this bound to be linear in the rank. However, neither paper provides an algorithm for finding the best parsing strategy, and Huang et al. (2009) mention that whether finding the optimal parsing strategy for an SCFG rule is NP-hard is an important problem for future work.

In this paper, we investigate the problem of rule binarization for LCFRSs in the context of *head-driven* parsing strategies. Head-driven strategies begin with one rhs symbol, and add one nonterminal at a time. This rules out any factorization in which two subsets of nonterminals of size greater than one are combined in a single step. Head-driven strategies allow for the techniques of lexicalization and Markovization that are widely used in (projective) statistical parsing (Collins, 1997). The statistical LCFRS parser of Kallmeyer and Maier (2010) binarizes rules head-outward, and therefore adopts what we refer to as a head-driven strategy. However, the binarization used by Kallmeyer and Maier (2010) simply proceeds left to right through the rule, without considering the impact of the parsing strategy on either time or space complexity. We examine the question of whether we can efficiently find the strategy that minimizes either the time complexity or the space complexity of parsing. While a naive algorithm can evaluate all $r!$ head-driven strategies in time $O(n \cdot r!)$, where r is the rule's rank and n is the total length of the rule's description, we wish to determine whether a polynomial-time algorithm is possible.

Since parsing problems can be cast in terms of logic programming (Shieber et al., 1995), we note that our problem can be thought of as a type of query optimization for logic programming. Query optimization for logic programming is NP-complete since query optimization for even simple conjunctive database queries is NP-complete (Chandra and Merlin, 1977). However, the fact that variables in queries arising from LCFRS rules correspond to the endpoints of spans in the string to be parsed means that these queries have certain structural properties (Gildea, 2011). We wish to determine whether the structure of LCFRS rules makes efficient factorization algorithms possible.

In the following, we show both the the time- and space-complexity problems to be NP-hard for head-driven strategies. We provide what is to our knowledge the first NP-hardness result for a grammar factorization problem, which we hope will aid in understanding parsing algorithms in general.

2 LCFRSs and parsing complexity

In this section we briefly introduce LCFRSs and define the problem of optimizing head-driven parsing complexity for these formalisms. For a positive integer n, we write $[n]$ to denote the set $\{1, \ldots, n\}$.

As already mentioned in the introduction, LCFRSs generate tuples of strings over some finite alphabet. This is done by associating each production p of a grammar with a function g that takes as input the tuples generated by the nonterminals in p's rhs, and rearranges their string components into a new tuple, possibly adding some alphabet symbols.

Let V be some finite alphabet. We write V^* for the set of all (finite) strings over V. For natural numbers $r \geq 0$ and $f, f_1, \ldots, f_r \geq 1$, consider a func-

tion $g : (V^*)^{f_1} \times \cdots \times (V^*)^{f_r} \rightarrow (V^*)^f$ defined by an equation of the form

$$g(\langle x_{1,1}, \ldots, x_{1,f_1}\rangle, \ldots, \langle x_{r,1}, \ldots, x_{r,f_r}\rangle) = \vec{\alpha}\ .$$

Here the $x_{i,j}$'s denote variables over strings in V^*, and $\vec{\alpha} = \langle \alpha_1, \ldots, \alpha_f\rangle$ is an f-tuple of strings over g's argument variables and symbols in V. We say that g is **linear, non-erasing** if $\vec{\alpha}$ contains *exactly one* occurrence of each argument variable. We call r and f the **rank** and the **fan-out** of g, respectively, and write $r(g)$ and $f(g)$ to denote these quantities.

Example 1 $g_1(\langle x_{1,1}, x_{1,2}\rangle) = \langle x_{1,1}x_{1,2}\rangle$ takes as input a tuple with two strings and returns a tuple with a single string, obtained by concatenating the components in the input tuple. $g_2(\langle x_{1,1}, x_{1,2}\rangle) = \langle ax_{1,1}b, cx_{1,2}d\rangle$ takes as input a tuple with two strings and wraps around these strings with symbols $a, b, c, d \in V$. Both functions are linear, non-erasing, and we have $r(g_1) = r(g_2) = 1$, $f(g_1) = 1$ and $f(g_2) = 2$. □

A **linear context-free rewriting system** is a tuple $G = (V_N, V_T, P, S)$, where V_N and V_T are finite, disjoint alphabets of nonterminal and terminal symbols, respectively. Each $A \in V_N$ is associated with a value $f(A)$, called its **fan-out**. The nonterminal S is the start symbol, with $f(S) = 1$. Finally, P is a set of productions of the form

$$p:\ \ A\ \ \rightarrow\ \ g(A_1, A_2, \ldots, A_{r(g)})\ , \qquad (1)$$

where $A, A_1, \ldots, A_{r(g)} \in V_N$, and $g : (V_T^*)^{f(A_1)} \times \cdots \times (V_T^*)^{f(A_{r(g)})} \rightarrow (V_T^*)^{f(A)}$ is a linear, non-erasing function.

Production (1) can be used to transform the $r(g)$ string tuples generated by the nonterminals $A_1, \ldots, A_{r(g)}$ into a tuple of $f(A)$ strings generated by A. The values $r(g)$ and $f(g)$ are called the **rank** and **fan-out** of p, respectively, written $r(p)$ and $f(p)$. Given that $f(S) = 1$, S generates a set of strings, defining the language $L(G)$.

Example 2 Let g_1 and g_2 be as in Example 1, and let $g_3() = \langle \varepsilon, \varepsilon\rangle$. Consider the LCFRS G defined by the productions $p_1 : S \rightarrow g_1(A)$, $p_2 : A \rightarrow g_2(A)$ and $p_3 : A \rightarrow g_3()$. We have $f(S) = 1$, $f(A) = f(G) = 2$, $r(p_3) = 0$ and $r(p_1) = r(p_2) = r(G) = 1$. We have $L(G) = \{a^n b^n c^n d^n \mid n \geq 1\}$. For instance, the string $a^3b^3c^3d^3$ is generated by means of the following bottom-up process. First, the tuple $\langle \varepsilon, \varepsilon\rangle$ is generated by A through p_3. We then iterate three times the application of p_2 to $\langle \varepsilon, \varepsilon\rangle$, resulting in the tuple $\langle a^3b^3, c^3d^3\rangle$. Finally, the tuple (string) $\langle a^3b^3c^3d^3\rangle$ is generated by S through application of p_1. □

fan-out	*strategy*
4	$((A_1 \oplus A_4) \oplus A_3)^* \oplus A_2$
3	$(A_1 \oplus A_4)^* \oplus (A_2 \oplus A_3)$
3	$((A_1 \oplus A_2)^* \oplus A_4) \oplus A_3$
2	$((A_2^* \oplus A_3) \oplus A_4) \oplus A_1$

Figure 1: Some parsing strategies for production p in Example 3, and the associated maximum value for fan-out. Symbol $\oplus$ denotes the merging operation, and superscript $*$ marks the first step in the strategy in which the highest fan-out is realized.

Existing parsing algorithms for LCFRSs exploit dynamic programming. These algorithms compute partial parses of the input string w, represented by means of specialized data structures called items. Each **item** indexes the boundaries of the segments of w that are spanned by the partial parse. In the special case of parsing based on CFGs, an item consists of two indices, while for TAGs four indices are required.

In the general case of LCFRSs, parsing of a production p as in (1) can be carried out in $r(g) - 1$ steps, collecting already available parses for nonterminals $A_1, \ldots, A_{r(g)}$ one at a time, and 'merging' these into intermediate partial parses. We refer to the order in which nonterminals are merged as a parsing strategy, or, equivalently, a factorization of the original grammar rule. Any parsing strategy results in a complete parse of p, spanning $f(p) = f(A)$ segments of w and represented by some item with $2f(A)$ indices. However, intermediate items obtained in the process might span more than $f(A)$ segments. We illustrate this through an example.

Example 3 Consider a linear non-erasing function $g(\langle x_{1,1}, x_{1,2}\rangle, \langle x_{2,1}, x_{2,2}\rangle, \langle x_{3,1}, x_{3,2}\rangle, \langle x_{4,1}, x_{4,2}\rangle) = \langle x_{1,1}x_{2,1}x_{3,1}x_{4,1},\ x_{3,2}x_{2,2}x_{4,2}x_{1,2}\rangle$, and a production $p : A \rightarrow g(A_1, A_2, A_3, A_4)$, where all the nonterminals involved have fan-out 2. We could parse p starting from A_1, and then merging with A_4,

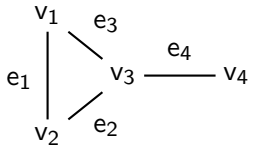


Figure 2: Example input graph for our construction of an LCFRS production.

A_3, and A_2. In this case, after we have collected the first three nonterminals, we have obtained a partial parse having fan-out 4, that is, an item spanning 4 segments of the input string. Alternatively, we could first merge A_1 and A_4, then merge A_2 and A_3, and finally merge the two obtained partial parses. This strategy is slightly better, resulting in a maximum fan-out of 3. Other possible strategies can be explored, displayed in Figure 1. It turns out that the best parsing strategy leads to fan-out 2. □

The maximum fan-out f realized by a parsing strategy determines the space complexity of the parsing algorithm. For an input string w, items will require (in the worst-case) $2f$ indices, each taking $\mathcal{O}(|w|)$ possible values. This results in space complexity of $\mathcal{O}(|w|^{2f})$. In the special cases of parsing based on CFGs and TAGs, this provides the well-known space complexity of $\mathcal{O}(|w|^2)$ and $\mathcal{O}(|w|^4)$, respectively.

It can also be shown that, if a partial parse having fan-out f is obtained by means of the combination of two partial parses with fan-out f_1 and f_2, respectively, the resulting time complexity will be $\mathcal{O}(|w|^{f+f_1+f_2})$ (Seki et al., 1991; Gildea, 2010). As an example, in the case of parsing based on CFGs, nonterminals as well as partial parses all have fan-out one, resulting in the standard time complexity of $\mathcal{O}(|w|^3)$ of dynamic programming methods. When parsing with TAGs, we have to manipulate objects with fan-out two (in the worst case), resulting in time complexity of $\mathcal{O}(|w|^6)$.

We investigate here the case of general LCFRS productions, whose internal structure is considerably more complex than the context-free or the tree adjoining case. Optimizing the parsing complexity for a production means finding a parsing strategy that results in minimum space or time complexity.

We now turn the above optimization problems into decision problems. In the MIN SPACE STRATEGY problem one takes as input an LCFRS production p and an integer k, and must decide whether there exists a parsing strategy for p with maximum fan-out not larger than k. In the MIN TIME STRATEGY problem one is given p and k as above and must decide whether there exists a parsing strategy for p such that, in any of its steps merging two partial parses with fan-out f_1 and f_2 and resulting in a partial parse with fan-out f, the relation $f + f_1 + f_2 \leq k$ holds.

In this paper we investigate the above problems in the context of a specific family of linguistically motivated parsing strategies for LCFRSs, called head-driven. In a **head-driven** strategy, one always starts parsing a production p from a fixed nonterminal in its rhs, called the **head** of p, and merges the remaining nonterminals one at a time with the partial parse containing the head. Thus, under these strategies, the construction of partial parses that do not include the head is forbidden, and each parsing step involves at most one partial parse. In Figure 1, all of the displayed strategies but the one in the second line are head-driven (for different choices of the head).

3 NP-completeness results

For an LCFRS production p, let H be its head nonterminal, and let $A_1, \ldots, A_n$ be all the non-head nonterminals in p's rhs, with $n + 1 = r(p)$. A head-driven parsing strategy can be represented as a permutation π over the set $[n]$, prescribing that the non-head nonterminals in p's rhs should be merged with H in the order $A_{\pi(1)}, A_{\pi(2)}, \ldots, A_{\pi(n)}$. Note that there are $n!$ possible head-driven parsing strategies.

To show that MIN SPACE STRATEGY is NP-hard under head-driven parsing strategies, we reduce from the MIN CUT LINEAR ARRANGEMENT problem, which is a decision problem over (undirected) graphs. Given a graph $M = (V, E)$ with set of vertices V and set of edges E, a **linear arrangement** of M is a bijective function h from V to $[n]$, where $|V| = n$. The **cutwidth** of M at gap $i \in [n-1]$ and with respect to a linear arrangement h is the number of edges crossing the gap between the i-th vertex and its successor:

$$cw(M, h, i) = |\{(u, v) \in E \mid h(u) \leq i < h(v)\}| \,.$$

$$p: \quad A \rightarrow g(H, A_1, A_2, A_3, A_4)$$

$$g(\langle x_{H,e_1}, x_{H,e_2}, x_{H,e_3}, x_{H,e_4}\rangle, \langle x_{A_1,e_1,l}, x_{A_1,e_1,r}, x_{A_1,e_3,l}, x_{A_1,e_3,r}\rangle, \langle x_{A_2,e_1,l}, x_{A_2,e_1,r}, x_{A_2,e_2,l}, x_{A_2,e_2,r}\rangle,$$
$$\langle x_{A_3,e_2,l}, x_{A_3,e_2,r}, x_{A_3,e_3,l}, x_{A_3,e_3,r}, x_{A_3,e_4,l}, x_{A_3,e_4,r}\rangle, \langle x_{A_4,e_4,l}, x_{A_4,e_4,r}\rangle) =$$
$$\langle \quad x_{A_1,e_1,l}x_{A_2,e_1,l}x_{H,e_1}x_{A_1,e_1,r}x_{A_2,e_1,r}, \quad x_{A_2,e_2,l}x_{A_3,e_2,l}x_{H,e_2}x_{A_2,e_2,r}x_{A_3,e_2,r},$$
$$x_{A_1,e_3,l}x_{A_3,e_3,l}x_{H,e_3}x_{A_1,e_3,r}x_{A_3,e_3,r}, \quad x_{A_3,e_4,l}x_{A_4,e_4,l}x_{H,e_4}x_{A_3,e_4,r}x_{A_4,e_4,r} \quad \rangle$$

Figure 3: The construction used to prove Theorem 1 builds the LCFRS production p shown, when given as input the graph of Figure 2.

The cutwidth of M is then defined as

$$cw(M) = \min_{h} \max_{i \in [n-1]} cw(M, h, i) \ .$$

In the MIN CUT LINEAR ARRANGEMENT problem, one is given as input a graph M and an integer k, and must decide whether $cw(M) \leq k$. This problem has been shown to be NP-complete (Gavril, 1977).

Theorem 1 *The* MIN SPACE STRATEGY *problem restricted to head-driven parsing strategies is NP-complete.*

PROOF We start with the NP-hardness part. Let $M = (V, E)$ and k be an input instance for MIN CUT LINEAR ARRANGEMENT, and let $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_q\}$. We assume there are no self loops in M, since these loops do not affect the value of the cutwidth and can therefore be removed. We construct an LCFRS production p and an integer k' as follows.

Production p has a head nonterminal H and a non-head nonterminal A_i for each vertex $v_i \in V$. We let H generate tuples with a string component for each edge $e_i \in E$. Thus, we have $f(H) = q$. Accordingly, we use variables x_{H,e_i}, for each $e_i \in E$, to denote the string components in tuples generated by H.

For each $v_i \in V$, let $E(v_i) \subseteq E$ be the set of edges impinging on v_i; thus $|E(v_i)|$ is the degree of v_i. We let A_i generate a tuple with two string components for each $e_j \in E(v_i)$. Thus, we have $f(A_i) = 2 \cdot |E(v_i)|$. Accordingly, we use variables $x_{A_i,e_j,l}$ and $x_{A_i,e_j,r}$, for each $e_j \in E(v_i)$, to denote the string components in tuples generated by A_i (here subscripts l and r indicate left and right positions, respectively; see below).

We set $r(p) = n + 1$ and $f(p) = q$, and define p by $A \rightarrow g(H, A_1, A_2, \ldots, A_n)$, with $g(t_H, t_{A_1}, \ldots, t_{A_n}) = \langle \alpha_1, \ldots, \alpha_q \rangle$. Here t_H is the tuple of variables for H and each t_{A_i}, $i \in [n]$, is the tuple of variables for A_i. Each string α_i, $i \in [q]$, is specified as follows. Let v_s and v_t be the endpoints of e_i, with $v_s, v_t \in V$ and $s < t$. We define

$$\alpha_i \quad = \quad x_{A_s,e_i,l}x_{A_t,e_i,l}x_{H,e_i}x_{A_s,e_i,r}x_{A_t,e_i,r} \ .$$

Observe that whenever edge e_i impinges on vertex v_j, then the left and right strings generated by A_j and associated with e_i wrap around the string generated by H and associated with the same edge. Finally, we set $k' = q + k$.

Example 4 Given the input graph of Figure 2, our reduction constructs the LCFRS production shown in Figure 3. Figure 4 gives a visualization of how the spans in this production fit together. For each edge in the graph of Figure 2, we have a group of five spans in the production: one for the head nonterminal, and two spans for each of the two nonterminals corresponding to the edge's endpoints. □

Assume now some head-driven parsing strategy π for p. For each $i \in [n]$, we define D_i^π to be the partial parse obtained after step i in π, consisting of the merge of nonterminals $H, A_{\pi(1)}, \ldots, A_{\pi(i)}$. Consider some edge $e_j = (v_s, v_t)$. We observe that for any D_i^π that includes or excludes both nonterminals A_s and A_t, the α_j component in the definition of p is associated with a single string, and therefore contributes with a single unit to the fan-out of the partial parse. On the other hand, if D_i^π includes only one nonterminal between A_s and A_t, the α_j component is associated with two strings and contributes with two units to the fan-out of the partial parse.

We can associate with π a linear arrangement h_π of M by letting $h_\pi(v_{\pi(i)}) = i$, for each $v_i \in V$. From the above observation on the fan-out of D_i^π,

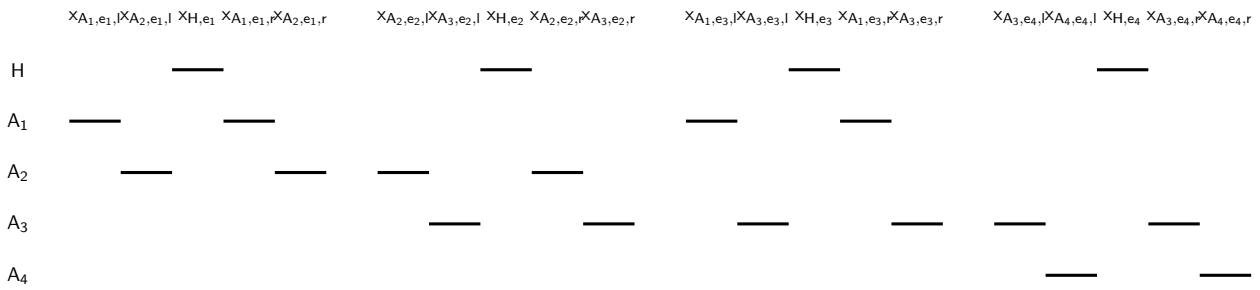


Figure 4: A visualization of how the spans for each nonterminal fit together in the left-to-right order defined by the production of Figure 3.

we have the following relation, for every $i \in [n-1]$:

$$f(D_i^\pi) \;\; = \;\; q + cw(M, h_\pi, i) \; .$$

We can then conclude that M, k is a positive instance of MIN CUT LINEAR ARRANGEMENT if and only if p, k' is a positive instance of MIN SPACE STRATEGY. This proves that MIN SPACE STRATEGY is NP-hard.

To show that MIN SPACE STRATEGY is in NP, consider a nondeterministic algorithm that, given an LCFRS production p and an integer k, guesses a parsing strategy π for p, and tests whether $f(D_i^\pi) \leq k$ for each $i \in [n]$. The algorithm accepts or rejects accordingly. Such an algorithm can clearly be implemented to run in polynomial time. ■

We now turn to the MIN TIME STRATEGY problem, restricted to head-driven parsing strategies. Recall that we are now concerned with the quantity $f_1 + f_2 + f$, where f_1 is the fan-out of some partial parse D, f_2 is the fan-out of a nonterminal A, and f is the fan out of the partial parse resulting from the merge of the two previous analyses.

We need to introduce the MODIFIED CUTWIDTH problem, which is a variant of the MIN CUT LINEAR ARRANGEMENT problem. Let $M = (V, E)$ be some graph with $|V| = n$, and let h be a linear arrangement for M. The **modified cutwidth** of M at position $i \in [n]$ and with respect to h is the number of edges crossing over the i-th vertex:

$$mcw(M, h, i) = |\{(u, v) \in E \,|\, h(u) < i < h(v)\}| \; .$$

The modified cutwidth of M is defined as

$$mcw(M) = \min_h \max_{i \in [n]} \; mcw(M, h, i) \; .$$

In the MODIFIED CUTWIDTH problem one is given as input a graph M and an integer k, and must decide whether $mcw(M) \leq k$. The MODIFIED CUTWIDTH problem has been shown to be NP-complete by Lengauer (1981). We strengthen this result below; recall that a cubic graph is a graph without self loops where each vertex has degree three.

Lemma 1 *The* MODIFIED CUTWIDTH *problem restricted to cubic graphs is NP-complete.*

PROOF The MODIFIED CUTWIDTH problem has been shown to be NP-complete when restricted to graphs of maximum degree three by Makedon et al. (1985), reducing from a graph problem known as bisection width (see also Monien and Sudborough (1988)). Specifically, the authors construct a graph G' of maximum degree three and an integer k' from an input graph $G = (V, E)$ with an even number n of vertices and an integer k, such that $mcw(G') \leq k'$ if and only if the **bisection width** $bw(G)$ of G is not greater than k, where

$$bw(G) = \min_{A,B \subseteq V} |\{(u, v) \in E \,|\, u \in A \wedge v \in B\}|$$

with $A \cap B = \emptyset$, $A \cup B = V$, and $|A| = |B|$.

The graph G' has vertices of degree two and three only, and it is based on a grid-like gadget $R(r, c)$; see Figure 5. For each vertex of G, G' includes a component $R(2n^4, 8n^4+8)$. Moreover, G' has a component called an H-shaped graph, containing left and right columns $R(3n^4, 12n^4 + 12)$ connected by a middle bar $R(2n^4, 12n^4 + 9)$; see Figure 6. From each of the n vertex components there is a sheaf of $2n^2$ edges connecting distinct degree 2 vertices in the component to $2n^2$ distinct degree 2 vertices in

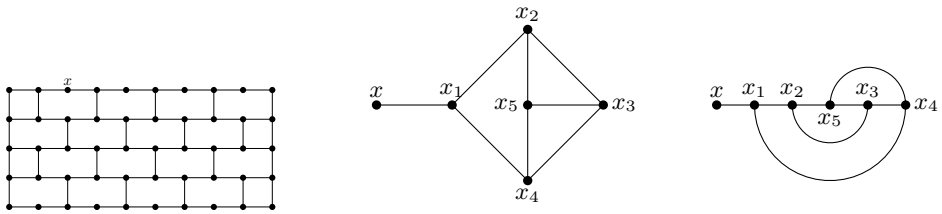


Figure 5: The $R(5, 10)$ component (left), the modification of its degree 2 vertex x (middle), and the corresponding arrangement (right).

the middle bar of the H-shaped graph. Finally, for each edge (v_i, v_j) of G there is an edge in G' connecting a degree 2 vertex in the component corresponding to the vertex v_i with a degree 2 vertex in the component corresponding to the vertex v_j. The integer k' is set to $3n^4 + n^3 + k - 1$.

Makedon et al. (1985) show that the modified cutwidth of $R(r, c)$ is $r - 1$ whenever $r \geq 3$ and $c \geq 4r + 8$. They also show that an optimal linear arrangement for G' has the form depicted in Figure 6, where half of the vertex components are to the left of the H-shaped graph and all the other vertex components are to the right. In this arrangement, the modified cutwidth is attested by the number of edges crossing over the vertices in the left and right columns of the H-shaped graph, which is equal to

$$3n^4 - 1 + \frac{n}{2}2n^2 + \gamma = 3n^4 + n^3 + \gamma - 1 \quad (2)$$

where γ denotes the number of edges connecting vertices to the left with vertices to the right of the H-shaped graph. Thus, $bw(G) \leq k$ if and only if $mcw(G') \leq k'$.

All we need to show now is how to modify the components of G' in order to make it cubic.

Modifying the vertex components All vertices x of degree 2 of the components corresponding to a vertex in G can be transformed into a vertex of degree 3 by adding five vertices $x_1, \ldots, x_5$ connected as shown in the middle bar of Figure 5. Observe that these five vertices can be positioned in the arrangement immediately after x in the order x_1, x_2, x_5, x_3, x_4 (see the right part of the figure). The resulting maximum modified cutwidth can increase by 2 in correspondence of vertex x_5. Since the vertices of these components, in the optimal arrangement, have modified cutwidth smaller than $2n^4 + n^3 + n^2$, an increase by 2 is still smaller than the maximum modified cutwidth of the entire graph, which is $3n^4 + \mathcal{O}(n^3)$.

Modifying the middle bar of the H-shaped graph The vertices of degree 2 of this part of the graph can be modified as in the previous paragraph. Indeed, in the optimal arrangement, these vertices have modified cutwidth smaller than $2n^4 + 2n^3 + n^2$, and an increase by 2 is still smaller than the maximum cutwidth of the entire graph.

Modifying the left/right columns of the H-shaped graph We replace the two copies of component $R(3n^4, 12n^4 + 12)$ with two copies of the new component $D(3n^4, 24n^4 + 16)$ shown in Figure 7, which is a cubic graph. In order to prove that relation (2) still holds, it suffices to show that the modified cutwidth of the component $D(r, c)$ is still $r - 1$ whenever $r \geq 3$ and $c = 8r + 16$.

We first observe that the linear arrangement obtained by visiting the vertices of $D(r, c)$ from top to bottom and from left to right has modified cutwidth $r - 1$. Let us now prove that, for any partition of the vertices into two subsets V_1 and V_2 with $|V_1|, |V_2| \geq 4r^2$, there exist at least r disjoint paths between vertices of V_1 and vertices of V_2. To this aim, we distinguish the following three cases.

- Any row has (at least) one vertex in V_1 and one vertex in V_2: in this case, it is easy to see there exist at least r disjoint paths between vertices of V_1 and vertices of V_2.

- There exist at least $3r$ 'mixed' columns, that is, columns with (at least) one vertex in V_1 and one vertex in V_2. Again, it is easy to see that there exist at least r disjoint paths between vertices

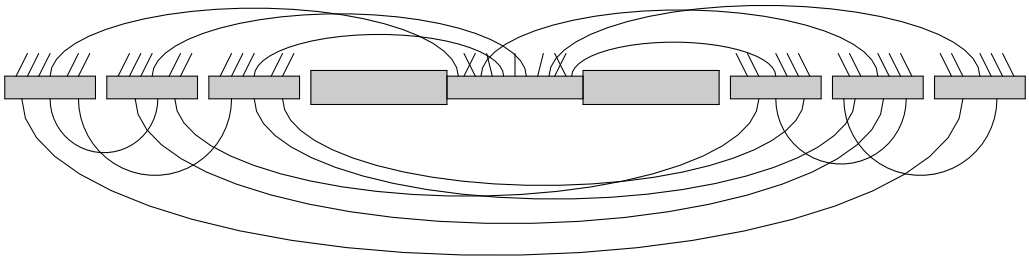


Figure 6: The optimal arrangement of G'.

of V_1 and vertices of V_2 (at least one path every three columns).

- The previous two cases do not apply. Hence, there exists a row entirely formed by vertices of V_1 (or, equivalently, of V_2). The worst case is when this row is the smallest one, that is, the one with $\frac{(c-3-1)}{2} + 1 = 4r + 7$ vertices. Since at most $3r - 1$ columns are mixed, we have that at most $(3r - 1)(r - 2) = 3r^2 - 7r + 2$ vertices of V_2 are on these mixed columns. Since $|V_2| \geq 4r^2$, this implies that at least r columns are fully contained in V_2. On the other hand, at least $4r+7-(3r-1) = r+8$ columns are fully contained in V_1. If the V_1-columns interleave with the V_2-columns, then there exist at least $2(r-1)$ disjoint paths between vertices of V_1 and vertices of V_2. Otherwise, all the V_1-columns precede or follow all the V_2-columns (this corresponds to the optimal arrangement): in this case, there are r disjoint paths between vertices of V_1 and vertices of V_2.

Observe now that any linear arrangement partitions the set of vertices in $D(r, c)$ into the sets V_1, consisting of the first $4r^2$ vertices in the arrangement, and V_2, consisting of all the remaining vertices. Since there are r disjoint paths connecting V_1 and V_2, there must be at least $r-1$ edges passing over every vertex in the arrangement which is assigned to a position between the $(4r^2 + 1)$-th and the position $4r^2 + 1$ from the right end of the arrangement: thus, the modified cutwidth of any linear arrangement of the vertices of $D(r, c)$ is at least $r - 1$.

We can then conclude that the original proof of Makedon et al. (1985) still applies, according to relation (2). ■

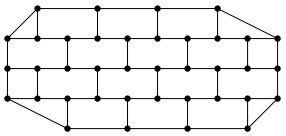


Figure 7: The $D(5, 10)$ component.

We can now reduce from the MODIFIED CUTWIDTH problem for cubic graphs to the MIN TIME STRATEGY problem restricted to head-driven parsing strategies.

Theorem 2 *The* MIN TIME STRATEGY *problem restricted to head-driven parsing strategies is NP-complete.*

PROOF We consider hardness first. Let M and k be an input instance of the MODIFIED CUTWIDTH problem restricted to cubic graphs, where $M = (V, E)$ and $V = \{v_1, \ldots, v_n\}$. We construct an LCFRS production p exactly as in the proof of Theorem 1, with rhs nonterminals $H, A_1, \ldots, A_n$. We also set $k' = 2 \cdot k + 2 \cdot |E| + 9$.

Assume now some head-driven parsing strategy π for p. After parsing step $i \in [n]$, we have a partial parse D_i^π consisting of the merge of nonterminals $H, A_{\pi(1)}, \ldots, A_{\pi(i)}$. We write $tc(p, \pi, i)$ to denote the exponent of the time complexity due to step i. As already mentioned, this quantity is defined as the sum of the fan-out of the two antecedents involved in the parsing step and the fan-out of its result:

$$tc(p, \pi, i) \quad = \quad f(D_{i-1}^\pi) + f(A_{\pi(i)}) + f(D_i^\pi) \; .$$

Again, we associate with π a linear arrangement h_π of M by letting $h_\pi(v_{\pi(i)}) = i$, for each $v_i \in V$. As in the proof of Theorem 1, the fan-out of D_i^π is then related to the cutwidth of the linear arrange-

ment h_π of M at position i by

$$f(D_i^\pi) \;=\; |E| + cw(M, h_\pi, i) \; .$$

From the proof of Theorem 1, the fan-out of nonterminal $A_{\pi(i)}$ is twice the degree of vertex $v_{\pi(i)}$, denoted by $|E(v_{\pi(i)})|$. We can then rewrite the above equation in terms of our graph M:

$$\begin{aligned} tc(p, \pi, i) = 2 \cdot |E| + cw(M, h_\pi, i-1) + \\ + 2 \cdot |E(v_{\pi(i)})| + cw(M, h_\pi, i) \; . \end{aligned}$$

The following general relation between cutwidth and modified cutwidth is rather intuitive:

$$\begin{aligned} mcw(M, h_\pi, i) = \frac{1}{2} \cdot [cw(M, h_\pi, i-1) + \\ - |E(v_{\pi(i)})| + cw(M, h_\pi, i)] \; . \end{aligned}$$

Combining the two equations above we obtain:

$$\begin{aligned} tc(p, \pi, i) \;=\; 2 \cdot |E| + 3 \cdot |E(v_{\pi(i)})| + \\ + 2 \cdot mcw(M, h_\pi, i) \; . \end{aligned}$$

Because we are restricting M to the class of cubic graphs, we can write:

$$tc(p, \pi, i) \;=\; 2 \cdot |E| + 9 + 2 \cdot mcw(M, h_\pi, i) \; .$$

We can thus conclude that there exists a head-driven parsing strategy for p with time complexity not greater than $2 \cdot |E| + 9 + 2 \cdot k = k'$ if and only if $mcw(M) \leq k$.

The membership of MODIFIED CUTWIDTH in NP follows from an argument similar to the one in the proof of Theorem 1. ∎

We have established the NP-completeness of both the MIN SPACE STRATEGY and the MIN TIME STRATEGY decision problems. It is now easy to see that the problem of finding a space- or time-optimal parsing strategy for a LCFRS production is NP-hard as well, and thus cannot be solved in polynomial (deterministic) time unless P = NP.

4 Concluding remarks

Head-driven strategies are important in parsing based on LCFRSs, both in order to allow statistical modeling of head-modifier dependencies and in order to generalize the Markovization of CFG parsers to parsers with discontinuous spans. However, there are $n!$ possible head-driven strategies for an LCFRS production with a head and n modifiers. Choosing among these possible strategies affects both the time and the space complexity of parsing. In this paper we have shown that optimizing the choice according to either metric is NP-hard. To our knowledge, our results are the first NP-hardness results for a grammar factorization problem.

SCFGs and STAGs are specific instances of LCFRSs. Grammar factorization for synchronous models is an important component of current machine translation systems (Zhang et al., 2006), and algorithms for factorization have been studied by Gildea et al. (2006) for SCFGs and by Nesson et al. (2008) for STAGs. These algorithms do not result in what we refer as head-driven strategies, although, as machine translation systems improve, lexicalized rules may become important in this setting as well. However, the results we have presented in this paper do not carry over to the above mentioned synchronous models, since the fan-out of these models is bounded by two, while in our reductions in Section 3 we freely use unbounded values for this parameter. Thus the computational complexity of optimizing the choice of the parsing strategy for SCFGs is still an open problem.

Finally, our results for LCFRSs only apply when we restrict ourselves to head-driven strategies. This is in contrast to the findings of Gildea (2011), which show that, for unrestricted parsing strategies, a polynomial time algorithm for minimizing parsing complexity would imply an improved approximation algorithm for finding the treewidth of general graphs. Our result is stronger, in that it shows strict NP-hardness, but also weaker, in that it applies only to head-driven strategies. Whether NP-hardness can be shown for unrestricted parsing strategies is an important question for future work.

Acknowledgments

The first and third authors are partially supported from the Italian PRIN project DISCO. The second author is partially supported by NSF grants IIS-0546554 and IIS-0910611.

References

Ashok K. Chandra and Philip M. Merlin. 1977. Optimal implementation of conjunctive queries in relational data bases. In *Proc. ninth annual ACM symposium on Theory of computing*, STOC '77, pages 77–90.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. 35th Annual Conference of the Association for Computational Linguistics (ACL-97)*, pages 16–23.

F. Gavril. 1977. Some NP-complete problems on graphs. In *Proc. 11th Conf. on Information Sciences and Systems*, pages 91–95.

Daniel Gildea and Daniel Štefankovič. 2007. Worst-case synchronous grammar rules. In *Proc. 2007 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-07)*, pages 147–154, Rochester, NY.

Daniel Gildea, Giorgio Satta, and Hao Zhang. 2006. Factoring synchronous grammars by sorting. In *Proc. International Conference on Computational Linguistics/Association for Computational Linguistics (COLING/ACL-06) Poster Session*, pages 279–286.

Daniel Gildea. 2010. Optimal parsing strategies for Linear Context-Free Rewriting Systems. In *Proc. 2010 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-10)*, pages 769–776.

Daniel Gildea. 2011. Grammar factorization by tree decomposition. *Computational Linguistics*, 37(1):231–248.

Carlos Gómez-Rodríguez, Marco Kuhlmann, Giorgio Satta, and David Weir. 2009. Optimal reduction of rule length in Linear Context-Free Rewriting Systems. In *Proc. 2009 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-09)*, pages 539–547.

Carlos Gómez-Rodríguez, Marco Kuhlmann, and Giorgio Satta. 2010. Efficient parsing of well-nested linear context-free rewriting systems. In *Proc. 2010 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-10)*, pages 276–284, Los Angeles, California.

John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA.

Liang Huang, Hao Zhang, Daniel Gildea, and Kevin Knight. 2009. Binarization of synchronous context-free grammars. *Computational Linguistics*, 35(4):559–595.

Laura Kallmeyer and Wolfgang Maier. 2010. Data-driven parsing with probabilistic linear context-free rewriting systems. In *Proc. 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 537–545.

Marco Kuhlmann and Giorgio Satta. 2009. Treebank grammar techniques for non-projective dependency parsing. In *Proc. 12th Conference of the European Chapter of the ACL (EACL-09)*, pages 478–486.

Thomas Lengauer. 1981. Black-white pebbles and graph separation. *Acta Informatica*, 16:465–475.

Wolfgang Maier and Anders Søgaard. 2008. Treebanks and mild context-sensitivity. In Philippe de Groote, editor, *Proc. 13th Conference on Formal Grammar (FG-2008)*, pages 61–76, Hamburg, Germany. CSLI Publications.

F. S. Makedon, C. H. Papadimitriou, and I. H. Sudborough. 1985. Topological bandwidth. *SIAM J. Alg. Disc. Meth.*, 6(3):418–444.

B. Monien and I.H. Sudborough. 1988. Min cut is NP-complete for edge weighted trees. *Theor. Comput. Sci.*, 58:209–229.

Rebecca Nesson, Giorgio Satta, and Stuart M. Shieber. 2008. Optimal k-arization of synchronous tree adjoining grammar. In *Proc. 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pages 604–612.

Owen Rambow and Giorgio Satta. 1999. Independent parallelism in finite copying parallel rewriting systems. *Theor. Comput. Sci.*, 223(1-2):87–120.

Benoît Sagot and Giorgio Satta. 2010. Optimal rank reduction for linear context-free rewriting systems with fan-out two. In *Proc. 48th Annual Meeting of the Association for Computational Linguistics*, pages 525–533, Uppsala, Sweden.

Giorgio Satta and Enoch Peserico. 2005. Some computational complexity results for synchronous context-free grammars. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 803–810, Vancouver, Canada.

H. Seki, T. Matsumura, M. Fujii, and T. Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229.

Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *The Journal of Logic Programming*, 24(1-2):3–36.

K. Vijay-Shankar, D. L. Weir, and A. K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proc. 25th Annual Conference of the Association for Computational Linguistics (ACL-87)*, pages 104–111.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proc. 2006 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-06)*, pages 256–263.

Prefix Probability for Probabilistic Synchronous Context-Free Grammars

Mark-Jan Nederhof
School of Computer Science
University of St Andrews
North Haugh, St Andrews, Fife
KY16 9SX
United Kingdom
markjan.nederhof@googlemail.com

Giorgio Satta
Dept. of Information Engineering
University of Padua
via Gradenigo, 6/A
I-35131 Padova
Italy
satta@dei.unipd.it

Abstract

We present a method for the computation of prefix probabilities for synchronous context-free grammars. Our framework is fairly general and relies on the combination of a simple, novel grammar transformation and standard techniques to bring grammars into normal forms.

1 Introduction

Within the area of statistical machine translation, there has been a growing interest in so-called syntax-based translation models, that is, models that define mappings between languages through hierarchical sentence structures. Several such statistical models that have been investigated in the literature are based on synchronous rewriting or tree transduction. Probabilistic synchronous context-free grammars (PSCFGs) are one among the most popular examples of such models. PSCFGs subsume several syntax-based statistical translation models, as for instance the stochastic inversion transduction grammars of Wu (1997), the statistical model used by the Hiero system of Chiang (2007), and systems which extract rules from parsed text, as in Galley et al. (2004).

Despite the widespread usage of models related to PSCFGs, our theoretical understanding of this class is quite limited. In contrast to the closely related class of probabilistic context-free grammars, a syntax model for which several interesting mathematical and statistical properties have been investigated, as for instance by Chi (1999), many theoretical problems are still unsolved for the class of PSCFGs.

This paper considers a parsing problem that is well understood for probabilistic context-free grammars but that has never been investigated in the context of PSCFGs, viz. the computation of prefix probabilities. In the case of a probabilistic context-free grammar, this problem is defined as follows. We are asked to compute the probability that a sentence generated by our model starts with a prefix string v given as input. This quantity is defined as the (possibly infinite) sum of the probabilities of all strings of the form vw, for any string w over the alphabet of the model. This problem has been studied by Jelinek and Lafferty (1991) and by Stolcke (1995). Prefix probabilities can be used to compute probability distributions for the next word or part-of-speech. This has applications in incremental processing of text or speech from left to right; see again (Jelinek and Lafferty, 1991). Prefix probabilities can also be exploited in speech understanding systems to score partial hypotheses in beam search (Corazza et al., 1991).

This paper investigates the problem of computing prefix probabilities for PSCFGs. In this context, a pair of strings v_1 and v_2 is given as input, and we are asked to compute the probability that any string in the source language starting with prefix v_1 is translated into any string in the target language starting with prefix v_2. This probability is more precisely defined as the sum of the probabilities of translation pairs of the form $[v_1w_1, v_2w_2]$, for any strings w_1 and w_2.

A special case of prefix probability for PSCFGs is the right prefix probability. This is defined as the probability that some (complete) input string w in the source language is translated into a string in the target language starting with an input prefix v.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 460–469,
Portland, Oregon, June 19-24, 2011.

Prefix probabilities and right prefix probabilities for PSCFGs can be exploited to compute probability distributions for the next word or part-of-speech in left-to-right incremental translation, essentially in the same way as described by Jelinek and Lafferty (1991) for probabilistic context-free grammars, as discussed later in this paper.

Our solution to the problem of computing prefix probabilities is formulated in quite different terms from the solutions by Jelinek and Lafferty (1991) and by Stolcke (1995) for probabilistic context-free grammars. In this paper we reduce the computation of prefix probabilities for PSCFGs to the computation of inside probabilities under the same model. Computation of inside probabilities for PSCFGs is a well-known problem that can be solved using off-the-shelf algorithms that extend basic parsing algorithms. Our reduction is a novel grammar transformation, and the proof of correctness proceeds by fairly conventional techniques from formal language theory, relying on the correctness of standard methods for the computation of inside probabilities for PSCFG. This contrasts with the techniques proposed by Jelinek and Lafferty (1991) and by Stolcke (1995), which are extensions of parsing algorithms for probabilistic context-free grammars, and require considerably more involved proofs of correctness.

Our method for computing the prefix probabilities for PSCFGs runs in exponential time, since that is the running time of existing methods for computing the inside probabilities for PSCFGs. It is unlikely this can be improved, because the recognition problem for PSCFG is NP-complete, as established by Satta and Peserico (2005), and there is a straightforward reduction from the recognition problem for PSCFGs to the problem of computing the prefix probabilities for PSCFGs.

2 Definitions

In this section we introduce basic definitions related to synchronous context-free grammars and their probabilistic extension; our notation follows Satta and Peserico (2005).

Let N and Σ be sets of nonterminal and terminal symbols, respectively. In what follows we need to represent bijections between the occurrences of nonterminals in two strings over $N \cup \Sigma$. This is realized by annotating nonterminals with **indices** from an infinite set. We define $\mathcal{I}(N) = \{A^{\boxed{t}} \mid A \in N,\ t \in \mathbb{N}\}$ and $V_I = \mathcal{I}(N) \cup \Sigma$. For a string $\gamma \in V_I^*$, we write $\mathsf{index}(\gamma)$ to denote the set of all indices that appear in symbols in γ.

Two strings $\gamma_1, \gamma_2 \in V_I^*$ are **synchronous** if each index from $\mathbb{N}$ occurs at most once in γ_1 and at most once in γ_2, and $\mathsf{index}(\gamma_1) = \mathsf{index}(\gamma_2)$. Therefore γ_1, γ_2 have the general form:

$$\gamma_1 = u_{10} A_{11}^{\boxed{t_1}} u_{11} A_{12}^{\boxed{t_2}} u_{12} \cdots u_{1r-1} A_{1r}^{\boxed{t_r}} u_{1r}$$
$$\gamma_2 = u_{20} A_{21}^{\boxed{t_{\pi(1)}}} u_{21} A_{22}^{\boxed{t_{\pi(2)}}} u_{22} \cdots u_{2r-1} A_{2r}^{\boxed{t_{\pi(r)}}} u_{2r}$$

where $r \geq 0$, $u_{1i}, u_{2i} \in \Sigma^*$, $A_{1i}^{\boxed{t_i}}, A_{2i}^{\boxed{t_{\pi(i)}}} \in \mathcal{I}(N)$, $t_i \neq t_j$ for $i \neq j$, and π is a permutation of the set $\{1, \ldots, r\}$.

A **synchronous context-free grammar** (SCFG) is a tuple $G = (N, \Sigma, P, S)$, where N and Σ are finite, disjoint sets of nonterminal and terminal symbols, respectively, $S \in N$ is the start symbol and P is a finite set of **synchronous rules**. Each synchronous rule has the form $s : [A_1 \rightarrow \alpha_1,\ \ A_2 \rightarrow \alpha_2]$, where $A_1, A_2 \in N$ and where $\alpha_1, \alpha_2 \in V_I^*$ are synchronous strings. The symbol s is the label of the rule, and each rule is uniquely identified by its label. For technical reasons, we allow the existence of multiple rules that are identical apart from their labels. We refer to $A_1 \rightarrow \alpha_1$ and $A_2 \rightarrow \alpha_2$, respectively, as the left and right **components** of rule s.

Example 1 The following synchronous rules implicitly define a SCFG:

$$\begin{aligned}
s_1 &: \ [S \rightarrow A^{\boxed{1}}\ B^{\boxed{2}},\ S \rightarrow B^{\boxed{2}}\ A^{\boxed{1}}] \\
s_2 &: \ [A \rightarrow aA^{\boxed{1}}b,\ A \rightarrow bA^{\boxed{1}}a] \\
s_3 &: \ [A \rightarrow ab,\ A \rightarrow ba] \\
s_4 &: \ [B \rightarrow cB^{\boxed{1}}d,\ B \rightarrow dB^{\boxed{1}}c] \\
s_5 &: \ [B \rightarrow cd,\ B \rightarrow dc]
\end{aligned}$$

□

In each step of the derivation process of a SCFG G, two nonterminals with the same index in a pair of synchronous strings are rewritten by a synchronous rule. This is done in such a way that the result is once more a pair of synchronous strings. An auxiliary notion is that of **reindexing**, which is an injective function f from $\mathbb{N}$ to $\mathbb{N}$. We extend f to V_I by letting $f(A^{\boxed{t}}) = A^{\boxed{f(t)}}$ for $A^{\boxed{t}} \in \mathcal{I}(N)$ and $f(a) = a$ for $a \in \Sigma$. We also extend f to strings in V_I^* by

letting $f(\varepsilon) = \varepsilon$ and $f(X\gamma) = f(X)f(\gamma)$, for each $X \in V_I$ and $\gamma \in V_I^*$.

Let γ_1, γ_2 be synchronous strings in V_I^*. The **derive** relation $[\gamma_1,\ \gamma_2] \Rightarrow_G [\delta_1,\ \delta_2]$ holds whenever there exist an index t in $\mathsf{index}(\gamma_1) = \mathsf{index}(\gamma_2)$, a synchronous rule $s : [A_1 \rightarrow \alpha_1,\ A_2 \rightarrow \alpha_2]$ in P and some reindexing f such that:

(i) $\mathsf{index}(f(\alpha_1)) \cap (\mathsf{index}(\gamma_1) \setminus \{t\}) = \emptyset$;

(ii) $\gamma_1 = \gamma_1' A_1^{\boxed{t}} \gamma_1''$, $\gamma_2 = \gamma_2' A_2^{\boxed{t}} \gamma_2''$; and

(iii) $\delta_1 = \gamma_1' f(\alpha_1) \gamma_1''$, $\delta_2 = \gamma_2' f(\alpha_2) \gamma_2''$.

We also write $[\gamma_1,\ \gamma_2] \Rightarrow_G^s [\delta_1,\ \delta_2]$ to explicitly indicate that the derive relation holds through rule s.

Note that δ_1, δ_2 above are guaranteed to be synchronous strings, because α_1 and α_2 are synchronous strings and because of (i) above. Note also that, for a given pair $[\gamma_1,\ \gamma_2]$ of synchronous strings, an index t and a rule s, there may be infinitely many choices of reindexing f such that the above constraints are satisfied. In this paper we will not further specify the choice of f.

We say the pair $[A_1, A_2]$ of nonterminals is **linked** (in G) if there is a rule of the form $s : [A_1 \rightarrow \alpha_1,\ A_2 \rightarrow \alpha_2]$. The set of linked nonterminal pairs is denoted by $N^{[2]}$.

A derivation is a sequence $\sigma = s_1 s_2 \cdots s_d$ of synchronous rules $s_i \in P$ with $d \geq 0$ ($\sigma = \varepsilon$ for $d = 0$) such that $[\gamma_{1i-1},\ \gamma_{2i-1}] \Rightarrow_G^{s_i} [\gamma_{1i},\ \gamma_{2i}]$ for every i with $1 \leq i \leq d$ and synchronous strings $[\gamma_{1i},\ \gamma_{2i}]$ with $0 \leq i \leq d$. Throughout this paper, we always implicitly assume some canonical form for derivations in G, by demanding for instance that each step rewrites a pair of nonterminal occurrences of which the first is leftmost in the left component. When we want to focus on the specific synchronous strings being derived, we also write derivations in the form $[\gamma_{10},\ \gamma_{20}] \Rightarrow_G^{\sigma} [\gamma_{1d},\ \gamma_{2d}]$, and we write $[\gamma_{10},\ \gamma_{20}] \Rightarrow_G^* [\gamma_{1d},\ \gamma_{2d}]$ when σ is not further specified. The **translation** generated by a SCFG G is defined as:

$$T(G) = \{[w_1,\ w_2] \mid [S^{\boxed{1}},\ S^{\boxed{1}}] \Rightarrow_G^* [w_1,\ w_2],\ w_1, w_2 \in \Sigma^*\}$$

For $w_1, w_2 \in \Sigma^*$, we write $D(G, [w_1, w_2])$ to denote the set of all (canonical) derivations σ such that $[S^{\boxed{1}},\ S^{\boxed{1}}] \Rightarrow_G^{\sigma} [w_1,\ w_2]$.

Analogously to standard terminology for context-free grammars, we call a SCFG **reduced** if every rule occurs in at least one derivation $\sigma \in D(G, [w_1, w_2])$, for some $w_1, w_2 \in \Sigma^*$. We assume without loss of generality that the start symbol S does not occur in the right-hand side of either component of any rule.

Example 2 Consider the SCFG G from example 1. The following is a canonical derivation in G, since it is always the leftmost nonterminal occurrence in the left component that is involved in a derivation step:

$$\begin{aligned}
[S^{\boxed{1}},\ S^{\boxed{1}}] &\Rightarrow_G [A^{\boxed{1}}B^{\boxed{2}}, B^{\boxed{2}}A^{\boxed{1}}] \\
&\Rightarrow_G [aA^{\boxed{3}}bB^{\boxed{2}}, B^{\boxed{2}}bA^{\boxed{3}}a] \\
&\Rightarrow_G [aaA^{\boxed{4}}bbB^{\boxed{2}}, B^{\boxed{2}}bbA^{\boxed{4}}aa] \\
&\Rightarrow_G [aaabbbB^{\boxed{2}}, B^{\boxed{2}}bbbaaa] \\
&\Rightarrow_G [aaabbbcB^{\boxed{5}}d, dB^{\boxed{5}}cbbbaaa] \\
&\Rightarrow_G [aaabbbccdd, ddccbbbaaa]
\end{aligned}$$

It is not difficult to see that the generated translation is $T(G) = \{[a^p b^p c^q d^q,\ d^q c^q b^p a^p] \mid p, q \geq 1\}$. □

The size of a synchronous rule $s : [A_1 \rightarrow \alpha_1,\ A_2 \rightarrow \alpha_2]$, is defined as $|s| = |A_1 \alpha_1 A_2 \alpha_2|$. The size of G is defined as $|G| = \sum_{s \in P} |s|$.

A **probabilistic** SCFG (PSCFG) is a pair $\mathcal{G} = (G, p_G)$ where $G = (N, \Sigma, P, S)$ is a SCFG and p_G is a function from P to real numbers in $[0, 1]$. We say that $\mathcal{G}$ is **proper** if for each pair $[A_1, A_2] \in N^{[2]}$ we have:

$$\sum_{s:[A_1 \rightarrow \alpha_1,\ A_2 \rightarrow \alpha_2]} p_G(s) = 1$$

Intuitively, properness ensures that where a pair of nonterminals in two synchronous strings can be rewritten, there is a probability distribution over the applicable rules.

For a (canonical) derivation $\sigma = s_1 s_2 \cdots s_d$, we define $p_G(\sigma) = \prod_{i=1}^{d} p_G(s_i)$. For $w_1, w_2 \in \Sigma^*$, we also define:

$$p_G([w_1, w_2]) = \sum_{\sigma \in D(G, [w_1, w_2])} p_G(\sigma) \qquad (1)$$

We say a PSCFG is **consistent** if p_G defines a probability distribution over the translation, or formally:

$$\sum_{w_1, w_2} p_G([w_1, w_2]) = 1$$

If the grammar is reduced, proper and consistent, then also:

$$\sum_{\substack{w_1,w_2\in\Sigma^*,\ \sigma\in P^* \\ \text{s.t. } [A_1^{\boxed{1}},\ A_2^{\boxed{1}}]\Rightarrow_G^{\sigma}[w_1,\ w_2]}} p_G(\sigma) \quad = \quad 1$$

for every pair $[A_1, A_2] \in N^{[2]}$. The proof is identical to that of the corresponding fact for probabilistic context-free grammars.

3 Effective PSCFG parsing

If $w = a_1 \cdots a_n$ then the expression $w[i, j]$, with $0 \leq i \leq j \leq n$, denotes the substring $a_{i+1} \cdots a_j$ (if $i = j$ then $w[i, j] = \varepsilon$). In this section, we assume the input is the pair $[w_1, w_2]$ of terminal strings. The task of a **recognizer** for SCFG G is to decide whether $[w_1, w_2] \in T(G)$.

We present a general algorithm for solving the above problem in terms of the specification of a deduction system, following Shieber et al. (1995). The items that are constructed by the system have the form $[m_1, A_1, m_1';\ m_2, A_2, m_2']$, where $[A_1, A_2] \in N^{[2]}$ and where m_1, m_1', m_2, m_2' are non-negative integers such that $0 \leq m_1 \leq m_1' \leq |w_1|$ and $0 \leq m_2 \leq m_2' \leq |w_2|$. Such an item can be derived by the deduction system if and only if:

$$[A_1^{\boxed{1}},\ A_2^{\boxed{1}}] \quad \Rightarrow_G^* \quad [w_1[m_1, m_1'],\ w_2[m_2, m_2']]$$

The deduction system has one inference rule, shown in figure 1. One of its side conditions has a synchronous rule in P of the form:

$$\begin{aligned} s : [A_1 \rightarrow u_{10} A_{11}^{\boxed{t_1}} u_{11} \cdots u_{1r-1} A_{1r}^{\boxed{t_r}} u_{1r}, \\ A_2 \rightarrow u_{20} A_{21}^{\boxed{t_{\pi(1)}}} u_{21} \cdots u_{2r-1} A_{2r}^{\boxed{t_{\pi(r)}}} u_{2r}] \end{aligned} \quad (2)$$

Observe that, in the right-hand side of the two rule components above, nonterminals A_{1i} and $A_{2\pi^{-1}(i)}$, $1 \leq i \leq r$, have both the same index. More precisely, A_{1i} has index t_i and $A_{2\pi^{-1}(i)}$ has index $t_{i'}$ with $i' = \pi(\pi^{-1}(i)) = i$. Thus the nonterminals in each antecedent item in figure 1 form a linked pair.

We now turn to a computational analysis of the above algorithm. In the inference rule in figure 1 there are $2(r + 1)$ variables that can be bound to positions in w_1, and as many that can be bound to positions in w_2. However, the side conditions imply $m_{ij}' = m_{ij} + |u_{ij}|$, for $i \in \{1, 2\}$ and $0 \leq j \leq r$, and therefore the number of free variables is only $r + 1$ for each component. By standard complexity analysis of deduction systems, for example following McAllester (2002), the time complexity of a straightforward implementation of the recognition algorithm is $\mathcal{O}(|P| \cdot |w_1|^{r_{\max}+1} \cdot |w_2|^{r_{\max}+1})$, where $r_{\max}$ is the maximum number of right-hand side nonterminals in either component of a synchronous rule. The algorithm therefore runs in exponential time, when the grammar G is considered as part of the input. Such computational behavior seems unavoidable, since the recognition problem for SCFG is NP-complete, as reported by Satta and Peserico (2005). See also Gildea and Stefankovic (2007) and Hopkins and Langmead (2010) for further analysis of the upper bound above.

The recognition algorithm above can easily be turned into a parsing algorithm by letting an implementation keep track of which items were derived from which other items, as instantiations of the consequent and the antecedents, respectively, of the inference rule in figure 1.

A probabilistic parsing algorithm that computes $p_G([w_1, w_2])$, defined in (1), can also be obtained from the recognition algorithm above, by associating each item with a probability. To explain the basic idea, let us first assume that each item can be inferred in finitely many ways by the inference rule in figure 1. Each instantiation of the inference rule should be associated with a term that is computed by multiplying the probability of the involved rule s and the product of all probabilities previously associated with the instantiations of the antecedents. The probability associated with an item is then computed as the sum of each term resulting from some instantiation of an inference rule deriving that item. This is a generalization to PSCFG of the inside algorithm defined for probabilistic context-free grammars (Manning and Schütze, 1999), and we can show that the probability associated with item $[0, S, |w_1|\,;\ \ 0, S, |w_2|]$ provides the desired value $p_G([w_1, w_2])$. We refer to the procedure sketched above as the **inside** algorithm for PSCFGs.

However, this simple procedure fails if there are cyclic dependencies, whereby the derivation of an item involves a proper subderivation of the same item. Cyclic dependencies can be excluded if it can

$$\frac{\begin{array}{c}[m'_{10}, A_{11}, m_{11};\ m'_{2\pi^{-1}(1)-1}, A_{2\pi^{-1}(1)}, m_{2\pi^{-1}(1)}] \\ \vdots \\ [m'_{1r-1}, A_{1r}, m_{1r};\ m'_{2\pi^{-1}(r)-1}, A_{2\pi^{-1}(r)}, m_{2\pi^{-1}(r)}]\end{array}}{[m_{10}, A_1, m'_{1r};\ m_{20}, A_2, m'_{2r}]} \quad \begin{cases} s{:}[A_1 \rightarrow u_{10} A_{11}^{\boxed{t_1}} u_{11} \cdots u_{1r-1} A_{1r}^{\boxed{t_r}} u_{1r}, \\ \quad A_2 \rightarrow u_{20} A_{21}^{\boxed{t_{\pi(1)}}} u_{21} \cdots u_{2r-1} A_{2r}^{\boxed{t_{\pi(r)}}} u_{2r}] \in P, \\ w_1[m_{10}, m'_{10}] = u_{10}, \quad w_2[m_{20}, m'_{20}] = u_{20}, \\ \quad \vdots \qquad\qquad\qquad \vdots \\ w_1[m_{1r}, m'_{1r}] = u_{1r}, \quad w_2[m_{2r}, m'_{2r}] = u_{2r} \end{cases}$$

Figure 1: SCFG recognition, by a deduction system consisting of a single inference rule.

be guaranteed that, in figure 1, $m'_{1r} - m_{10}$ is greater than $m_{1j} - m'_{1j-1}$ for each j ($1 \leq j \leq r$), or $m'_{2r} - m_{20}$ is greater than $m_{2j} - m'_{2j-1}$ for each j ($1 \leq j \leq r$).

Consider again a synchronous rule s of the form in (2). We say s is an **epsilon** rule if $r = 0$ and $u_{10} = u_{20} = \epsilon$. We say s is a **unit** rule if $r = 1$ and $u_{10} = u_{11} = u_{20} = u_{21} = \epsilon$. Similarly to context-free grammars, absence of epsilon rules and unit rules guarantees that there are no cyclic dependencies between items and in this case the inside algorithm correctly computes $p_G([w_1, w_2])$.

Epsilon rules can be eliminated from PSCFGs by a grammar transformation that is very similar to the transformation eliminating epsilon rules from a probabilistic context-free grammar (Abney et al., 1999). This is sketched in what follows. We first compute the set of all **nullable** linked pairs of nonterminals of the underlying SCFG, that is, the set of all $[A_1, A_2] \in N^{[2]}$ such that $[A_1^{\boxed{1}},\ A_2^{\boxed{1}}] \Rightarrow_G^* [\varepsilon,\ \varepsilon]$. This can be done in linear time $\mathcal{O}(|G|)$ using essentially the same algorithm that identifies nullable nonterminals in a context-free grammar, as presented for instance by Sippu and Soisalon-Soininen (1988).

Next, we identify all occurrences of nullable pairs $[A_1, A_2]$ in the right-hand side components of a rule s, such that A_1 and A_2 have the same index. For every possible choice of a subset $\mathcal{U}$ of these occurrences, we add to our grammar a new rule $s_{\mathcal{U}}$ constructed by omitting all of the nullable occurrences in $\mathcal{U}$. The probability of $s_{\mathcal{U}}$ is computed as the probability of s multiplied by terms of the form:

$$\sum_{\sigma \text{ s.t. } [A_1^{\boxed{1}}, A_2^{\boxed{1}}] \Rightarrow_G^{\sigma} [\varepsilon,\ \varepsilon]} p_G(\sigma) \tag{3}$$

for every pair $[A_1, A_2]$ in $\mathcal{U}$. After adding these extra rules, which in effect circumvents the use of epsilon-generating subderivations, we can safely remove all epsilon rules, with the only exception of a possible rule of the form $[S \rightarrow \epsilon, S \rightarrow \epsilon]$. The translation and the associated probability distribution in the resulting grammar will be the same as those in the source grammar.

One problem with the above construction is that we have to create new synchronous rules $s_{\mathcal{U}}$ for each possible choice of subset $\mathcal{U}$. In the worst case, this may result in an exponential blow-up of the source grammar. In the case of context-free grammars, this is usually circumvented by casting the rules in binary form prior to epsilon rule elimination. However, this is not possible in our case, since SCFGs do not allow normal forms with a constant bound on the length of the right-hand side of each component. This follows from a result due to Aho and Ullman (1969) for a formalism called syntax directed translation schemata, which is a syntactic variant of SCFGs.

An additional complication with our construction is that finding any of the values in (3) may involve solving a system of non-linear equations, similarly to the case of probabilistic context-free grammars; see again Abney et al. (1999), and Stolcke (1995). Approximate solution of such systems might take exponential time, as pointed out by Kiefer et al. (2007).

Notwithstanding the worst cases mentioned above, there is a special case that can be easily dealt with. Assume that, for each nullable pair $[A_1, A_2]$ in G we have that $[A_1^{\boxed{1}},\ A_2^{\boxed{1}}] \Rightarrow_G^* [w_1,\ w_2]$ does not hold for any w_1 and w_2 with $w_1 \neq \varepsilon$ or $w_2 \neq \varepsilon$. Then each of the values in (3) is guaranteed to be 1, and furthermore we can remove the instances of the nullable pairs in the source rule s all at the same time. This means that the overall construction of

elimination of nullable rules from G can be implemented in linear time $|G|$. It is this special case that we will encounter in section 4.

After elimination of epsilon rules, one can eliminate unit rules. We define $C^{\text{unit}}([A_1, A_2], [B_1, B_2])$ as the sum of the probabilities of all derivations deriving $[B_1, B_2]$ from $[A_1, A_2]$ with arbitrary indices, or more precisely:

$$\sum_{\substack{\sigma \in P^* \text{ s.t. } \exists t \in \mathbb{N}, \\ [A_1^{\boxed{1}}, A_2^{\boxed{1}}] \Rightarrow_G^{\sigma} [B_1^{\boxed{t}}, B_2^{\boxed{t}}]}} p_G(\sigma)$$

Note that $[A_1, A_2]$ may be equal to $[B_1, B_2]$ and σ may be ε, in which case $C^{\text{unit}}([A_1, A_2], [B_1, B_2])$ is at least 1, but it may be larger if there are unit rules. Therefore $C^{\text{unit}}([A_1, A_2], [B_1, B_2])$ should *not* be seen as a probability.

Consider a pair $[A_1, A_2] \in N^{[2]}$ and let all unit rules with left-hand sides A_1 and A_2 be:

$$\begin{aligned} s_1 &: [A_1, A_2] \rightarrow [A_{11}^{\boxed{t_1}}, A_{21}^{\boxed{t_1}}] \\ &\vdots \\ s_m &: [A_1, A_2] \rightarrow [A_{1m}^{\boxed{t_m}}, A_{2m}^{\boxed{t_m}}] \end{aligned}$$

The values of $C^{\text{unit}}(\cdot, \cdot)$ are related by the following:

$$\begin{aligned} &C^{\text{unit}}([A_1, A_2], [B_1, B_2]) = \\ &\quad \delta([A_1, A_2] = [B_1, B_2]) + \\ &\quad \sum_i p_G(s_i) \cdot C^{\text{unit}}([A_{1i}, A_{2i}], [B_1, B_2]) \end{aligned}$$

where $\delta([A_1, A_2] = [B_1, B_2])$ is defined to be 1 if $[A_1, A_2] = [B_1, B_2]$ and 0 otherwise. This forms a system of linear equations in the unknown variables $C^{\text{unit}}(\cdot, \cdot)$. Such a system can be solved in polynomial time in the number of variables, for example using Gaussian elimination.

The elimination of unit rules starts with adding a rule $s' : [A_1 \rightarrow \alpha_1, A_2 \rightarrow \alpha_2]$ for each non-unit rule $s : [B_1 \rightarrow \alpha_1, B_2 \rightarrow \alpha_2]$ and pair $[A_1, A_2]$ such that $C^{\text{unit}}([A_1, A_2], [B_1, B_2]) > 0$. We assign to the new rule s' the probability $p_G(s) \cdot C^{\text{unit}}([A_1, A_2], [B_1, B_2])$. The unit rules can now be removed from the grammar. Again, in the resulting grammar the translation and the associated probability distribution will be the same as those in the source grammar. The new grammar has size $\mathcal{O}(|G|^2)$, where G is the input grammar. The time complexity is dominated by the computation of the solution of the linear system of equations. This computation takes cubic time in the number of variables. The number of variables in this case is $\mathcal{O}(|G|^2)$, which makes the running time $\mathcal{O}(|G|^6)$.

4 Prefix probabilities

The **joint prefix probability** $p_G^{\text{prefix}}([v_1, v_2])$ of a pair $[v_1, v_2]$ of terminal strings is the sum of the probabilities of all pairs of strings that have v_1 and v_2, respectively, as their prefixes. Formally:

$$p_G^{\text{prefix}}([v_1, v_2]) = \sum_{w_1, w_2 \in \Sigma^*} p_G([v_1 w_1, v_2 w_2])$$

At first sight, it is not clear this quantity can be effectively computed, as it involves a sum over infinitely many choices of w_1 and w_2. However, analogously to the case of context-free prefix probabilities (Jelinek and Lafferty, 1991), we can isolate two parts in the computation. One part involves infinite sums, which are independent of the input strings v_1 and v_2, and can be precomputed by solving a system of linear equations. The second part does rely on v_1 and v_2, and involves the actual evaluation of $p_G^{\text{prefix}}([v_1, v_2])$. This second part can be realized effectively, on the basis of the precomputed values from the first part.

In order to keep the presentation simple, and to allow for simple proofs of correctness, we solve the problem in a modular fashion. First, we present a transformation from a PSCFG $\mathcal{G} = (G, p_G)$, with $G = (N, \Sigma, P, S)$, to a PSCFG $\mathcal{G}_{\text{prefix}} = (G_{\text{prefix}}, p_{G_{\text{prefix}}})$, with $G_{\text{prefix}} = (N_{\text{prefix}}, \Sigma, P_{\text{prefix}}, S^{\downarrow})$. The latter grammar derives all possible pairs $[v_1, v_2]$ such that $[v_1 w_1, v_2 w_2]$ can be derived from G, for some w_1 and w_2. Moreover, $p_{G_{\text{prefix}}}([v_1, v_2]) = p_G^{\text{prefix}}([v_1, v_2])$, as will be verified later.

Computing $p_{G_{\text{prefix}}}([v_1, v_2])$ directly using a generic probabilistic parsing algorithm for PSCFGs is difficult, due to the presence of epsilon rules and unit rules. The next step will be to transform $\mathcal{G}_{\text{prefix}}$ into a third grammar $\mathcal{G}'_{\text{prefix}}$ by eliminating epsilon rules and unit rules from the underlying SCFG, and preserving the probability distribution over pairs of strings. Using $\mathcal{G}'_{\text{prefix}}$ one can then effectively

apply generic probabilistic parsing algorithms for PSCFGs, such as the inside algorithm discussed in section 3, in order to compute the desired prefix probabilities for the source PSCFG $\mathcal{G}$.

For each nonterminal A in the source SCFG G, the grammar G_{prefix} contains three nonterminals, namely A itself, $A^{\downarrow}$ and A^{ε}. The meaning of A remains unchanged, whereas $A^{\downarrow}$ is intended to generate a string that is a suffix of a known prefix v_1 or v_2. Nonterminals A^{ε} generate only the empty string, and are used to simulate the generation by G of infixes of the unknown suffix w_1 or w_2. The two left-hand sides of a synchronous rule in G_{prefix} can contain different combinations of nonterminals of the forms A, $A^{\downarrow}$, or A^{ε}. The start symbol of G_{prefix} is $S^{\downarrow}$. The structure of the rules from the source grammar is largely retained, except that some terminal symbols are omitted in order to obtain the intended interpretation of $A^{\downarrow}$ and A^{ε}.

In more detail, let us consider a synchronous rule $s : [A_1 \rightarrow \alpha_1, \quad A_2 \rightarrow \alpha_2]$ from the source grammar, where for $i \in \{1, 2\}$ we have:

$$\alpha_i = u_{i0} A_{i1}^{\boxed{t_{i1}}} u_{i1} \cdots u_{ir-1} A_{ir}^{\boxed{t_{ir}}} u_{ir}$$

The transformed grammar then contains a large number of rules, each of which is of the form $s' : [B_1 \rightarrow \beta_1, \quad B_2 \rightarrow \beta_2]$, where $B_i \rightarrow \beta_i$ is of one of three forms, namely $A_i \rightarrow \alpha_i$, $A_i^{\downarrow} \rightarrow \alpha_i^{\downarrow}$ or $A_i^{\varepsilon} \rightarrow \alpha_i^{\varepsilon}$, where $\alpha_i^{\downarrow}$ and α_i^{ε} are explained below. The choices for $i = 1$ and for $i = 2$ are independent, so that we can have $3 * 3 = 9$ kinds of synchronous rules, to be further subdivided in what follows. A unique label s' is produced for each new rule, and the probability of each new rule equals that of s.

The right-hand side α_i^{ε} is constructed by omitting all terminals and propagating downwards the ε superscript, resulting in:

$$\alpha_i^{\varepsilon} = A_{i1}^{\varepsilon\boxed{t_{i1}}} \cdots A_{ir}^{\varepsilon\boxed{t_{ir}}}$$

It is more difficult to define $\alpha_i^{\downarrow}$. In fact, there can be a number of choices for $\alpha_i^{\downarrow}$ and, for each choice, the transformed grammar contains an instance of the synchronous rule $s' : [B_1 \rightarrow \beta_1, \quad B_2 \rightarrow \beta_2]$ as defined above. The reason why different choices need to be considered is because the boundary between the known prefix v_i and the unknown suffix w_i can occur at different positions, either within a terminal string u_{ij} or else further down in a subderivation involving A_{ij}. In the first case, we have for some j $(0 \leq j \leq r)$:

$$\begin{aligned}\alpha_i^{\downarrow} = {} & u_{i0} A_{i1}^{\boxed{t_{i1}}} u_{i1} A_{i2}^{\boxed{t_{i2}}} \cdots \\ & u_{ij-1} A_{ij}^{\boxed{t_{ij}}} u'_{ij} A_{ij+1}^{\varepsilon\boxed{t_{ij+1}}} A_{ij+2}^{\varepsilon\boxed{t_{ij+2}}} \cdots A_{ir}^{\varepsilon\boxed{t_{ir}}}\end{aligned}$$

where u'_{ij} is a choice of a prefix of u_{ij}. In words, the known prefix ends after u'_{ij} and, thereafter, no more terminals are generated. We demand that u'_{ij} must not be the empty string, unless $A_i = S$ and $j = 0$. The reason for this restriction is that we want to avoid an overlap with the second case. In this second case, we have for some j $(1 \leq j \leq r)$:

$$\begin{aligned}\alpha_i^{\downarrow} = {} & u_{i0} A_{i1}^{\boxed{t_{i1}}} u_{i1} A_{i2}^{\boxed{t_{i2}}} \cdots \\ & u_{ij-1} A_{ij}^{\downarrow\boxed{t_{ij}}} A_{ij+1}^{\varepsilon\boxed{t_{ij+1}}} A_{ij+2}^{\varepsilon\boxed{t_{ij+2}}} \cdots A_{ir}^{\varepsilon\boxed{t_{ir}}}\end{aligned}$$

Here the known prefix of the input ends within a subderivation involving A_{ij}, and further to the right no more terminals are generated.

Example 3 Consider the synchronous rule $s : [A \rightarrow aB^{\boxed{1}}bc\,C^{\boxed{2}}d, D \rightarrow efE^{\boxed{2}}F^{\boxed{1}}]$. The first component of a synchronous rule derived from this can be one of the following eight:

$$\begin{aligned}A^{\varepsilon} &\rightarrow B^{\varepsilon\boxed{1}} C^{\varepsilon\boxed{2}} \\ A^{\downarrow} &\rightarrow aB^{\varepsilon\boxed{1}} C^{\varepsilon\boxed{2}} \\ A^{\downarrow} &\rightarrow aB^{\downarrow\boxed{1}} C^{\varepsilon\boxed{2}} \\ A^{\downarrow} &\rightarrow aB^{\boxed{1}}b\,C^{\varepsilon\boxed{2}} \\ A^{\downarrow} &\rightarrow aB^{\boxed{1}}bc\,C^{\varepsilon\boxed{2}} \\ A^{\downarrow} &\rightarrow aB^{\boxed{1}}bc\,C^{\downarrow\boxed{2}} \\ A^{\downarrow} &\rightarrow aB^{\boxed{1}}bc\,C^{\boxed{2}}d \\ A &\rightarrow aB^{\boxed{1}}bc\,C^{\boxed{2}}d\end{aligned}$$

The second component can be one of the following six:

$$\begin{aligned}D^{\varepsilon} &\rightarrow E^{\varepsilon\boxed{2}} F^{\varepsilon\boxed{1}} \\ D^{\downarrow} &\rightarrow eE^{\varepsilon\boxed{2}} F^{\varepsilon\boxed{1}} \\ D^{\downarrow} &\rightarrow efE^{\varepsilon\boxed{2}} F^{\varepsilon\boxed{1}} \\ D^{\downarrow} &\rightarrow efE^{\downarrow\boxed{2}} F^{\varepsilon\boxed{1}} \\ D^{\downarrow} &\rightarrow efE^{\boxed{2}} F^{\downarrow\boxed{1}} \\ D &\rightarrow efE^{\boxed{2}} F^{\boxed{1}}\end{aligned}$$

In total, the transformed grammar will contain $8 * 6 = 48$ synchronous rules derived from s. □

For each synchronous rule s, the above grammar transformation produces $\mathcal{O}(|s|)$ left rule components and as many right rule components. This means the number of new synchronous rules is $\mathcal{O}(|s|^2)$, and the size of each such rule is $\mathcal{O}(|s|)$. If we sum $\mathcal{O}(|s|^3)$ for every rule s we obtain a time and space complexity of $\mathcal{O}(|G|^3)$.

We now investigate formal properties of our grammar transformation, in order to relate it to prefix probabilities. We define the relation $\vdash$ between P and P_{prefix} such that $s \vdash s'$ if and only if s' was obtained from s by the transformation described above. This is extended in a natural way to derivations, such that $s_1 \cdots s_d \vdash s'_1 \cdots s'_{d'}$ if and only if $d = d'$ and $s_i \vdash s'_i$ for each i ($1 \leq i \leq d$).

The formal relation between G and G_{prefix} is revealed by the following two lemmas.

Lemma 1 *For each v_1, v_2, w_1, $w_2 \in \Sigma^*$ and $\sigma \in P^*$ such that $[S, S] \Rightarrow_G^{\sigma} [v_1 w_1, v_2 w_2]$, there is a unique $\sigma' \in P^*_{\text{prefix}}$ such that $[S^{\downarrow}, S^{\downarrow}] \Rightarrow_{G_{\text{prefix}}}^{\sigma'} [v_1, v_2]$ and $\sigma \vdash \sigma'$.* □

Lemma 2 *For each v_1, $v_2 \in \Sigma^*$ and derivation $\sigma' \in P^*_{\text{prefix}}$ such that $[S^{\downarrow}, S^{\downarrow}] \Rightarrow_{G_{\text{prefix}}}^{\sigma'} [v_1, v_2]$, there is a unique $\sigma \in P^*$ and unique $w_1, w_2 \in \Sigma^*$ such that $[S, S] \Rightarrow_G^{\sigma} [v_1 w_1, v_2 w_2]$ and $\sigma \vdash \sigma'$.* □

The only non-trivial issue in the proof of Lemma 1 is the uniqueness of σ'. This follows from the observation that the length of v_1 in $v_1 w_1$ uniquely determines how occurrences of left components of rules in P found in σ are mapped to occurrences of left components of rules in P_{prefix} found in σ'. The same applies to the length of v_2 in $v_2 w_2$ and the right components.

Lemma 2 is easy to prove as the structure of the transformation ensures that the terminals that are in rules from P but not in the corresponding rules from P_{prefix} occur at the end of a string v_1 (and v_2) to form the longer string $v_1 w_1$ (and $v_2 w_2$, respectively).

The transformation also ensures that $s \vdash s'$ implies $p_G(s) = p_{G_{\text{prefix}}}(s')$. Therefore $\sigma \vdash \sigma'$ implies $p_G(\sigma) = p_{G_{\text{prefix}}}(\sigma')$. By this and Lemmas 1 and 2 we may conclude:

Theorem 1 $p_{G_{\text{prefix}}}([v_1, v_2]) = p_G^{\text{prefix}}([v_1, v_2])$. □

Because of the introduction of rules with left-hand sides of the form A^{ε} in both the left and right components of synchronous rules, it is not straightforward to do effective probabilistic parsing with the grammar $\mathcal{G}_{\text{prefix}}$. We can however apply the transformations from section 3 to eliminate epsilon rules and thereafter eliminate unit rules, in a way that leaves the derived string pairs and their probabilities unchanged.

The simplest case is when the source grammar $\mathcal{G}$ is reduced, proper and consistent, and has no epsilon rules. The only nullable pairs of nonterminals in $\mathcal{G}_{\text{prefix}}$ will then be of the form $[A_1^{\varepsilon}, A_2^{\varepsilon}]$. Consider such a pair $[A_1^{\varepsilon}, A_2^{\varepsilon}]$. Because of reduction, properness and consistency of $\mathcal{G}$ we have:

$$\sum_{\substack{w_1, w_2 \in \Sigma^*,\ \sigma \in P^* \text{ s.t.} \\ [A_1^{\boxed{1}},\ A_2^{\boxed{1}}] \Rightarrow_G^{\sigma} [w_1,\ w_2]}} p_G(\sigma) \quad = \quad 1$$

Because of the structure of the grammar transformation by which $\mathcal{G}_{\text{prefix}}$ was obtained from $\mathcal{G}$, we also have:

$$\sum_{\substack{\sigma \in P^* \text{ s.t.} \\ [A_1^{\varepsilon \boxed{1}},\ A_2^{\varepsilon \boxed{1}}] \Rightarrow_{G_{\text{prefix}}}^{\sigma} [\varepsilon,\ \varepsilon]}} p_{G_{\text{prefix}}}(\sigma) = 1$$

Therefore pairs of occurrences of A_1^{ε} and A_2^{ε} with the same index in synchronous rules of G_{prefix} can be systematically removed without affecting the probability of the resulting rule, as outlined in section 3. Thereafter, unit rules can be removed to allow parsing by the inside algorithm for PSCFGs.

Following the computational analyses for all of the constructions presented in section 3, and for the grammar transformation discussed in this section, we can conclude that the running time of the proposed algorithm for the computation of prefix probabilities is dominated by the running time of the inside algorithm, which in the worst case is exponential in $|G|$. This result is not unexpected, as already pointed out in the introduction, since the recognition problem for PSCFGs is NP-complete, as established by Satta and Peserico (2005), and there is a straightforward reduction from the recognition problem for PSCFGs to the problem of computing the prefix probabilities for PSCFGs.

One should add that, in real world machine translation applications, it has been observed that recognition (and computation of inside probabilities) for SCFGs can typically be carried out in low-degree polynomial time, and the worst cases mentioned above are not observed with real data. Further discussion on this issue is due to Zhang et al. (2006).

5 Discussion

We have shown that the computation of joint prefix probabilities for PSCFGs can be reduced to the computation of inside probabilities for the same model. Our reduction relies on a novel grammar transformation, followed by elimination of epsilon rules and unit rules.

Next to the joint prefix probability, we can also consider the **right prefix probability**, which is defined by:

$$p_G^{\mathrm{r-prefix}}([v_1, v_2]) \quad = \quad \sum_w p_G([v_1, v_2 w])$$

In words, the entire left string is given, along with a prefix of the right string, and the task is to sum the probabilities of all string pairs for different suffixes following the given right prefix. This can be computed as a special case of the joint prefix probability. Concretely, one can extend the input and the grammar by introducing an end-of-sentence marker \$. Let G' be the underlying SCFG grammar after the extension. Then:

$$p_G^{\mathrm{r-prefix}}([v_1, v_2]) = p_{G'}^{\mathrm{prefix}}([v_1 \$, v_2])$$

Prefix probabilities and right prefix probabilities for PSCFGs can be exploited to compute probability distributions for the next word or part-of-speech in left-to-right incremental translation of speech, or alternatively as a predictive tool in applications of interactive machine translation, of the kind described by Foster et al. (2002). We provide some technical details here, generalizing to PSCFGs the approach by Jelinek and Lafferty (1991).

Let $\mathcal{G} = (G, p_G)$ be a PSCFG, with Σ the alphabet of terminal symbols. We are interested in the probability that the next terminal in the target translation is $a \in \Sigma$, after having processed a prefix v_1 of the source sentence and having produced a prefix v_2 of the target translation. This can be computed as:

$$p_G^{\mathrm{r-word}}(a \mid [v_1, v_2]) \quad = \quad \frac{p_G^{\mathrm{prefix}}([v_1, v_2 a])}{p_G^{\mathrm{prefix}}([v_1, v_2])}$$

Two considerations are relevant when applying the above formula in practice. First, the computation of $p_G^{\mathrm{prefix}}([v_1, v_2 a])$ need not be computed from scratch if $p_G^{\mathrm{prefix}}([v_1, v_2])$ has been computed already. Because of the tabular nature of the inside algorithm, one can extend the table for $p_G^{\mathrm{prefix}}([v_1, v_2])$ by adding new entries to obtain the table for $p_G^{\mathrm{prefix}}([v_1, v_2 a])$. The same holds for the computation of $p_G^{\mathrm{prefix}}([v_1 b, v_2])$.

Secondly, the computation of $p_G^{\mathrm{prefix}}([v_1, v_2 a])$ for all possible $a \in \Sigma$ may be impractical. However, one may also compute the probability that the next part-of-speech in the target translation is A. This can be realised by adding a rule $s' : [B \to b, A \to c_A]$ for each rule $s : [B \to b, A \to a]$ from the source grammar, where A is a nonterminal representing a part-of-speech and c_A is a (pre-)terminal specific to A. The probability of s' is the same as that of s. If G' is the underlying SCFG after adding such rules, then the required value is $p_{G'}^{\mathrm{prefix}}([v_1, v_2\, c_A])$.

One variant of the definitions presented in this paper is the notion of *infix* probability, which is useful in island-driven speech translation. Here we are interested in the probability that any string in the source language with infix v_1 is translated into any string in the target language with infix v_2. However, just as infix probabilities are difficult to compute for probabilistic context-free grammars (Corazza et al., 1991; Nederhof and Satta, 2008) so (joint) infix probabilities are difficult to compute for PSCFGs. The problem lies in the possibility that a given infix may occur more than once in a string in the language. The computation of infix probabilities can be reduced to that of solving non-linear systems of equations, which can be approximated using for instance Newton's algorithm. However, such a system of equations is built from the input strings, which entails that the computational effort of solving the system primarily affects parse time rather than parser-generation time.

References

S. Abney, D. McAllester, and F. Pereira. 1999. Relating probabilistic grammars and automata. In *37th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 542–549, Maryland, USA, June.

A.V. Aho and J.D. Ullman. 1969. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3:37–56.

Z. Chi. 1999. Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, 25(1):131–160.

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

A. Corazza, R. De Mori, R. Gretter, and G. Satta. 1991. Computation of probabilities for an island-driven parser. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):936–950.

G. Foster, P. Langlais, and G. Lapalme. 2002. User-friendly text prediction for translators. In *Conference on Empirical Methods in Natural Language Processing*, pages 148–155, University of Pennsylvania, Philadelphia, PA, USA, July.

M. Galley, M. Hopkins, K. Knight, and D. Marcu. 2004. What's in a translation rule? In *HLT-NAACL 2004, Proceedings of the Main Conference*, Boston, Massachusetts, USA, May.

D. Gildea and D. Stefankovic. 2007. Worst-case synchronous grammar rules. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics, Proceedings of the Main Conference*, pages 147–154, Rochester, New York, USA, April.

M. Hopkins and G. Langmead. 2010. SCFG decoding without binarization. In *Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 646–655, October.

F. Jelinek and J.D. Lafferty. 1991. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17(3):315–323.

S. Kiefer, M. Luttenberger, and J. Esparza. 2007. On the convergence of Newton's method for monotone systems of polynomial equations. In *Proceedings of the 39th ACM Symposium on Theory of Computing*, pages 217–266.

C.D. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

D. McAllester. 2002. On the complexity analysis of static analyses. *Journal of the ACM*, 49(4):512–537.

M.-J. Nederhof and G. Satta. 2008. Computing partition functions of PCFGs. *Research on Language and Computation*, 6(2):139–162.

G. Satta and E. Peserico. 2005. Some computational complexity results for synchronous context-free grammars. In *Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 803–810.

S.M. Shieber, Y. Schabes, and F.C.N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–36.

S. Sippu and E. Soisalon-Soininen. 1988. *Parsing Theory, Vol. I: Languages and Parsing*, volume 15 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag.

A. Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):167–201.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 256–263, New York, USA, June.

A Comparison of Loopy Belief Propagation and Dual Decomposition for Integrated CCG Supertagging and Parsing

Michael Auli
School of Informatics
University of Edinburgh
m.auli@sms.ed.ac.uk

Adam Lopez
HLTCOE
Johns Hopkins University
alopez@cs.jhu.edu

Abstract

Via an oracle experiment, we show that the upper bound on accuracy of a CCG parser is significantly lowered when its search space is pruned using a supertagger, though the supertagger also prunes many bad parses. Inspired by this analysis, we design a single model with both supertagging and parsing features, rather than separating them into distinct models chained together in a pipeline. To overcome the resulting increase in complexity, we experiment with both belief propagation and dual decomposition approaches to inference, the first empirical comparison of these algorithms that we are aware of on a structured natural language processing problem. On CCGbank we achieve a labelled dependency F-measure of 88.8% on gold POS tags, and 86.7% on automatic part-of-speeoch tags, the best reported results for this task.

1 Introduction

Accurate and efficient parsing of Combinatorial Categorial Grammar (CCG; Steedman, 2000) is a long-standing problem in computational linguistics, due to the complexities associated its mild context sensitivity. Even for practical CCG that are strongly context-free (Fowler and Penn, 2010), parsing is much harder than with Penn Treebank-style context-free grammars, with vast numbers of nonterminal categories leading to increased grammar constants. Where a typical Penn Treebank grammar may have fewer than 100 nonterminals (Hockenmaier and Steedman, 2002), we found that a CCG grammar derived from CCGbank contained over 1500. The same grammar assigns an average of 22 lexical categories per word (Clark and Curran, 2004a), resulting in an enormous space of possible derivations.

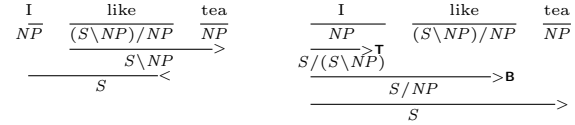
The most successful approach to CCG parsing is based on a pipeline strategy (§2). First, we tag (or multitag) each word of the sentence with a lexical category using a *supertagger*, a sequence model over these categories (Bangalore and Joshi, 1999; Clark, 2002). Second, we parse the sentence under the requirement that the lexical categories are fixed to those preferred by the supertagger. Variations on this approach drive the widely-used, broad coverage C&C parser (Clark and Curran, 2004a; Clark and Curran, 2007; Kummerfeld et al., 2010). However, it fails when the supertagger makes errors. We show experimentally that this pipeline significantly lowers the upper bound on parsing accuracy (§3).

The same experiment shows that the supertagger prunes many bad parses. So, while we want to avoid the error propagation inherent to a pipeline, ideally we still want to benefit from the key insight of supertagging: that a sequence model over lexical categories can be quite accurate. Our solution is to combine the features of both the supertagger and the parser into a single, less aggressively pruned model. The challenge with this model is its prohibitive complexity, which we address with *approximate* methods: dual decomposition and belief propagation (§4). We present the first side-by-side comparison of these algorithms on an NLP task of this complexity, measuring accuracy, convergence behavior, and runtime. In both cases our model significantly outperforms the pipeline approach, leading to the best published results in CCG parsing (§5).

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 470–480,
Portland, Oregon, June 19-24, 2011.

2 CCG and Supertagging

CCG is a lexicalized grammar formalism encoding for each word lexical categories that are either basic (eg. NN, JJ) or complex. Complex lexical categories specify the number and directionality of arguments. For example, one lexical category for the verb *like* is $(S\backslash NP)/NP$, specifying the first argument as an NP to the right and the second as an NP to the left; there are over 100 lexical categories for *like* in our lexicon. In parsing, adjacent spans are combined using a small number of binary combinatory rules like forward application or composition (Steedman, 2000; Fowler and Penn, 2010). In the first derivation below, $(S\backslash NP)/NP$ and NP combine to form the spanning category $S\backslash NP$, which only requires an NP to its left to form a complete sentence-spanning S. The second derivation uses type-raising to change the category type of *I*.



As can be inferred from even this small example, a key difficulty in parsing CCG is that the number of categories quickly becomes extremely large, and there are typically many ways to analyze every span of a sentence.

Supertagging (Bangalore and Joshi, 1999; Clark, 2002) treats the assignment of lexical categories (or *supertags*) as a sequence tagging problem. Because they do this with high accuracy, they are often exploited to prune the parser's search space: the parser only considers lexical categories with high posterior probability (or other figure of merit) under the supertagging model (Clark and Curran, 2004a). The posterior probabilities are then discarded; it is the extensive pruning of lexical categories that leads to substantially faster parsing times.

Pruning the categories in advance this way has a specific failure mode: sometimes it is not possible to produce a sentence-spanning derivation from the tag sequences preferred by the supertagger, since it does not enforce grammaticality. A workaround for this problem is the *adaptive supertagging* (AST) approach of Clark and Curran (2004a). It is based on a step function over supertagger beam widths, relaxing the pruning threshold for lexical categories only if the parser fails to find an analysis. The process either succeeds and returns a parse after some iteration or gives up after a predefined number of iterations. As Clark and Curran (2004a) show, most sentences can be parsed with a very small number of supertags per word. However, the technique is inherently approximate: it will return a lower probability parse under the parsing model if a higher probability parse can only be constructed from a supertag sequence returned by a subsequent iteration. In this way it prioritizes speed over exactness, although the tradeoff can be modified by adjusting the beam step function. Regardless, the effect of the approximation is unbounded.

We will also explore *reverse adaptive supertagging*, a much less aggressive pruning method that seeks only to make sentences parseable when they otherwise would not be due to an impractically large search space. Reverse AST starts with a wide beam, narrowing it at each iteration only if a maximum chart size is exceeded. In this way it prioritizes exactness over speed.

3 Oracle Parsing

What is the effect of these approximations? To answer this question we computed oracle best and worst values for labelled dependency F-score using the algorithm of Huang (2008) on the hybrid model of Clark and Curran (2007), the best model of their C&C parser. We computed the oracle on our development data, Section 00 of CCGbank (Hockenmaier and Steedman, 2007), using both AST and Reverse AST beams settings shown in Table 1.

The results (Table 2) show that the oracle best accuracy for reverse AST is more than 3% higher than the aggressive AST pruning.[1] In fact, it is almost as high as the upper bound oracle accuracy of 97.73% obtained using *perfect* supertags—in other words, the search space for reverse AST is theoretically near-optimal.[2] We also observe that the oracle

[1]The numbers reported here and in later sections differ slightly from those in a previously circulated draft of this paper, for two reasons: we evaluate only on sentences for which a parse was returned instead of all parses, to enable direct comparison with Clark and Curran (2007); and we use their hybrid model instead of their normal-form model, except where noted. Despite these changes our main findings remained unchanged.

[2]This idealized oracle reproduces a result from Clark and Cur-

Condition	Parameter	Iteration 1	2	3	4	5
AST	β (beam width)	0.075	0.03	0.01	0.005	0.001
	k (dictionary cutoff)	20	20	20	20	150
Reverse	β	0.001	0.005	0.01	0.03	0.075
	k	150	20	20	20	20

Table 1: Beam step function used for standard (AST) and less aggressive (Reverse) AST throughout our experiments. Parameter β is a beam threshold while k bounds the use of a part-of-speech tag dictionary, which is used for words seen less than k times.

	Viterbi F-score			Oracle Max F-score			Oracle Min F-score			
	LF	LP	LR	LF	LP	LR	LF	LP	LR	cat/word
AST	**87.38**	**87.83**	86.93	94.35	95.24	93.49	54.31	54.81	53.83	1.3-3.6
Reverse	87.36	87.55	**87.17**	**97.65**	**98.21**	**97.09**	18.09	17.75	18.43	3.6-1.3

Table 2: Comparison of adaptive supertagging (AST) and a less restrictive setting (Reverse) with Viterbi and oracle F-scores on CCGbank Section 00. The table shows the labelled F-score (LF), precision (LP) and recall (LR) and the the number of lexical categories per word used (from first to last parsing attempt).

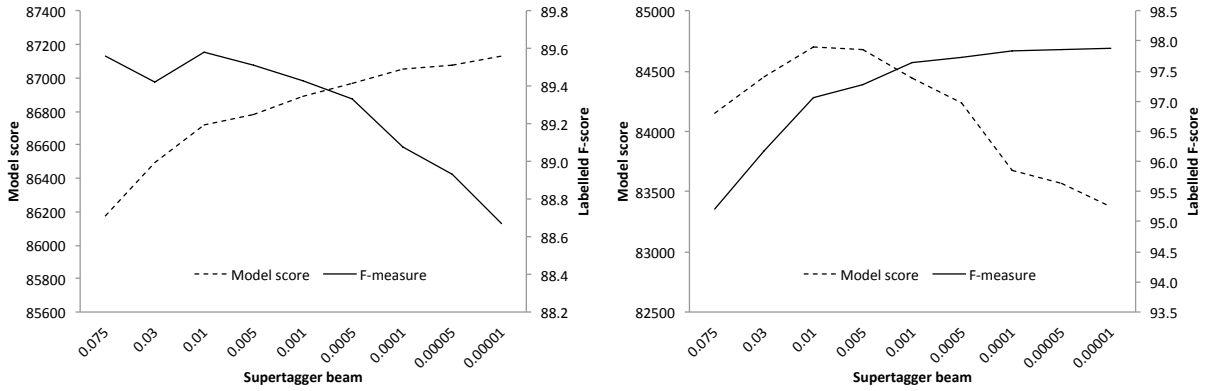


Figure 1: Comparison between model score and Viterbi F-score (left); and between model score and oracle F-score (right) for different supertagger beams on a subset of CCGbank Section 00.

worst accuracy is much lower in the reverse setting. It is clear that the supertagger pipeline has two effects: while it beneficially prunes many bad parses, it harmfully prunes some very good parses. We can also see from the scores of the Viterbi parses that while the reverse condition has access to much better parses, the model doesn't actually find them. This mirrors the result of Clark and Curran (2007) that they use to justify AST.

Digging deeper, we compared parser model score against Viterbi F-score and oracle F-score at a variety of fixed beam settings (Figure 1), considering only the subset of our development set which could be parsed with *all* beam settings. The inverse relationship between model score and F-score shows that the supertagger restricts the parser to mostly good parses (under F-measure) that the model would otherwise disprefer. Exactly this effect is exploited in the pipeline model. However, when the supertagger makes a mistake, the parser cannot recover.

4 Integrated Supertagging and Parsing

The supertagger obviously has good but not perfect predictive features. An obvious way to exploit this without being bound by its decisions is to incorporate these features directly into the parsing model.

ran (2004b). The reason that using the gold-standard supertags doesn't result in 100% oracle parsing accuracy is that some of the development set parses cannot be constructed by the learned grammar.

In our case both the parser and the supertagger are feature-based models, so from the perspective of a single parse tree, the change is simple: the tree is simply scored by the weights corresponding to all of its active features. However, since the features of the supertagger are all Markov features on adjacent supertags, the change has serious implications for search. If we think of the supertagger as defining a weighted regular language consisting of all supertag sequences, and the parser as defining a weighted mildly context-sensitive language consisting of only a *subset* of these sequences, then the search problem is equivalent to finding the optimal derivation in the weighted intersection of a regular and mildly context-sensitive language. Even allowing for the observation of Fowler and Penn (2010) that our practical CCG is context-free, this problem still reduces to the construction of Bar-Hillel et al. (1964), making search very expensive. Therefore we need approximations.

Fortunately, recent literature has introduced two relevant approximations to the NLP community: loopy belief propagation (Pearl, 1988), applied to dependency parsing by Smith and Eisner (2008); and dual decomposition (Dantzig and Wolfe, 1960; Komodakis et al., 2007; Sontag et al., 2010, *inter alia*), applied to dependency parsing by Koo et al. (2010) and lexicalized CFG parsing by Rush et al. (2010). We apply both techniques to our integrated supertagging and parsing model.

4.1 Loopy Belief Propagation

Belief propagation (BP) is an algorithm for computing marginals (i.e. expectations) on structured models. These marginals can be used for decoding (parsing) in a minimum-risk framework (Smith and Eisner, 2008); or for training using a variety of algorithms (Sutton and McCallum, 2010). We experiment with both uses in §5. Many researchers in NLP are familiar with two special cases of belief propagation: the forward-backward and inside-outside algorithms, used for computing expectations in sequence models and context-free grammars, respectively.[3] Our use of belief propagation builds directly on these two familiar algorithms.

[3]Forward-backward and inside-outside are formally shown to be special cases of belief propagation by Smyth et al. (1997) and Sato (2007), respectively.

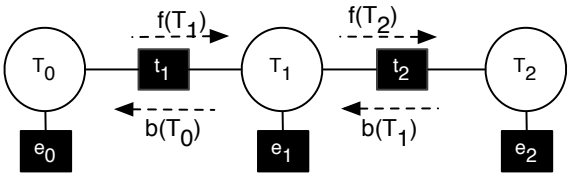


Figure 2: Supertagging factor graph with messages. Circles are variables and filled squares are factors.

BP is usually understood as an algorithm on bipartite *factor graphs*, which structure a global function into local functions over subsets of variables (Kschischang et al., 1998). Variables maintain a *belief* (expectation) over a distribution of values and BP passes *messages* about these beliefs between variables and factors. The idea is to iteratively update each variable's beliefs based on the beliefs of neighboring variables (through a shared factor), using the sum-product rule.

This results in the following equation for a message $m_{x\to f}(x)$ from a variable x to a factor f

$$m_{x\to f}(x) = \prod_{h\in n(x)\setminus f} m_{h\to x}(x) \qquad (1)$$

where $n(x)$ is the set of all neighbours of x. The message $m_{f\to x}$ from a factor to a variable is

$$m_{f\to x}(x) = \sum_{\sim\{x\}} f(X) \prod_{y\in n(f)\setminus x} m_{y\to f}(y) \qquad (2)$$

where $\sim\{x\}$ represents all variables other than x, $X = n(f)$ and $f(X)$ is the set of arguments of the factor function f.

Making this concrete, our supertagger defines a distribution over tags $T_0...T_I$, based on emission factors $e_0...e_I$ and transition factors $t_1...t_I$ (Figure 2). The message f_i a variable T_i receives from its neighbor to the left corresponds to the forward probability, while messages from the right correspond to backward probability b_i.

$$f_i(T_i) = \sum_{T_{i-1}} f_{i-1}(T_{i-1}) e_{i-1}(T_{i-1}) t_i(T_{i-1}, T_i) \quad (3)$$

$$b_i(T_i) = \sum_{T_{i+1}} b_{i+1}(T_{i+1}) e_{i+1}(T_{i+1}) t_{i+1}(T_i, T_{i+1}) \quad (4)$$

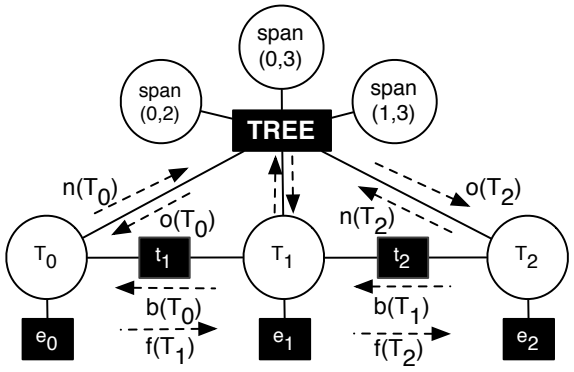


Figure 3: Factor graph for the combined parsing and supertagging model.

The current belief $B_x(x)$ for variable x can be computed by taking the normalized product of all its incoming messages.

$$B_x(x) = \frac{1}{Z} \prod_{h \in n(x)} m_{h \to x}(x) \tag{5}$$

In the supertagger model, this is just:

$$p(T_i) = \frac{1}{Z} f_i(T_i) b_i(T_i) e_i(T_i) \tag{6}$$

Our parsing model is *also* a distribution over variables T_i, along with an additional quadratic number of $span(i, j)$ variables. Though difficult to represent pictorially, a distribution over parses is captured by an extension to graphical models called case-factor diagrams (McAllester et al., 2008). We add this complex distribution to our model as a single factor (Figure 3). This is a natural extension to the use of complex factors described by Smith and Eisner (2008) and Dreyer and Eisner (2009).

When a factor graph is a tree as in Figure 2, BP converges in a single iteration to the exact marginals. However, when the model contains cycles, as in Figure 3, we can iterate message passing. Under certain assumptions this *loopy* BP it will converge to approximate marginals that are bounded under an interpretation from statistical physics (Yedidia et al., 2001; Sutton and McCallum, 2010).

The TREE factor exchanges *inside* n_i and *outside* o_i messages with the tag and span variables, taking into account beliefs from the sequence model. We will omit the unchanged outside recursion for brevity, but inside messages $n(C_{i,j})$ for category $C_{i,j}$ in $span(i, j)$ are computed using rule probabilities r as follows:

$$n(C_{i,j}) = \begin{cases} f_i(C_{i,j}) b_i(C_{i,j}) e_i(C_{i,j}) & \text{if j=i+1} \\ \sum_{k,X,Y} n(X_{i,k}) n(Y_{k,j}) r(C_{i,j}, X_{i,k}, Y_{k,j}) \end{cases} \tag{7}$$

Note that the only difference from the classic inside algorithm is that the recursive base case of a category spanning a single word has been replaced by a message from the supertag that contains both forward and backward factors, along with a unary emission factor, which doubles as a unary rule factor and thus contains the only shared features of the original models. This difference is also mirrored in the forward and backward messages, which are identical to Equations 3 and 4, except that they also incorporate outside messages from the tree factor.

Once all forward-backward and inside-outside probabilities have been calculated the belief of supertag T_i can be computed as the product of all incoming messages. The only difference from Equation 6 is the addition of the outside message.

$$p(T_i) = \frac{1}{Z} f_i(T_i) b_i(T_i) e_i(T_i) o_i(T_i) \tag{8}$$

The algorithm repeatedly runs forward-backward and inside-outside, passing their messages back and forth, until these quantities converge.

4.2 Dual Decomposition

Dual decomposition (Rush et al., 2010; Koo et al., 2010) is a decoding (i.e. search) algorithm for problems that can be decomposed into exactly solvable subproblems: in our case, supertagging and parsing. Formally, given Y as the set of valid parses, Z as the set of valid supertag sequences, and T as the set of supertags, we want to solve the following optimization for parser $f(y)$ and supertagger $g(z)$.

$$\arg\max_{y \in Y, z \in Z} f(y) + g(z) \tag{9}$$

$$\text{such that } y(i,t) = z(i,t) \text{ for all } (i,t) \in I \tag{10}$$

Here $y(i, t)$ is a binary function indicating whether word i is assigned supertag t by the parser, for the

set $I = \{(i,t) : i \in 1 \ldots n, t \in T\}$ denoting the set of permitted supertags for each word; similarly $z(i,t)$ for the supertagger. To enforce the constraint that the parser and supertagger agree on a tag sequence we introduce Lagrangian multipliers $u = \{u(i,t) : (i,t) \in I\}$ and construct a dual objective over variables $u(i,t)$.

$$L(u) = \max_{y \in Y}(f(y) - \sum_{i,t} u(i,t)y(i,t)) \quad (11)$$
$$+ \max_{z \in Z}(f(z) + \sum_{i,t} u(i,t)z(i,t))$$

This objective is an upper bound that we want to make as tight as possible by solving for $\min_u L(u)$. We optimize the values of the $u(i,t)$ variables using the same algorithm as Rush et al. (2010) for their tagging and parsing problem (essentially a perceptron update).[4] An advantages of DD is that, on convergence, it recovers *exact* solutions to the combined problem. However, if it does not converge or we stop early, an approximation must be returned: following Rush et al. (2010) we used the highest scoring output of the parsing submodel over all iterations.

5 Experiments

Parser. We use the C&C parser (Clark and Curran, 2007) and its supertagger (Clark, 2002). Our baseline is the hybrid model of Clark and Curran (2007); our integrated model simply adds the supertagger features to this model. The parser relies solely on the supertagger for pruning, using CKY for search over the pruned space. Training requires repeated calculation of feature expectations over packed charts of derivations. For training, we limited the number of items in this chart to 0.3 million, and for testing, 1 million. We also used a more permissive training supertagger beam (Table 3) than in previous work (Clark and Curran, 2007). Models were trained with the parser's L-BFGS trainer.

Evaluation. We evaluated on CCGbank (Hockenmaier and Steedman, 2007), a right-most normal-form CCG version of the Penn Treebank. We use sections 02-21 (39603 sentences) for training, section 00 (1913 sentences) for development and section 23 (2407 sentences) for testing. We supply gold-standard part-of-speech tags to the parsers. Evaluation is based on labelled and unlabelled predicate argument structure recovery and supertag accuracy. We only evaluate on sentences for which an analysis was returned; the coverage for *all* parsers is 99.22% on section 00, and 99.63% on section 23.

Model combination. We combine the parser and the supertagger over the search space defined by the set of supertags within the supertagger beam (see Table 1); this avoids having to perform inference over the prohibitively large set of parses spanned by all supertags. Hence at each beam setting, the model operates over the same search space as the baseline; the difference is that we search with our integrated model.

5.1 Parsing Accuracy

We first experiment with the separately trained supertagger and parser, which are then combined using belief propagation (BP) and dual decomposition (DD). We run the algorithms for many iterations, and irrespective of convergence, for BP we compute the minimum risk parse from the current marginals, and for DD we choose the highest-scoring parse seen over all iterations. We measured the evolving accuracy of the models on the development set (Figure 4). In line with our oracle experiment, these results demonstrate that we can coax more accurate parses from the larger search space provided by the reverse setting; the influence of the supertagger features allow us to exploit this advantage.

One behavior we observe in the graph is that the DD results tend to incrementally improve in accuracy while the BP results quickly stabilize, mirroring the result of Smith and Eisner (2008). This occurs because DD continues to find higher scoring parses at each iteration, and hence the results change. However for BP, even if the marginals have not converged, the minimum risk solution turns out to be fairly stable across successive iterations.

We next compare the algorithms against the baseline on our test set (Table 4). We find that the early stability of BP's performance generalises to the test set as does DD's improvement over several iterations. More importantly, we find that the applying

[4] The u terms can be interpreted as the *messages* from factors to variables (Sontag et al., 2010) and the resulting message passing algorithms are similar to the *max-product algorithm*, a sister algorithm to BP.

	Parameter	Iteration 1	2	3	4	5	6	7
`Training`	β	0.001	0.001	0.0045	0.0055	0.01	0.05	0.1
	k	150	20	20	20	20	20	20

Table 3: Beam step function used for training (cf. Table 1).

	section 00 (dev)						section 23 (test)					
	AST			Reverse			AST			Reverse		
	LF	UF	ST	LF	UF	ST	LF	UF	ST	LF	UF	ST
Baseline	87.38	93.08	94.21	87.36	93.13	93.99	87.73	93.09	94.33	87.65	93.06	94.01
C&C '07	87.24	93.00	94.16	-	-	-	87.64	93.00	94.32	-	-	-
$BP_{k=1}$	87.70	93.28	94.44	**88.35**	93.69	**94.73**	**88.20**	**93.28**	**94.60**	88.78	93.66	94.81
$BP_{k=25}$	87.70	93.31	94.44	88.33	**93.72**	94.71	88.19	93.27	94.59	88.80	93.68	94.81
$DD_{k=1}$	87.40	93.09	94.23	87.38	93.15	94.03	87.74	93.10	94.33	87.67	93.07	94.02
$DD_{k=25}$	**87.71**	**93.32**	**94.44**	88.29	93.71	94.67	88.14	93.24	94.59	**88.80**	**93.68**	**94.82**

Table 4: Results for individually-trained submodels combined using dual decomposition (DD) or belief propagation (BP) for k iterations, evaluated by labelled and unlabelled F-score (LF/UF) and supertag accuracy (ST). We compare against the previous best result of Clark and Curran (2007); our baseline is their model with wider training beams (cf. Table 3).

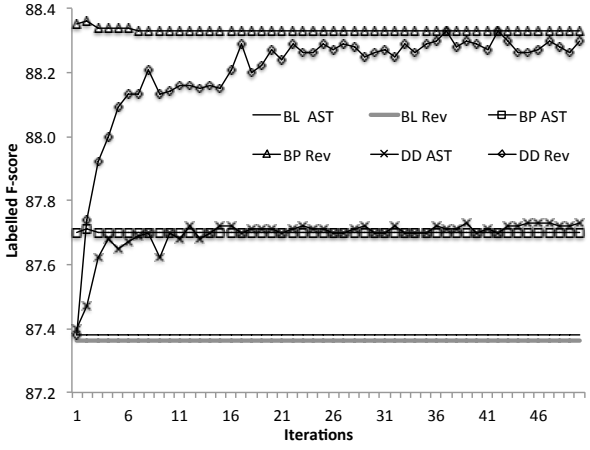


Figure 4: Labelled F-score of baseline (BL), belief propagation (BP), and dual decomposition (DD) on section 00.

our combined model using either algorithm *consistently* outperforms the baseline after only a few iterations. Overall, we improve the labelled F-measure by almost 1.1% and unlabelled F-measure by 0.6% over the baseline. To the best of our knowledge, the results obtained with BP and DD are the best reported results on this task using gold POS tags.

Next, we evaluate performance when using *automatic* part-of-speech tags as input to our parser and supertagger (Table 5). This enables us to compare against the results of Fowler and Penn (2010), who trained the Petrov parser (Petrov et al., 2006) on CCGbank. We outperform them on all criteria. Hence our combined model represents the best CCG parsing results under any setting.

Finally, we revisit the oracle experiment of §3 using our combined models (Figure 5). Both show an improved relationship between model score and F-measure.

5.2 Algorithmic Convergence

Figure 4 shows that parse accuracy converges after a few iterations. Do the *algorithms* converge? BP converges when the marginals do not change between iterations, and DD converges when both submodels agree on all supertags. We measured the convergence of each algorithm under these criteria over 1000 iterations (Figure 6). DD converges much faster, while BP in the reverse condition converges quite slowly. This is interesting when contrasted with its behavior on parse accuracy—its rate of convergence after one iteration is 1.5%, but its accuracy is already the highest at this point. Over the entire 1000 iterations, most sentences converge: all but 3 for BP (both in AST and reverse) and all but

	section 00 (dev)						section 23 (test)					
	LF	LP	LR	UF	UP	UR	LF	LP	LR	UF	UP	UR
Baseline	85.53	85.73	85.33	91.99	92.20	91.77	85.74	85.90	85.58	91.92	92.09	91.75
Petrov I-5	85.79	86.09	85.50	92.44	92.76	92.13	86.01	86.29	85.74	92.34	92.64	92.04
$BP_{k=1}$	**86.44**	**86.74**	**86.14**	**92.54**	**92.86**	**92.23**	**86.73**	**86.95**	**86.50**	**92.45**	**92.69**	**92.21**
$DD_{k=25}$	86.35	86.65	86.05	92.52	92.85	92.20	86.68	86.90	86.46	92.44	92.67	92.21

Table 5: Results on automatically assigned POS tags. *Petrov I-5* is based on the parser output of Fowler and Penn (2010); we evaluate on sentences for which *all* parsers returned an analysis (2323 sentences for section 23 and 1834 sentences for section 00).

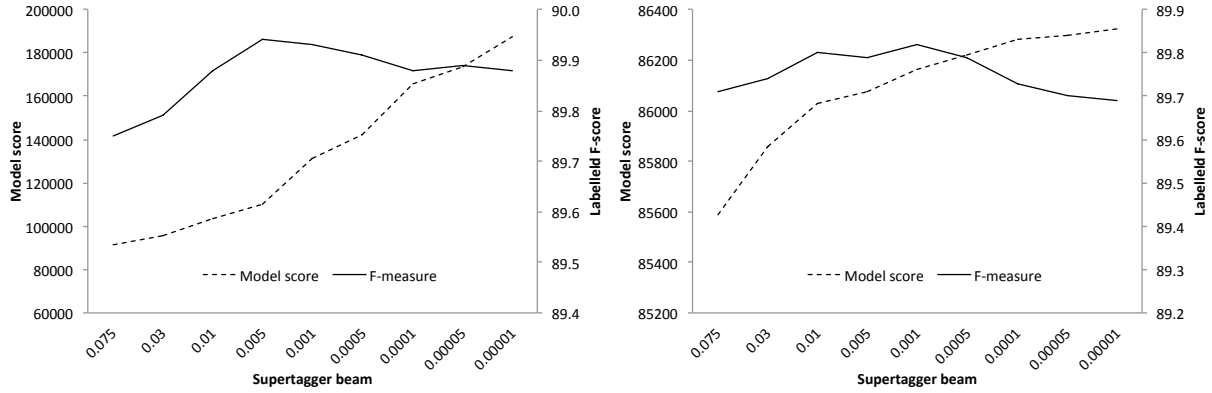


Figure 5: Comparison between model score and Viterbi F-score for the integrated model using belief propagation (left) and dual decomposition (right); the results are based on the same data as Figure 1.

.

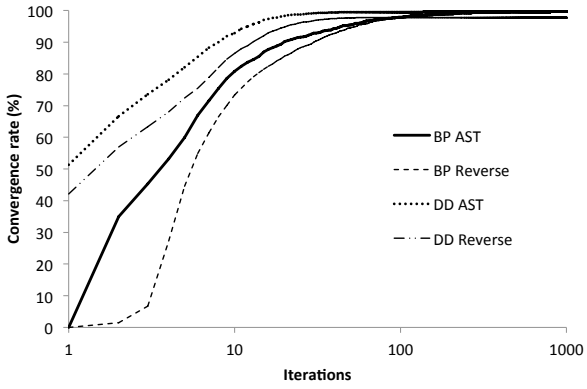


Figure 6: Rate of convergence for belief propagation (BP) and dual decomposition (DD) with maximum $k = 1000$.

41 (2.6%) for DD in reverse (6 in AST).

5.3 Parsing Speed

Because the C&C parser with AST is very fast, we wondered about the effect on speed for our model. We measured the runtime of the algorithms under the condition that we stopped at a particular iteration (Table 6). Although our models improve substantially over C&C, there is a significant cost in speed for the best result.

5.4 Training the Integrated Model

In the experiments reported so far, the parsing and supertagging models were trained separately, and only combined at test time. Although the outcome of these experiments was successful, we wondered if we could obtain further improvements by training the model parameters together.

Since the gradients produced by (loopy) BP are approximate, for these experiments we used a stochastic gradient descent (SGD) trainer (Bottou, 2003). We found that the SGD parameters described by Finkel et al. (2008) worked equally well for our models, and, on the baseline, produced similar results to L-BFGS. Curiously, however, we found that the combined model does not perform as well when

	AST		Reverse	
	sent/sec	LF	sent/sec	LF
Baseline	65.8	87.38	5.9	87.36
$BP_{k=1}$	60.8	87.70	5.8	88.35
$BP_{k=5}$	46.7	87.70	4.7	88.34
$BP_{k=25}$	35.3	87.70	3.5	88.33
$DD_{k=1}$	64.6	87.40	5.9	87.38
$DD_{k=5}$	41.9	87.65	3.1	88.09
$DD_{k=25}$	32.5	87.71	1.9	88.29

Table 6: Parsing time in seconds per sentence (vs. F-measure) on section 00.

	AST			Reverse		
	LF	UF	ST	LF	UF	ST
Baseline	86.7	92.7	94.0	86.7	92.7	93.9
BP inf	86.8	92.8	94.1	87.2	93.1	94.2
BP train	86.3	92.5	93.8	85.6	92.1	93.2

Table 7: Results of training with SGD on approximate gradients from LPB on section 00. We test LBP in both inference and training (train) as well as in inference only (inf); a maximum number of 10 iterations is used.

the parameters are trained together (Table 7). A possible reason for this is that we used a stricter supertagger beam setting during training (Clark and Curran, 2007) to make training on a single machine practical. This leads to lower performance, particularly in the Reverse condition. Training a model using DD would require a different optimization algorithm based on Viterbi results (e.g. the perceptron) which we will pursue in future work.

6 Conclusion and Future Work

Our approach of combining models to avoid the pipeline problem (Felzenszwalb and McAllester, 2007) is very much in line with much recent work in NLP. Such diverse topics as machine translation (Dyer et al., 2008; Dyer and Resnik, 2010; Mi et al., 2008), part-of-speech tagging (Jiang et al., 2008), named entity recognition (Finkel and Manning, 2009) semantic role labelling (Sutton and McCallum, 2005; Finkel et al., 2006), and others have also been improved by combined models. Our empirical comparison of BP and DD also complements the theoretically-oriented comparison of marginal- and margin-based variational approximations for parsing described by Martins et al. (2010).

We have shown that the aggressive pruning used in adaptive supertagging significantly harms the oracle performance of the parser, though it mostly prunes bad parses. Based on these findings, we combined parser and supertagger features into a single model. Using belief propagation and dual decomposition, we obtained more principled—and more accurate—approximations than a pipeline. Models combined using belief propagation achieve very good performance immediately, despite an initial convergence rate just over 1%, while dual decomposition produces comparable results after several iterations, and algorithmically converges more quickly. Our best result of 88.8% represents the state-of-the art in CCG parsing accuracy.

In future work we plan to integrate the POS tagger, which is crucial to parsing accuracy (Clark and Curran, 2004b). We also plan to revisit the idea of combined training. Though we have focused on CCG in this work we expect these methods to be equally useful for other linguistically motivated but computationally complex formalisms such as lexicalized tree adjoining grammar.

Acknowledgements

We would like to thank Phil Blunsom, Prachya Boonkwan, Christos Christodoulopoulos, Stephen Clark, Michael Collins, Chris Dyer, Timothy Fowler, Mark Granroth-Wilding, Philipp Koehn, Terry Koo, Tom Kwiatkowski, André Martins, Matt Post, David Smith, David Sontag, Mark Steedman, and Charles Sutton for helpful discussion related to this work and comments on previous drafts, and the anonymous reviewers for helpful comments. We also acknowledge funding from EPSRC grant EP/P504171/1 (Auli); the EuroMatrixPlus project funded by the European Commission, 7th Framework Programme (Lopez); and the resources provided by the Edinburgh Compute and Data Facility (`http://www.ecdf.ed.ac.uk`). The ECDF is partially supported by the eDIKT initiative (`http://www.edikt.org.uk`).

References

S. Bangalore and A. K. Joshi. 1999. Supertagging: An Approach to Almost Parsing. *Computational Linguis-*

tics, 25(2):238–265, June.

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

L. Bottou. 2003. Stochastic learning. In *Advanced Lectures in Machine Learning*, pages 146–168.

S. Clark and J. R. Curran. 2004a. The importance of supertagging for wide-coverage CCG parsing. In *COLING*, Morristown, NJ, USA.

S. Clark and J. R. Curran. 2004b. Parsing the WSJ using CCG and log-linear models. In *Proc. of ACL*, pages 104–111, Barcelona, Spain.

S. Clark and J. R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4):493–552.

S. Clark. 2002. Supertagging for Combinatory Categorial Grammar. In *TAG+6*.

G. B. Dantzig and P. Wolfe. 1960. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111.

M. Dreyer and J. Eisner. 2009. Graphical models over multiple strings. In *Proc. of EMNLP*.

C. Dyer and P. Resnik. 2010. Context-free reordering, finite-state translation. In *Proc. of HLT-NAACL*.

C. J. Dyer, S. Muresan, and P. Resnik. 2008. Generalizing word lattice translation. In *Proc. of ACL*.

P. F. Felzenszwalb and D. McAllester. 2007. The Generalized A* Architecture. In *Journal of Artificial Intelligence Research*, volume 29, pages 153–190.

J. R. Finkel and C. D. Manning. 2009. Joint parsing and named entity recognition. In *Proc. of NAACL*. Association for Computational Linguistics.

J. R. Finkel, C. D. Manning, and A. Y. Ng. 2006. Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proc. of EMNLP*.

J. R. Finkel, A. Kleeman, and C. D. Manning. 2008. Feature-based, conditional random field parsing. In *Proceedings of ACL-HLT*.

T. A. D. Fowler and G. Penn. 2010. Accurate context-free parsing with combinatory categorial grammar. In *Proc. of ACL*.

J. Hockenmaier and M. Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proc. of ACL*.

J. Hockenmaier and M. Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

L. Huang. 2008. Forest Reranking: Discriminative parsing with Non-Local Features. In *Proceedings of ACL-08: HLT*.

W. Jiang, L. Huang, Q. Liu, and Y. Lü. 2008. A cascaded linear model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL-08: HLT*.

N. Komodakis, N. Paragios, and G. Tziritas. 2007. MRF optimization via dual decomposition: Message-passing revisited. In *Proc. of Int. Conf. on Computer Vision (ICCV)*.

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

F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. 1998. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47:498–519.

J. K. Kummerfeld, J. Rosener, T. Dawborn, J. Haggerty, J. R. Curran, and S. Clark. 2010. Faster parsing by supertagger adaptation. In *Proc. of ACL*.

A. F. T. Martins, N. A. Smith, E. P. Xing, P. M. Q. Aguiar, and M. A. T. Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proc. of EMNLP*.

D. McAllester, M. Collins, and F. Pereira. 2008. Case-factor diagrams for structured probabilistic modeling. *Journal of Computer and System Sciences*, 74(1):84–96.

H. Mi, L. Huang, and Q. Liu. 2008. Forest-based translation. In *Proc. of ACL-HLT*.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. of ACL*.

A. M. Rush, D. Sontag, M. Collins, and T. Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *In Proc. EMNLP*.

T. Sato. 2007. Inside-outside probability computation for belief propagation. In *Proc. of IJCAI*.

D. A. Smith and J. Eisner. 2008. Dependency parsing by belief propagation. In *Proc. of EMNLP*.

P. Smyth, D. Heckerman, and M. Jordan. 1997. Probabilistic independence networks for hidden Markov probability models. *Neural computation*, 9(2):227–269.

D. Sontag, A. Globerson, and T. Jaakkola. 2010. Introduction to dual decomposition. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*. MIT Press.

M. Steedman. 2000. *The syntactic process*. MIT Press, Cambridge, MA.

C. Sutton and A. McCallum. 2005. Joint parsing and semantic role labelling. In *Proc. of CoNLL*.

C. Sutton and A. McCallum. 2010. An introduction to conditional random fields. arXiv:stat.ML/1011.4088.

J. Yedidia, W. Freeman, and Y. Weiss. 2001. Generalized belief propagation. In *Proc. of NIPS*.

Jointly Learning to Extract and Compress

Taylor Berg-Kirkpatrick **Dan Gillick** **Dan Klein**
Computer Science Division
University of California at Berkeley
{tberg, dgillick, klein}@cs.berkeley.edu

Abstract

We learn a joint model of sentence extraction and compression for multi-document summarization. Our model scores candidate summaries according to a combined linear model whose features factor over (1) the n-gram types in the summary and (2) the compressions used. We train the model using a margin-based objective whose loss captures end summary quality. Because of the exponentially large set of candidate summaries, we use a cutting-plane algorithm to incrementally detect and add active constraints efficiently. Inference in our model can be cast as an ILP and thereby solved in reasonable time; we also present a fast approximation scheme which achieves similar performance. Our jointly extracted and compressed summaries outperform both unlearned baselines and our learned extraction-only system on both ROUGE and Pyramid, without a drop in judged linguistic quality. We achieve the highest published ROUGE results to date on the TAC 2008 data set.

1 Introduction

Applications of machine learning to automatic summarization have met with limited success, and, as a result, many top-performing systems remain largely ad-hoc. One reason learning may have provided limited gains is that typical models do not learn to optimize end summary quality directly, but rather learn intermediate quantities in isolation. For example, many models learn to score each input sentence independently (Teufel and Moens, 1997; Shen et al., 2007; Schilder and Kondadadi, 2008), and then assemble extractive summaries from the top-ranked sentences in a way not incorporated into the learning process. This extraction is often done in the presence of a heuristic that limits redundancy. As another example, Yih et al. (2007) learn predictors of individual words' appearance in the references, but in isolation from the sentence selection procedure. Exceptions are Li et al. (2009) who take a max-margin approach to learning sentence values jointly, but still have ad hoc constraints to handle redundancy. One main contribution of the current paper is the direct optimization of summary quality in a single model; we find that our learned systems substantially outperform unlearned counterparts on both automatic and manual metrics.

While pure extraction is certainly simple and does guarantee some minimal readability, Lin (2003) showed that sentence compression (Knight and Marcu, 2001; McDonald, 2006; Clarke and Lapata, 2008) has the potential to improve the resulting summaries. However, attempts to incorporate compression into a summarization system have largely failed to realize large gains. For example, Zajic et al (2006) use a pipeline approach, pre-processing to yield additional candidates for extraction by applying heuristic sentence compressions, but their system does not outperform state-of-the-art purely extractive systems. Similarly, Gillick and Favre (2009), though not learning weights, do a limited form of compression jointly with extraction. They report a marginal increase in the automatic word-overlap metric ROUGE (Lin, 2004), but a decline in manual Pyramid (Nenkova and Passonneau, 2004).

A second contribution of the current work is to show a system for jointly learning to jointly compress and extract that exhibits gains in both ROUGE and content metrics over purely extractive systems. Both Martins and Smith (2009) and Woodsend and Lapata (2010) build models that jointly extract and compress, but learn scores for sentences (or phrases) using independent classifiers. Daumé III (2006)

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 481–490,
Portland, Oregon, June 19-24, 2011.

learns parameters for compression and extraction jointly using an approximate training procedure, but his results are not competitive with state-of-the-art extractive systems, and he does not report improvements on manual content or quality metrics.

In our approach, we define a linear model that scores candidate summaries according to features that factor over the n-gram types that appear in the summary and the structural compressions used to create the sentences in the summary. We train these parameters jointly using a margin-based objective whose loss captures end summary quality through the ROUGE metric. Because of the exponentially large set of candidate summaries, we use a cutting plane algorithm to incrementally detect and add active constraints efficiently. To make joint learning possible we introduce a new, manually-annotated data set of extracted, compressed sentences. Inference in our model can be cast as an integer linear program (ILP) and solved in reasonable time using a generic ILP solver; we also introduce a fast approximation scheme which achieves similar performance. Our jointly extracted and compressed summaries outperform both unlearned baselines and our learned extraction-only system on both ROUGE and Pyramid, without a drop in judged linguistic quality. We achieve the highest published comparable results (ROUGE) to date on our test set.

2 Joint Model

We focus on the task of multi-document summarization. The input is a collection of documents, each consisting of multiple sentences. The output is a summary of length no greater than L_{max}. Let x be the input document set, and let $\mathbf{y}$ be a representation of a summary as a vector. For an extractive summary, $\mathbf{y}$ is as a vector of indicators $\mathbf{y} = (y_s : s \in x)$, one indicator y_s for each sentence s in x. A sentence s is present in the summary if and only if its indicator $y_s = 1$ (see Figure 1a). Let $Y(x)$ be the set of valid summaries of document set x with length no greater than L_{max}.

While past extractive methods have assigned value to individual sentences and then explicitly represented the notion of redundancy (Carbonell and Goldstein, 1998), recent methods show greater success by using a simpler notion of coverage: bigrams

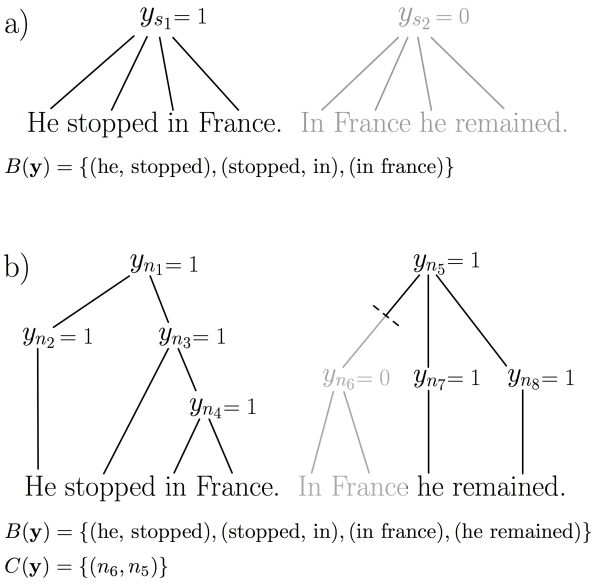


Figure 1: Diagram of (a) extractive and (b) joint extractive and compressive summarization models. Variables y_s indicate the presence of sentences in the summary. Variables y_n indicate the presence of parse tree nodes. Note that there is intentionally a bigram missing from (a).

contribute content, and redundancy is implicitly encoded in the fact that redundant sentences cover fewer bigrams (Nenkova and Vanderwende, 2005; Gillick and Favre, 2009). This later approach is associated with the following objective function:

$$\max_{\mathbf{y} \in Y(x)} \sum_{b \in B(\mathbf{y})} v_b \quad (1)$$

Here, v_b is the value of bigram b, and $B(\mathbf{y})$ is the set of bigrams present in the summary encoded by $\mathbf{y}$. Gillick and Favre (2009) produced a state-of-the-art system[1] by directly optimizing this objective. They let the value v_b of each bigram be given by the number of input documents the bigram appears in. Our implementation of their system will serve as a baseline, referred to as EXTRACTIVE BASELINE.

We extend objective 1 so that it assigns value not just to the bigrams that appear in the summary, but also to the choices made in the creation of the summary. In our complete model, which jointly extracts and compresses sentences, we choose whether or not to cut individual subtrees in the constituency parses

[1] See Text Analysis Conference results in 2008 and 2009.

of each sentence. This is in contrast to the extractive case where choices are made on full sentences.

$$\max_{\mathbf{y} \in Y(x)} \sum_{b \in B(\mathbf{y})} v_b + \sum_{c \in C(\mathbf{y})} v_c \quad (2)$$

$C(\mathbf{y})$ is the set of cut choices made in $\mathbf{y}$, and v_c assigns value to each.

Next, we present details of our representation of compressive summaries. Assume a constituency parse t_s for every sentence s. We represent a compressive summary as a vector $\mathbf{y} = (y_n : n \in t_s, s \in x)$ of indicators, one for each non-terminal node in each parse tree of the sentences in the document set x. A word is present in the output summary if and only if its parent parse tree node n has $y_n = 1$ (see Figure 1b). In addition to the length constraint on the members of $Y(x)$, we require that each node n may have $y_n = 1$ only if its parent $\pi(n)$ has $y_{\pi(n)} = 1$. This ensures that only subtrees may be deleted. While we use constituency parses rather than dependency parses, this model has similarities with the vine-growth model of Daumé III (2006).

For the compressive model we define the set of cut choices $C(\mathbf{y})$ for a summary $\mathbf{y}$ to be the set of edges in each parse that are broken in order to delete a subtree (see Figure 1b). We require that each subtree has a non-terminal node for a root, and say that an edge $(n, \pi(n))$ between a node and its parent is broken if the parent has $y_{\pi(n)} = 1$ but the child has $y_n = 0$. Notice that breaking a single edge deletes an entire subtree.

2.1 Parameterization

Before learning weights in Section 3, we parameterize objectives 1 and 2 using features. This entails to parameterizing each bigram score v_b and each subtree deletion score v_c. For weights $\mathbf{w} \in \mathbb{R}^d$ and feature functions $\mathbf{g}(b, x) \in \mathbb{R}^d$ and $\mathbf{h}(c, x) \in \mathbb{R}^d$ we let:

$$v_b = \mathbf{w}^{\mathrm{T}} \mathbf{g}(b, x)$$
$$v_c = \mathbf{w}^{\mathrm{T}} \mathbf{h}(c, x)$$

For example, $\mathbf{g}(b, x)$ might include a feature the counts the number of documents in x that b appears in, and $\mathbf{h}(c, x)$ might include a feature that indicates whether the deleted subtree is an SBAR modifying a noun.

This parameterization allows us to cast summarization as structured prediction. We can define a feature function $f(\mathbf{y}, x) \in \mathbb{R}^d$ which factors over summaries $\mathbf{y}$ through $B(\mathbf{y})$ and $C(\mathbf{y})$:

$$\mathbf{f}(\mathbf{y}, x) = \sum_{b \in B(\mathbf{y})} \mathbf{g}(b, x) + \sum_{c \in C(\mathbf{y})} \mathbf{h}(c, x)$$

Using this characterization of summaries as feature vectors we can define a linear predictor for summarization:

$$\begin{aligned} d(x; \mathbf{w}) &= \arg\max_{\mathbf{y} \in Y(x)} \mathbf{w}^{\mathrm{T}} f(\mathbf{y}, x) \quad (3) \\ &= \arg\max_{\mathbf{y} \in Y(x)} \sum_{b \in B(\mathbf{y})} v_b + \sum_{c \in C(\mathbf{y})} v_c \end{aligned}$$

The arg max in Equation 3 optimizes Objective 2.

Learning weights for Objective 1 where $Y(x)$ is the set of *extractive* summaries gives our LEARNED EXTRACTIVE system. Learning weights for Objective 2 where $Y(x)$ is the set of *compressive* summaries, and $C(\mathbf{y})$ the set of broken edges that produce subtree deletions, gives our LEARNED COMPRESSIVE system, which is our joint model of extraction and compression.

3 Structured Learning

Discriminative training attempts to minimize the loss incurred during prediction by optimizing an objective on the training set. We will perform discriminative training using a loss function that directly measures end-to-end summarization quality.

In Section 4 we show that finding summaries that optimize Objective 2, Viterbi prediction, is efficient. Online learning algorithms like perceptron or the margin-infused relaxed algorithm (MIRA) (Crammer and Singer, 2003) are frequently used for structured problems where Viterbi inference is available. However, we find that such methods are unstable on our problem. We instead turn to an approach that optimizes a batch objective which is sensitive to all constraints on all instances, but is efficient by adding these constraints incrementally.

3.1 Max-margin objective

For our problem the data set consists of pairs of document sets and label summaries, $D = \{(x_i, \mathbf{y}_i^*) : i \in 1, \ldots, N\}$. Note that the label summaries

can be expressed as vectors $\mathbf{y}^*$ because our training summaries are variously extractive or extractive and compressive (see Section 5). We use a soft-margin support vector machine (SVM) (Vapnik, 1998) objective over the full structured output space (Taskar et al., 2003; Tsochantaridis et al., 2004) of extractive and compressive summaries:

$$\min_{\mathbf{w}} \ \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i \quad (4)$$

$$s.t. \ \forall i, \forall \mathbf{y} \in Y(x_i) \quad (5)$$

$$\mathbf{w}^{\mathrm{T}}\Big(f(\mathbf{y}_i^*, x_i) - f(\mathbf{y}, x_i)\Big) \geq \ell(\mathbf{y}, \mathbf{y}_i^*) - \xi_i$$

The constraints in Equation 5 require that the difference in model score between each possible summary $\mathbf{y}$ and the gold summary $\mathbf{y}_i^*$ be no smaller than the loss $\ell(\mathbf{y}, \mathbf{y}_i^*)$, padded by a per-instance slack of ξ_i. We use bigram recall as our loss function (see Section 3.3). C is the regularization constant. When the output space $Y(x_i)$ is small these constraints can be explicitly enumerated. In this case it is standard to solve the dual, which is a quadratic program. Unfortunately, the size of the output space of extractive summaries is exponential in the number of sentences in the input document set.

3.2 Cutting-plane algorithm

The cutting-plane algorithm deals with the exponential number of constraints in Equation 5 by performing constraint induction (Tsochantaridis et al., 2004). It alternates between solving Objective 4 with a reduced set of currently active constraints, and adding newly active constraints to the set. In our application, this approach efficiently solves the structured SVM training problem up to some specified tolerance ϵ.

Suppose $\hat{\mathbf{w}}$ and $\hat{\boldsymbol{\xi}}$ optimize Objective 4 under the currently active constraints on a given iteration. Notice that the $\hat{\mathbf{y}}_i$ satisfying

$$\hat{\mathbf{y}}_i = \arg\max_{\mathbf{y} \in Y(x_i)} \left[\hat{\mathbf{w}}^{\mathrm{T}}\mathbf{f}(\mathbf{y}, x_i) + \ell(\mathbf{y}, \mathbf{y}_i^*)\right] \quad (6)$$

corresponds to the constraint in the fully constrained problem, for training instance $(x_i, \mathbf{y}_i^*)$, most violated by $\hat{\mathbf{w}}$ and $\hat{\boldsymbol{\xi}}$. On each round of constraint induction the cutting-plane algorithm computes the arg max in Equation 6 for a training instance, which is referred to as loss-augmented prediction, and adds the corresponding constraint to the active set.

The constraints from Equation (5) are equivalent to: $\forall i \ \ \mathbf{w}^{\mathrm{T}}\mathbf{f}(\mathbf{y}_i^*, x_i) \geq \max_{\mathbf{y} \in Y(x_i)} \left[\mathbf{w}^{\mathrm{T}}\mathbf{f}(\mathbf{y}, x_i) + \ell(\mathbf{y}, \mathbf{y}_i^*)\right] - \xi_i$. Thus, if loss-augmented prediction turns up no new constraints on a given iteration, the current solution to the reduced problem, $\hat{\mathbf{w}}$ and $\hat{\boldsymbol{\xi}}$, is the solution to the full SVM training problem. In practice, constraints are only added if the right hand side of Equation (5) exceeds the left hand side by at least ϵ. Tsochantaridis et al. (2004) prove that only $O(\frac{N}{\epsilon})$ constraints are added before constraint induction finds a $C\epsilon$-optimal solution.

Loss-augmented prediction is not always tractable. Luckily, our choice of loss function, bigram recall, factors over bigrams. Thus, we can easily perform loss-augmented prediction using the same procedure we use to perform Viterbi prediction (described in Section 4). We simply modify each bigram value v_b to include bigram b's contribution to the total loss. We solve the intermediate partially-constrained max-margin problems using the factored sequential minimal optimization (SMO) algorithm (Platt, 1999; Taskar et al., 2004). In practice, for $\epsilon = 10^{-4}$, the cutting-plane algorithm converges after only three passes through the training set when applied to our summarization task.

3.3 Loss function

In the simplest case, 0-1 loss, the system only receives credit for exactly identifying the label summary. Since there are many reasonable summaries we are less interested in exactly matching any specific training instance, and more interested in the degree to which a predicted summary deviates from a label.

The standard method for automatically evaluating a summary against a reference is ROUGE, which we simplify slightly to *bigram recall*. With an extractive reference denoted by $\mathbf{y}^*$, our loss function is:

$$\ell(\mathbf{y}, \mathbf{y}^*) = \frac{|B(\mathbf{y}) \bigcap B(\mathbf{y}^*)|}{|B(\mathbf{y}^*)|}$$

We verified that bigram recall correlates well with ROUGE and with manual metrics.

4 Efficient Prediction

We show how to perform prediction with the extractive and compressive models by solving ILPs. For many instances, a generic ILP solver can find exact solutions to the prediction problems in a matter of seconds. For difficult instances, we present a fast approximate algorithm.

4.1 ILP for extraction

Gillick and Favre (2009) express the optimization of Objective 1 for extractive summarization as an ILP. We begin here with their algorithm. Let each input sentence s have length l_s. Let the presence of each bigram b in $B(\mathbf{y})$ be indicated by the binary variable z_b. Let Q_{sb} be an indicator of the presence of bigram b in sentence s. They specify the following ILP over binary variables $\mathbf{y}$ and $\mathbf{z}$:

$$\max_{\mathbf{y},\mathbf{z}} \sum_b v_b z_b$$

$$s.t. \sum_s l_s y_s \leq L_{\text{max}}$$

$$\forall b \quad \sum_s Q_{sb} \leq z_b \tag{7}$$

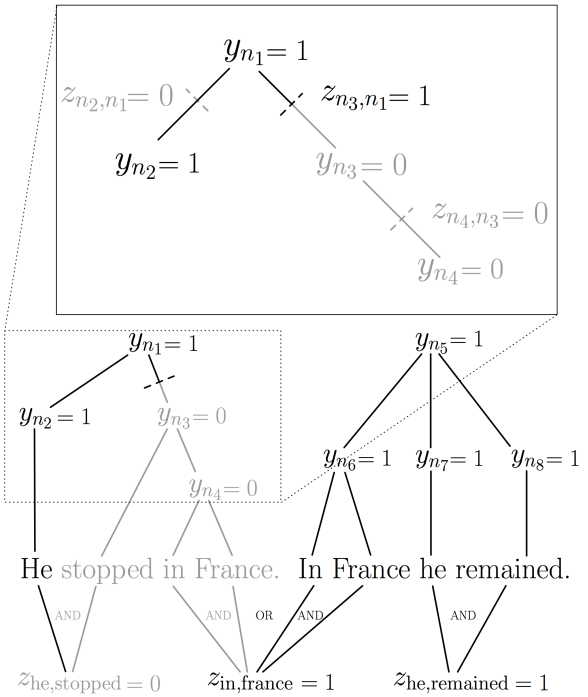
$$\forall s, b \quad y_s Q_{sb} \geq z_b \tag{8}$$

Constraints 7 and 8 ensure consistency between sentences and bigrams. Notice that the Constraint 7 requires that selecting a sentence entails selecting all its bigrams, and Constraint 8 requires that selecting a bigram entails selecting at least one sentence that contains it. Solving the ILP is fast in practice. Using the GNU Linear Programming Kit (GLPK) on a 3.2GHz Intel machine, decoding took less than a second on most instances.

4.2 ILP for joint compression and extraction

We can extend the ILP formulation of extraction to solve the compressive problem. Let l_n be the number of words node n has as children. With this notation we can write the length restriction as $\sum_n l_n y_n \leq L_{\text{max}}$. Let the presence of each cut c in $C(\mathbf{y})$ be indicated by the binary variable z_c, which is active if and only if $y_n = 0$ but $y_{\pi(n)} = 1$, where node $\pi(n)$ is the parent of node n. The constraints on z_c are diagrammed in Figure 2.

While it is possible to let $B(\mathbf{y})$ contain all bigrams present in the compressive summary, the reduction of $B(\mathbf{y})$ makes the ILP formulation efficient. We omit from $B(\mathbf{y})$ bigrams that are the result of deleted intermediate words. As a result the required number of variables z_b is linear in the length of a sentence. The constraints on z_b are given in Figure 2. They can be expressed in terms of the variables y_n.



$B(\mathbf{y}) = \{(\text{in, france}), (\text{france, he}), (\text{he, remained})\}$
$C(\mathbf{y}) = \{(n_3, n_1)\}$

Figure 2: Diagram of ILP for joint extraction and compression. Variables z_b indicate the presence of bigrams in the summary. Variables z_c indicate edges in the parse tree that have been cut in order to remove subtrees. The figure suppresses bigram variables $z_{\text{stopped,in}}$ and $z_{\text{france,he}}$ to reduce clutter. Note that the edit shown is intentionally bad. It demonstrates a loss of bigram coverage.

By solving the following ILP we can compute the arg max required for prediction in the joint model:

$$\max_{\mathbf{y},\mathbf{z}} \sum_b v_b z_b + \sum_c v_c z_c$$

$$s.t. \sum_n l_n y_n \leq L_{\text{max}}$$

$$\forall n \;\; y_n \leq y_{\pi(n)} \tag{9}$$

$$\forall b \;\; z_b = \mathbb{1}\big[b \in B(\mathbf{y})\big] \tag{10}$$

$$\forall c \;\; z_c = \mathbb{1}\big[c \in C(\mathbf{y})\big] \tag{11}$$

Constraint 9 encodes the requirement that only full subtrees may be deleted. For simplicity, we have written Constraints 10 and 11 in implicit form. These constraints can be encoded explicitly using $O(N)$ linear constraints, where N is the number of words in the document set x. The reduction of $B(\mathbf{y})$ to include only bigrams not resulting from deleted intermediate words avoids $O(N^2)$ required constraints.

In practice, solving this ILP for joint extraction and compression is, on average, an order of magnitude slower than solving the ILP for pure extraction, and for certain instances finding the exact solution is prohibitively slow.

4.3 Fast approximate prediction

One common way to quickly approximate an ILP is to solve its LP relaxation (and round the results). We found that, while very fast, the LP relaxation of the joint ILP gave poor results, finding unacceptably suboptimal solutions. This appears possibly to have been problematic for Martins and Smith (2009) as well. We developed an alternative fast approximate joint extractive and compressive solver that gives better results in terms of both objective value and bigram recall of resulting solutions.

The approximate joint solver first extracts a subset of the sentences in the document set that total no more than M words. In a second step, we apply the exact joint extractive and compressive summarizer (see Section 4.2) to the resulting extraction. The objective we maximize in performing the initial extraction is different from the one used in extractive summarization. Specifically, we pick an extraction that maximizes $\sum_{s \in \mathbf{y}} \sum_{b \in s} v_b$. This objective rewards redundant bigrams, and thus is likely to give the joint solver multiple options for including the same piece of relevant content.

M is a parameter that trades-off between approximation quality and problem difficulty. When M is the size of the document set x, the approximate solver solves the exact joint problem. In Figure 3 we plot the trade-off between approximation quality and computation time, comparing to the exact joint solver, an exact solver that is limited to extractive solutions, and the LP relaxation solver. The results show that the approximate joint solver yields substantial improvements over the LP relaxation, and can achieve results comparable to those produced by the exact solver with a 5-fold reduction in computation time. On particularly difficult instances the parameter M can be decreased, ensuring that all instances are solved in a reasonable time period.

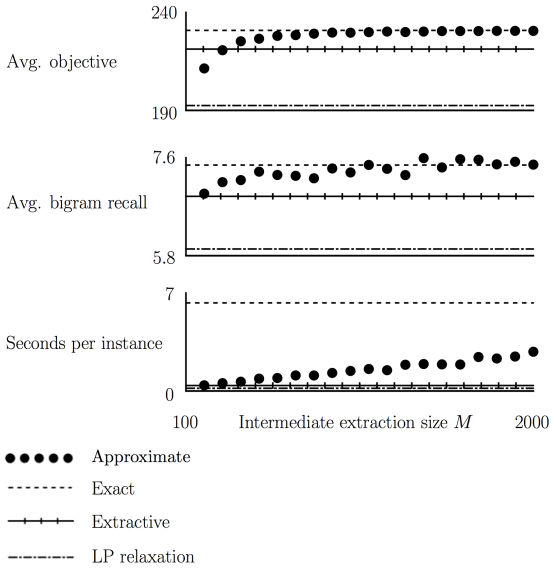


Figure 3: Plot of objective value, bigram recall, and elapsed time for the approximate joint extractive and compressive solver against size of intermediate extraction set. Also shown are values for an LP relaxation approximate solver, a solver that is restricted to extractive solutions, and finally the exact compressive solver. These solvers do not use an intermediate extraction. Results are for 44 document sets, averaging about 5000 words per document set.

5 Data

We use the data from the Text Analysis Conference (TAC) evaluations from 2008 and 2009, a total of 92 multi-document summarization problems. Each problem asks for a 100-word-limited summary of 10 related input documents and provides a set of four abstracts written by experts. These are the non-update portions of the TAC 2008 and 2009 tasks.

To train the extractive system described in Section 2, we use as our labels $\mathbf{y}^*$ the extractions with the largest bigram recall values relative to the sets of references. While these extractions are inferior to the abstracts, they are attainable by our model, a quality found to be advantageous in discriminative training for machine translation (Liang et al., 2006;

COUNT:	$\mathbb{1}(\text{docCount}(b) = \cdot)$ where $\text{docCount}(b)$ is the number of documents containing b.
STOP:	$\mathbb{1}(\text{isStop}(b_1) = \cdot, \text{isStop}(b_2) = \cdot)$ where $\text{isStop}(w)$ indicates a stop word.
POSITION:	$\mathbb{1}(\text{docPosition}(b) = \cdot)$ where $\text{docPosition}(b)$ is the earliest position in a document of any sentence containing b, buckets earliest positions ≥ 4.
CONJ:	All two- and three-way conjunctions of COUNT, STOP, and POSITION features.
BIAS:	Bias feature, active on all bigrams.

Table 1: **Bigram features**: component feature functions in $\mathbf{g}(b, x)$ that we use to characterize the bigram b in both the extractive and compressive models.

Chiang et al., 2008).

Previous work has referred to the lack of extracted, compressed data sets as an obstacle to joint learning for summarizaiton (Daumé III, 2006; Martins and Smith, 2009). We collected joint data via a Mechanical Turk task. To make the joint annotation task more feasible, we adopted an approximate approach that closely matches our fast approximate prediction procedure. Annotators were shown a 150-word maximum bigram recall extractions from the full document set and instructed to form a compressed summary by deleting words until 100 or fewer words remained. Each task was performed by two annotators. We chose the summary we judged to be of highest quality from each pair to add to our corpus. This gave one gold compressive summary $\mathbf{y}^*$ for each of the 44 problems in the TAC 2009 set. We used these labels to train our joint extractive and compressive system described in Section 2. Of the 288 total sentences presented to annotators, 38 were unedited, 45 were deleted, and 205 were compressed by an average of 7.5 words.

6 Features

Here we describe the features used to parameterize our model. Relative to some NLP tasks, our feature sets are small: roughly two hundred features on bigrams and thirteen features on subtree deletions. This is because our data set is small; with only 48 training documents we do not have the statistical support to learn weights for more features. For larger training sets one could imagine lexicalized versions of the features we describe.

COORD:	Indicates phrase involved in coordination. Four versions of this feature: NP, VP, S, SBAR.
S-ADJUNCT:	Indicates a child of an S, adjunct to and left of the matrix verb. Four version of this feature: CC, PP, ADVP, SBAR.
REL-C:	Indicates a relative clause, SBAR modifying a noun.
ATTR-C:	Indicates a sentence-final attribution clause, e.g. 'the senator announced Friday.'
ATTR-PP:	Indicates a PP attribution, e.g. 'according to the senator.'
TEMP-PP:	Indicates a temporal PP, e.g. 'on Friday.'
TEMP-NP:	Indicates a temporal NP, e.g. 'Friday.'
BIAS:	Bias feature, active on all subtree deletions.

Table 2: **Subtree deletion features**: component feature functions in $\mathbf{h}(c, x)$ that we use to characterize the subtree deleted by cutting edge $c = (n, \pi(n))$ in the joint extractive and compressive model.

6.1 Bigram features

Our bigram features include document counts, the earliest position in a document of a sentence that contains the bigram, and membership of each word in a standard set of stopwords. We also include all possible two- and three-way conjuctions of these features. Table 1 describes the features in detail. We use stemmed bigrams and prune bigrams that appear in fewer than three input documents.

6.2 Subtree deletion features

Table 2 gives a description of our subtree tree deletion features. Of course, by training to optimize a metric like ROUGE, the system benefits from restrictions on the syntactic variety of edits; the learning is therefore more about deciding when an edit is worth the coverage trade-offs rather than fine-grained decisions about grammaticality.

We constrain the model to only allow subtree deletions where one of the features in Table 2 (aside from BIAS) is active. The root, and thus the entire sentence, may always be cut. We choose this particular set of allowed deletions by looking at human annotated data and taking note of the most common types of edits. Edits which are made rarely by humans should be avoided in most scenarios, and we simply don't have enough data to learn when to do them safely.

System	BR	R-2	R-SU4	Pyr	LQ
LAST DOCUMENT	4.00	5.85	9.39	23.5	**7.2**
EXT. BASELINE	6.85	10.05	13.00	35.0	6.2
LEARNED EXT.	7.43	11.05	13.86	38.4	6.6
LEARNED COMP.	**7.75**	**11.70**	**14.38**	**41.3**	6.5

Table 3: Bigram Recall (BR), ROUGE (R-2 and R-SU4) and Pyramid (Pyr) scores are multiplied by 100; Linguistic Quality (LQ) is scored on a 1 (very poor) to 10 (very good) scale.

System	Sents	Words/Sent	Word Types
LAST DOCUMENT	4.0	25.0	36.5
EXT. BASELINE	5.0	20.8	36.3
LEARNED EXT.	4.8	21.8	37.1
LEARNED COMP.	4.5	22.9	38.8

Table 4: Summary statistics for the summaries generated by each system: Average number of sentences per summary, average number of words per summary sentence, and average number of non-stopword word types per summary.

7 Experiments

7.1 Experimental setup

We set aside the TAC 2008 data set (48 problems) for testing and use the TAC 2009 data set (44 problems) for training, with hyper-parameters set to maximize six-fold cross-validation bigram recall on the training set. We run the factored SMO algorithm until convergence, and run the cutting-plane algorithm until convergence for $\epsilon = 10^{-4}$. We used GLPK to solve all ILPs. We solved extractive ILPs exactly, and joint extractive and compressive ILPs approximately using an intermediate extraction size of 1000. Constituency parses were produced using the Berkeley parser (Petrov and Klein, 2007). We show results for three systems, EXTRACTIVE BASELINE, LEARNED EXTRACTIVE, LEARNED COMPRESSIVE, and the standard baseline that extracts the first 100 words in the the most recent document, LAST DOCUMENT.

7.2 Results

Our evaluation results are shown in Table 3. ROUGE-2 (based on bigrams) and ROUGE-SU4 (based on both unigrams and skip-bigrams, separated by up to four words) are given by the official ROUGE toolkit with the standard options (Lin, 2004).

Pyramid (Nenkova and Passonneau, 2004) is a manually evaluated measure of recall on facts or *Semantic Content Units* appearing in the reference summaries. It is designed to help annotators distinguish information content from linguistic quality. Two annotators performed the entire evaluation without overlap by splitting the set of problems in half.

To evaluate linguistic quality, we sent all the summaries to Mechanical Turk (with two times redundancy), using the template and instructions designed by Gillick and Liu (2010). They report that Turkers can faithfully reproduce experts' rankings of average system linguistic quality (though their judgements of content are poorer). The table shows average linguistic quality.

All the content-based metrics show substantial improvement for learned systems over unlearned ones, and we see an extremely large improvement for the learned joint extractive and compressive system over the previous state-of-the-art EXTRACTIVE BASELINE. The ROUGE scores for the learned joint system, LEARNED COMPRESSIVE, are, to our knowledge, the highest reported on this task. We cannot compare Pyramid scores to other reported scores because of annotator difference. As expected, the LAST DOCUMENT baseline outperforms other systems in terms of linguistic quality. But, importantly, the gains achieved by the joint extractive and compressive system in content-based metrics do not come at the cost of linguistic quality when compared to purely extractive systems.

Table 4 shows statistics on the outputs of the systems we evaluated. The joint extractive and compressive system fits more word types into a summary than the extractive systems, but also produces longer sentences on average. Reading the output summaries more carefully suggests that by learning to extract and compress jointly, our joint system has the flexibility to use or create reasonable, medium-length sentences, whereas the extractive systems are stuck with a few valuable long sentences, but several less productive shorter sentences. Example summaries produced by the joint system are given in Figure 4 along with reference summaries produced by humans.

LEARNED COMPRESSIVE: The country's work safety authority will release the list of the first batch of coal mines to be closed down said Wang Xianzheng, deputy director of the National Bureau of Production Safety Supervision and Administration. With its coal mining safety a hot issue, attracting wide attention from both home and overseas, China is seeking solutions from the world to improve its coal mining safety system. Despite government promises to stem the carnage the death toll in China's disaster-plagued coal mine industry is rising according to the latest statistics released by the government Friday. Fatal coal mine accidents in China rose 8.5 percent in the first eight months of this year with thousands dying despite stepped-up efforts to make the industry safer state media said Wednesday.

REFERENCE: China's accident-plagued coal mines cause thousands of deaths and injuries annually. 2004 saw over 6,000 mine deaths. January through August 2005, deaths rose 8.5% over the same period in 2004. Most accidents are gas explosions, but fires, floods, and cave-ins also occur. Ignored safety procedures, outdated equipment, and corrupted officials exacerbate the problem. Official responses include shutting down thousands of ill-managed and illegally-run mines, punishing errant owners, issuing new safety regulations and measures, and outlawing local officials from investing in mines. China also sought solutions at the Conference on South African Coal Mining Safety Technology and Equipment held in Beijing.

LEARNED COMPRESSIVE: Karl Rove the White House deputy chief of staff told President George W. Bush and others that he never engaged in an effort to disclose a CIA operative's identity to discredit her husband's criticism of the administration's Iraq policy according to people with knowledge of Rove's account in the investigation. In a potentially damaging sign for the Bush administration special counsel Patrick Fitzgerald said that although his investigation is nearly complete it's not over. Lewis Scooter Libby Vice President Dick Cheney's chief of staff and a key architect of the Iraq war was indicted Friday on felony charges of perjury making false statements to FBI agents and obstruction of justice for impeding the federal grand jury investigating the CIA leak case.

REFERENCE: Special Prosecutor Patrick Fitzgerald is investigating who leaked to the press that Valerie Plame, wife of former Ambassador Joseph Wilson, was an undercover CIA agent. Wilson was a critic of the Bush administration. Administration staffers Karl Rove and I. Lewis Libby are the focus of the investigation. NY Times correspondent Judith Miller was jailed for 85 days for refusing to testify about Libby. Libby was eventually indicted on five counts: 2 false statements, 1 obstruction of justice, 2 perjury. Libby resigned immediately. He faces 30 years in prison and a fine of $1.25 million if convicted. Libby pleaded not guilty.

Figure 4: Example summaries produced by our learned joint model of extraction and compression. These are each 100-word-limited summaries of a collection of ten documents from the TAC 2008 data set. Constituents that have been removed via subtree deletion are grayed out. References summaries produced by humans are provided for comparison.

8 Conclusion

Jointly learning to extract and compress within a unified model outperforms learning pure extraction, which in turn outperforms a state-of-the-art extractive baseline. Our system gives substantial increases in both automatic and manual content metrics, while maintaining high linguistic quality scores.

Acknowledgements

We thank the anonymous reviewers for their comments. This project is supported by DARPA under grant N10AP20007.

References

J. Carbonell and J. Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proc. of SIGIR*.

D. Chiang, Y. Marton, and P. Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proc. of EMNLP*.

J. Clarke and M. Lapata. 2008. Global Inference for Sentence Compression: An Integer Linear Programming Approach. *Journal of Artificial Intelligence Research*, 31:399–429.

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

H.C. Daumé III. 2006. *Practical structured learning techniques for natural language processing*. Ph.D. thesis, University of Southern California.

D. Gillick and B. Favre. 2009. A scalable global model for summarization. In *Proc. of ACL Workshop on Integer Linear Programming for Natural Language Processing*.

D. Gillick and Y. Liu. 2010. Non-Expert Evaluation of Summarization Systems is Risky. In *Proc. of NAACL Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

K. Knight and D. Marcu. 2001. Statistics-based summarization-step one: Sentence compression. In *Proc. of AAAI*.

L. Li, K. Zhou, G.R. Xue, H. Zha, and Y. Yu. 2009. Enhancing diversity, coverage and balance for summarization through structure learning. In *Proc. of the 18th International Conference on World Wide Web*.

P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. of the ACL*.

C.Y. Lin. 2003. Improving summarization performance by sentence compression: a pilot study. In *Proc. of ACL Workshop on Information Retrieval with Asian Languages*.

C.Y. Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc. of ACL Workshop on Text Summarization Branches Out*.

A.F.T. Martins and N.A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proc. of NAACL Workshop on Integer Linear Programming for Natural Language Processing*.

R. McDonald. 2006. Discriminative sentence compression with soft syntactic constraints. In *Proc. of EACL*.

A. Nenkova and R. Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Proc. of NAACL*.

A. Nenkova and L. Vanderwende. 2005. The impact of frequency on summarization. Technical report, MSR-TR-2005-101. Redmond, Washington: Microsoft Research.

S. Petrov and D. Klein. 2007. Learning and inference for hierarchically split PCFGs. In *AAAI*.

J.C. Platt. 1999. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods*. MIT press.

F. Schilder and R. Kondadadi. 2008. Fastsum: Fast and accurate query-based multi-document summarization. In *Proc. of ACL*.

D. Shen, J.T. Sun, H. Li, Q. Yang, and Z. Chen. 2007. Document summarization using conditional random fields. In *Proc. of IJCAI*.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *Proc. of NIPS*.

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

S. Teufel and M. Moens. 1997. Sentence extraction as a classification task. In *Proc. of ACL Workshop on Intelligent and Scalable Text Summarization*.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. of ICML*.

V.N. Vapnik. 1998. *Statistical learning theory*. John Wiley and Sons, New York.

K. Woodsend and M. Lapata. 2010. Automatic generation of story highlights. In *Proc. of ACL*.

W. Yih, J. Goodman, L. Vanderwende, and H. Suzuki. 2007. Multi-document summarization by maximizing informative content-words. In *Proc. of IJCAI*.

D.M. Zajic, B.J. Dorr, R. Schwartz, and J. Lin. 2006. Sentence compression as a component of a multi-document summarization system. In *Proc. of the 2006 Document Understanding Workshop*.

Discovery of Topically Coherent Sentences for Extractive Summarization

Asli Celikyilmaz
Microsoft Speech Labs
Mountain View, CA, 94041
asli@ieee.org

Dilek Hakkani-Tür
Microsoft Speech Labs | Microsoft Research
Mountain View, CA, 94041
dilek@ieee.org

Abstract

Extractive methods for multi-document summarization are mainly governed by information overlap, coherence, and content constraints. We present an unsupervised probabilistic approach to model the hidden abstract concepts across documents as well as the correlation between these concepts, to generate topically coherent and non-redundant summaries. Based on human evaluations our models generate summaries with higher linguistic quality in terms of coherence, readability, and redundancy compared to benchmark systems. Although our system is unsupervised and optimized for topical coherence, we achieve a 44.1 ROUGE on the DUC-07 test set, roughly in the range of state-of-the-art supervised models.

1 Introduction

A query-focused multi-document summarization model produces a short-summary text of a set of documents, which are retrieved based on a user's query. An ideal generated summary text should contain the shared relevant content among set of documents *only once*, plus other unique information from individual documents that are directly related to the user's query addressing different levels of *detail*. Recent approaches to the summarization task has somewhat focused on the *redundancy* and *coherence* issues. In this paper, we introduce a series of new generative models for multiple-documents, based on a discovery of hierarchical topics and their correlations to extract topically coherent sentences.

Prior research has demonstrated the usefulness of sentence extraction for generating summary text taking advantage of surface level features such as word repetition, position in text, cue phrases, etc, (Radev, 2004; Nenkova and Vanderwende, 2005a; Wan and Yang, 2006; Nenkova et al., 2006). Because documents have pre-defined structures (e.g., sections, paragraphs, sentences) for different levels of concepts in a hierarchy, most recent summarization work has focused on structured probabilistic models to represent the corpus concepts (Barzilay et al., 1999; Daumé-III and Marcu, 2006; Eisenstein and Barzilay, 2008; Tang et al., 2009; Chen et al., 2000; Wang et al., 2009). In particular (Haghighi and Vanderwende, 2009; Celikyilmaz and Hakkani-Tur, 2010) build hierarchical topic models to identify salient sentences that contain abstract concepts rather than specific concepts. Nonetheless, all these systems crucially rely on extracting various levels of generality from documents, focusing little on redundancy and coherence issues in model building. A model than can focus on both issues is deemed to be more beneficial for a summarization task.

Topical coherence in text involves identifying key concepts, the relationships between these concepts, and linking these relationships into a hierarchy. In this paper, we present a novel, fully generative Bayesian model of document corpus, which can discover topically coherent sentences that contain key shared information with as little detail and redundancy as possible. Our model can discover hierarchical latent structure of multi-documents, in which some words are governed by low-level topics (T) and others by high-level topics (H). The main contributions of this work are:

− construction of a novel bayesian framework to

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 491–499,
Portland, Oregon, June 19-24, 2011.

capture higher level topics (concepts) related to summary text discussed in §3,
– representation of a linguistic system as a sequence of increasingly enriched models, which use posterior topic correlation probabilities in sentences to design a novel sentence ranking method in §4 and 5,
– application of the new hierarchical learning method for generation of less redundant summaries discussed in §6. Our models achieve comparable qualitative results on summarization of multiple newswire documents. Human evaluations of generated summaries confirm that our model can generate non-redundant and topically coherent summaries.

2 Multi-Document Summarization Models

Prior research has demonstrated the usefulness of sentence extraction for summarization based on lexical, semantic, and discourse constraints. Such models often rely on different approaches including: identifying important keywords (Nenkova et al., 2006); topic signatures based on user queries (Lin and Hovy, 2002; Conroy et al., 2006; Harabagiu et al., 2007); high frequency content word feature based learning (Nenkova and Vanderwende, 2005a; Nenkova and Vanderwende, 2005b), to name a few.

Recent research focusing on the extraction of latent concepts from document clusters are close in spirit to our work (Barzilay and Lee, 2004; Daumé-III and Marcu, 2006; Eisenstein and Barzilay, 2008; Tang et al., 2009; Wang et al., 2009). Some of these work (Haghighi and Vanderwende, 2009; Celikyilmaz and Hakkani-Tur, 2010) focus on the discovery of hierarchical concepts from documents (from abstract to specific) using extensions of hierarchal topic models (Blei et al., 2004) and reflect this hierarchy on the sentences. Hierarchical concept learning models help to discover, for instance, that "baseball" and "football" are both contained in a general class "sports", so that the summaries reference terms related to more abstract concepts like "sports".

Although successful, the issue with concept learning methods for summarization is that the extracted sentences usually contain correlated concepts. We need a model that can identify salient sentences referring to general concepts of documents and there should be minimum correlation between them.

Our approach differs from the early work, in that, we utilize the advantages of previous topic models and build an unsupervised generative model that can associate each word in each document with three random variables: a sentence $\boldsymbol{S}$, a higher-level topic $\boldsymbol{H}$, and a lower-level topic $\boldsymbol{T}$, in an analogical way to PAM models (Li and McCallum, 2006), i.e., a directed acyclic graph (DAG) representing mixtures of hierarchical structure, where super-topics are multinomials over sub-topics at lower levels in the DAG. We define a tiered-topic clustering in which the upper nodes in the DAG are *higher-level topics* $\boldsymbol{H}$, representing common co-occurence patterns (correlations) between lower-level topics $\boldsymbol{T}$ in documents. This has not been the focus in prior work on generative approaches for summarization task. Mainly, our model can discover correlated topics to eliminate redundant sentences in summary text.

Rather than representing sentences as a layer in hierarchical models, e.g., (Haghighi and Vanderwende, 2009; Celikyilmaz and Hakkani-Tur, 2010), we model sentences as *meta-variables*. This is similar to author-topic models (Rosen-Zvi et al., 2004), in which words are generated by first selecting an author uniformly from an observed author list and then selecting a topic from a distribution over topics that is specific to that author. In our model, words are generated from different topics of documents by first selecting a sentence containing the word and then topics that are specific to that sentence. This way we can directly extract from documents the summary related sentences that contain high-level topics. In addition in (Celikyilmaz and Hakkani-Tur, 2010), the sentences can only share topics if the sentences are represented on the same path of captured topic hierarchy, restricting topic sharing across sentences on different paths. Our DAG identifies tiered topics distributed over document clusters that can be shared by each sentence.

3 Topic Coherence for Summarization

In this section we discuss the main contribution, our two hierarchical mixture models, which improve summary generation performance through the use of tiered topic models. Our models can identify lower-level topics T (concepts) defined as distributions over words or higher-level topics H, which represent correlations between these lower level topics given

sentences. We present our synthetic experiment for model development to evaluate extracted summaries on redundancy measure. In §6, we demonstrate the performance of our models on coherence and informativeness of generated summaries by qualitative and intrinsic evaluations.

For model development we use the DUC 2005 dataset[1], which consists of 45 document clusters, each of which include 1-4 set of human generated summaries (10-15 sentences each). Each document cluster consists $\sim$ 25 documents (25-30 sentences/document) retrieved based on a user query. We consider each document cluster as a corpus and build 45 separate models.

For the synthetic experiments, we include the provided human generated summaries of each corpus as additional documents. The sentences in human summaries include general concepts mentioned in the corpus, the salient sentences of documents. Contrary to usual qualitative evaluations of summarization tasks, our aim during development is to measure the percentage of sentences in a human summary that our model can identify as salient among all other document cluster sentences. Because human produced summaries generally contain non-redundant sentences, we use total number of top-ranked human summary sentences as a qualitative redundancy measure in our synthetic experiments.

In each model, a document d is a vector of N_d words $\mathbf{w}_d$, where each w_{id} is chosen from a vocabulary of size V, and a vector of sentences $\mathbf{S}$, representing all sentences in a corpus of size S_D. We identify sentences as meta-variables of document clusters, which the generative process models both sentences and documents using *tiered* topics. A sentence's relatedness to summary text is tied to the document cluster's user query. The idea is that a lexical word present or related to a query should increase its sentence's probability of relatedness.

4 Two-Tiered Topic Model - TTM

Our base model, the two-tiered topic model (TTM), is inspired by the hierarchical topic model, PAM, proposed by Li and McCallum (2006). PAM structures documents to represent and learn arbitrary, nested, and possibly sparse topic correlations using a directed acyclic graph. Our goals are not so different: we aim to discover concepts from documents that would attribute for the general topics related to a user query, however, we want to relate this information to sentences. We represent sentences $\boldsymbol{S}$ by discovery of general (more general) to specific topics (Fig.1). Similarly, we represent summary unrelated (document specific) sentences as corpus specific distributions θ over background words $\mathbf{w_B}$, (functional words like prepositions, etc.).

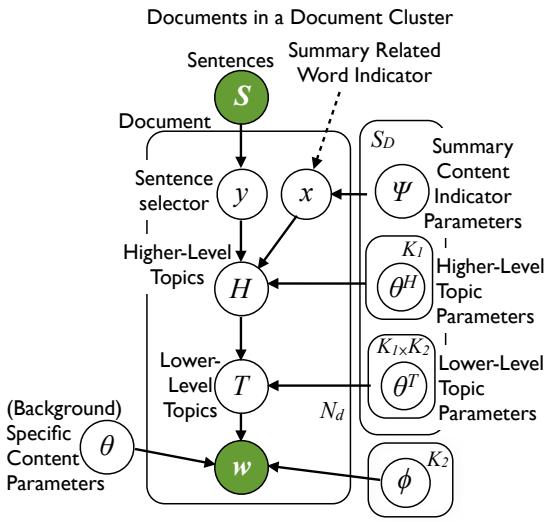


Figure 1: Graphical model depiction of two-tiered topic model (TTM) described in section §4. $\mathbf{S}$ are sentences $s_{i=1..S_D}$ in document clusters. The high-level topics ($H_{k_1=1...K_1}$), representing topic correlations, are modeled as distributions over low-level-topics ($T_{k_2=1...K_2}$). Shaded nodes indicate observed variables. Hyper-parameters for ϕ, θ^H, θ^T, θ are omitted.

Our two-tiered topic model for salient sentence discovery can be generated for each word in the document (Algorithm 1) as follows: For a word w_{id} in document d, a random variable x_{id} is drawn, which determines if w_{id} is query related, i.e., w_{id} either exists in the query or is related to the query[2]. Otherwise, w_{id} is unrelated to the user query. Then sentence s_i is chosen uniformly at random ($y_{s_i} \sim Uniform(s_i)$) from sentences in the document containing w_{id} (deterministic if there is only one sentence containing w_{id}). We assume that if a word is related to a query, it is likely to be summary-related

[1] www-nlpir.nist.gov/projects/duc/data.html

[2] We measure relatedness to a query if a word exists in the query or it is synonymous based on information extracted from WordNet (Miller, 1995).

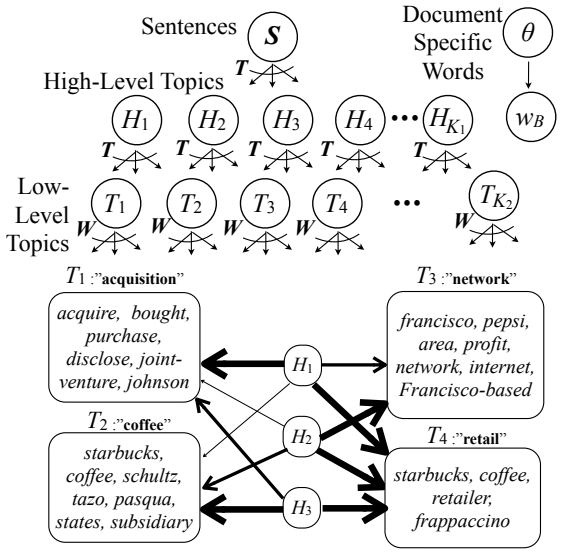


Figure 2: Depiction of TTM given the query *"D0718D: Starbucks Coffee :* ***How has Starbucks Coffee attempted to expand and diversify through joint ventures, acquisitions, or subsidiaries?"***. If a word is query/summary related sentence S, first a sentence then a high-level (H) and a low-level (T) topic is sampled. ($\overset{C}{\curlywedge}$ represents that a random variable is a parent of all C random variables.) The bolded links from $H-T$ represent correlated low-level topics.

Algorithm 1 Two-Tiered Topic Model Generation

Sample: $s_i = 1..S_D$: $\Psi \sim Beta(\eta)$,
$k_1 = 1...K_1$: $\theta^H \sim Dirichlet(\alpha^H)$,
$k_2 = 1...K_1 \times K_2$: $\theta^T \sim Dirichlet(\alpha^T)$,
and $k = 1..K_2$: $\phi \sim Dirichlet(\beta)$.
for documents $d \leftarrow 1, ..., D$ **do**
  for words w_{id}, $i \leftarrow 1, ..., N_d$ **do**
    - Draw a discrete $x \sim Binomial(\Psi_{w_{id}})^{\star}$
    - If $x = 1$, w_{id} is summary related;
      · conditioned on S draw a sentence
        $y_{s_i} \sim Uniform(s_i)$ containing w_i,
      · sample a high-level topic $H_{k_1} \sim \theta^H_{k_1}(\alpha^H)$,
        and a low-level topic $T_{k_2} \sim \theta^T_{k_2}(\alpha^T)$,
      · sample a word $w_{ik_1k_2} \sim \phi_{H_{k_1}T_{k_2}}(\alpha)$,
    - If $x = 0$, the word is unrelated **
      sample a word $w_B \sim \theta(\alpha)$,
      corpus specific distribution.
  end for
end for

$\star$ if w_{id} exists or related to the the query then $x = 1$ deterministic, otherwise it is stochastically assigned $x \sim Bin(\Psi)$.
$\star\star$ w_{id} is a background word.

(so as the sampled sentence s_i). We keep track of the frequency of s_i's in a vector, $DS \in Z^{S_D}$. Every time an s_i is sampled for a query related w_{id}, we increment its count, a degree of sentence saliency.

Given that w_{id} is related to a query, it is associated with two-tiered multinomial distributions: high-level *H* topics and low-level *T* topics. A high-level topic H_{k_i} is chosen first from a distribution over low-level topics *T* specific to that s_i and one low-level topic T_{k_j} is chosen from a distribution over words, and w_{id} is generated from the sampled low-level topic. If w_{id} is *not* query-related, it is generated as a background word w_B.

The resulting tiered model is shown as a graph and plate diagrams in Fig.1 & 2. A sentence sampled from a query related word is associated with a distribution over K_1 number of high-level topics H_{k_i}, each of which are also associated with K_2 number of low-level topics T_{k_j}, a multinomial over lexical words of a corpus. In Fig.2 the most confident words of four low-level topics is shown. The bolded links between H_{k_i} and T_{k_j} represent the strength of correlation between T_{k_j}'s, e.g., the topic *"acquisition"* is found to be more correlated with *"retail"* than the "network" topic given H_1. This information is used to rank sentences based on the correlated topics.

4.1 Learning and Inference for TTM

Our learning procedure involves finding parameters, which likely integrates out model's posterior distribution $P(\mathbf{H}, \mathbf{T}|\mathbf{W_d}, \mathbf{S})$, $d \in D$. EM algorithms might face problems with local maxima in topic models (Blei et al., 2003) suggesting implementation of approximate methods in which some of the parameters, e.g., θ^H, θ^T, ψ, and θ, can be integrated out, resulting in standard Dirichlet-multinomial as well as binomial distributions. We use Gibbs sampling which allows a combination of estimates from several local maxima of the posterior distribution.

For each word, x_{id} is sampled from a sentence specific binomial ψ which in turn has a smoothing prior η to determine if the sampled word w_{id} is (query) summary-related or document-specific. Depending on x_{id}, we either sample a sentence along with a high/low-level topic pair or just sample background words w_B. The probability distribution over sentence assignments, $P(y_{s_i} = s|\mathbf{S})$ $s_i \in \mathbf{S}$, is assumed to be uniform over the elements of **S**, and deterministic if there is only one sentence in the docu-

ment containing the corresponding word. The optimum hyper-parameters are set based on the training dataset model performance via cross-validation [3].

For each word we sample a high-level H_{k_i} and a low-level T_{k_j} topic if the word is query related ($x_{id} = 1$). The sampling distribution for TTM for a word given the remaining topics and hyper-parameters $\alpha^H, \alpha^T, \alpha, \beta, \eta$ is:

$$p_{\texttt{TTM}}(H_{k_1}, T_{k_2}, x = 1 | \mathbf{w}, \mathbf{H}_{-k_1}, \mathbf{T}_{-k_2}) \propto \frac{\alpha^H + n_d^{k_1}}{\sum_{H\prime} \alpha^{H\prime} + n_d} * \frac{\alpha^T + n_d^{k_1 k_2}}{\sum_{T\prime} \alpha^{T\prime} + n_d{}^H} * \frac{\eta + n_x^{k_1 k_2}}{2\eta + n_{k_1 k_2}} * \frac{\beta_w + n^w_{k_1 k_2 x}}{\sum_{w\prime} \beta_{w\prime} + n_{k_1 k_2 x}}$$

and when $x = 0$ (a corpus specific word),

$$p_{\texttt{TTM}}(x = 0 | \mathbf{w}, \mathbf{z}_{H-k}, \mathbf{z}_{t-k}) \propto \frac{\eta + n^x_{k_1 k_2}}{2\eta + n_{k_1 k_2}} * \frac{\alpha_w + n^w}{\sum_{w\prime} \alpha_{w\prime} + n}$$

The $n_d^{k_1}$ is the number of occurrences of high-level topic k_1 in document d, and $n_d^{k_1 k_2}$ is the number of times the low-level topic k_2 is sampled together with high-level topic k_1 in d, $n^w_{k_1 k_2 x}$ is the number of occurrences of word w sampled from path H-T given that the word is query related. Note that the number of tiered topics in the model is fixed to K_1 and K_2, which is optimized with validation experiments. It is also possible to construct extended models of TTM using non-parametric priors, e.g., hierarchal Dirichlet processes (Li et al., 2007) (left for future work).

4.2 Summary Generation with TTM

We can observe the frequency of draws of every sentence in a document cluster **S**, given it's words are related, through $DS \in \mathbb{Z}^{S_D}$. We obtain DS during Gibbs sampling (in §4.1), which indicates a saliency score of each sentence $s_j \in \mathbf{S}$, $j = 1..S_D$:

$$score^{\texttt{TTM}}(s_j) \propto \#\left[w_{id} \in s_j, x_{id} = 1\right] / nw_j \quad (1)$$

where w_{id} indicates a word in a document d that exists in s_j and is sampled as summary related based on random indicator variable x_{id}. nw_j is the number of words in s_j and normalizes the score favoring sentences with many related words. We rank sentences based on (1). We compare TTM results on synthetic experiments against PAM (Li and McCallum, 2006) a similar topic model that clusters topics in a hierarchical structure, where super-topics are distributions over sub-topics. We obtain sentence scores for PAM models by calculating the sub-topic significance (TS) based on super-topic correlations, and discover topic correlations over the entire document space (corpus wide). Hence; we calculate the TS of a given sub-topic, $k = 1, .., K_2$ by:

$$TS(z_k) = \frac{1}{D} \sum_{d \in D} \frac{1}{K_1} \sum_{k_1}^{K_1} p(z^k_{sub} | z^{k_1}_{sup}) \quad (2)$$

where z^k_{sub} is a sub-topic $k = 1..K_2$ and $z^{k_1}_{sup}$ is a super-topic k_1. The conditional probability of a sub-topic k given a super-topic k_1, $p(z^k_{sub}|z^{k_1}_{sup})$, explains the variation of that sub-topic in relation to other sub-topics. The higher the variation over the entire corpus, the better it represents the general theme of the documents. So, sentences including such topics will have higher saliency scores, which we quantify by imposing topic's significance on vocabulary:

$$score^{\texttt{PAM}}(s_i) = \frac{1}{K_2} \sum_{k}^{K_2} \prod_{w \in s_i} p(w | z^k_{sub}) * TS(z_k) \quad (3)$$

Fig. 4 illustrates the average salience sentence selection performance of TTM and PAM models (for 45 models). The x-axis represents the percentage of sentences selected by the model among all sentences in the DUC2005 corpus. 100% means all sentences in the corpus included in the summary text. The y-axis is the % of selected human sentences over all sentences. The higher the human summary sentences are ranked, the better the model is in selecting the salient sentences. Hence, the system which peaks sooner indicates a better model.

In Fig.4 TTM is significantly better in identifying human sentences as salient in comparison to PAM. The statistical significance is measured based on the area under the curve averaged over 45 models.

5 Enriched Two-Tiered Topic Model

Our model can discover words that are related to summary text using posteriors $\hat{P}(\theta^H)$ and $\hat{P}(\theta^T)$,

[3] An alternative way would be to use Dirichlet priors (Blei et al., 2003) which we opted for due to computational reasons but will be investigated as future research.

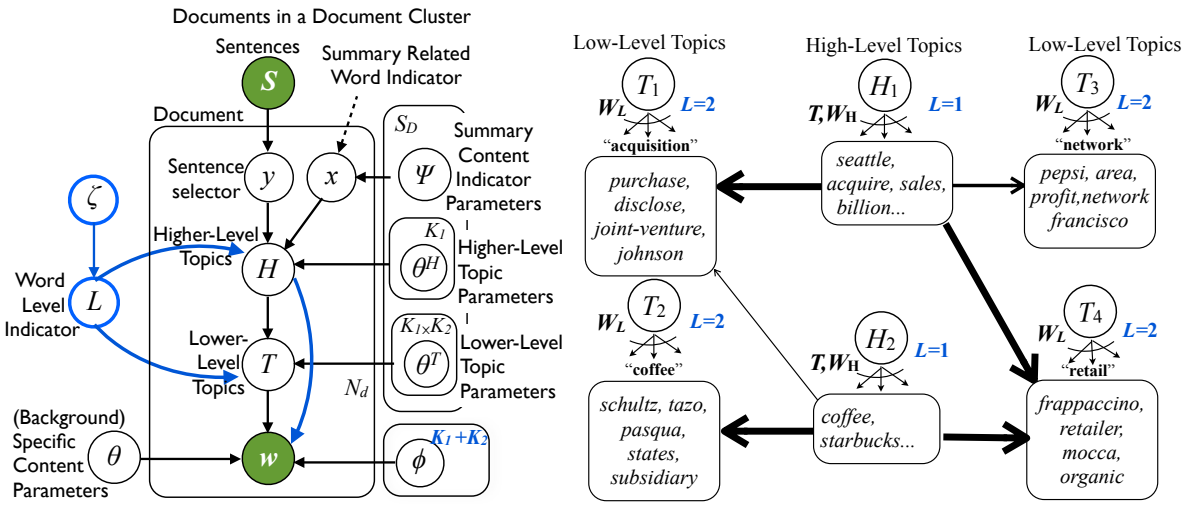


Figure 3: Graphical model depiction of sentence level enriched two-tiered model (ETTM) described in section §5. Each path defined by **H/T** pair $k_1 k_2$, has a multinomial ζ over which level of the path outputs a given word. **L** indicates which level, i.e, high or low, the word is sampled from. On the right is the high-level topic-word and low-level topic-word distributions characterized by ETTM. Each $\mathbf{H_{k_1}}$ also represented as distributions over general words $\mathbf{W_H}$ as well as indicates the degree of correlation between low-level topics denoted by boldness of the arrows.

as well as words w_B specific to documents (via $\hat{P}(\theta)$) (Fig.1). TTM can discover topic correlations, but cannot differentiate if a word in a sentence is more general or specific given a query. Sentences with general words would be more suitable to include in summary text compared to sentences containing specific words. For instance for a given sentence: *"Starbucks Coffee has attempted to expand and diversify through joint ventures, and acquisitions."*, *"starbucks"* and *"coffee"* are more general words given the document clusters compared to *"joint"* and *"ventures"* (see Fig.2), because they appear more frequently in document clusters. However, TTM has no way of knowing that *"starbucks"* and *"coffee"* are common terms given the context. We would like to associate general words with high-level topics, and context specific words with low-level topics. Sentence containing words that are sampled from high-level topics would be a better candidate for summary text. Thus; we present enriched TTM (ETTM) generative process (Fig.3), which samples words not only from low-level topics but also from high-level topics as well.

ETTM discovers three separate distributions over words: (*i*) high-level topics *H* as distributions over corpus general words $\mathbf{W_H}$, (*ii*) low-level topics *T* as distributions over corpus specific words $\mathbf{W_L}$, and

Level Generation for Enriched TTM

Fetch $\zeta_k \sim Beta(\gamma)$; $k = 1...K_1 \times K_2$.
For w_{id}, $i = 1, ..., N_d$, $d = 1, ..D$:
  If $x = 1$, sentence s_i is summary related;
    - sample H_{k_1} and T_{k_2}
    - sample a level L from $Bin(\zeta_{k_1 k_2})$
    - If $L = 1$ (general word); $w_{id} \sim \phi_{H_{k_i}}$
    - else if $L = 2$ (context specific); $w_{id} \sim \phi_{H_{k_1} T_{k_2}}$
  else if $x = 0$, do Step 14-16 in Alg. 1.

(*iii*) background word distributions, i.e,. document specific $\mathbf{W_B}$ (less confidence for summary text). Similar to TTM's generative process, if w_{id} is related to a given query, then $x = 1$ is deterministic, otherwise $x \in \{0, 1\}$ is stochastically determined if w_{id} should be sampled as a background word (w_B) or through hierarchical path, i.e., *H-T* pairs. We first sample a sentence s_i for w_{id} uniformly at random from the sentences containing the word $y_{s_i} \sim Uniform(s_i)$). At this stage we sample a *level* $\mathbf{L}_{w_{id}} \in \{1, 2\}$ for w_{id} to determine if it is a high-level word, e.g., more general to context like *"starbucks"* or *"coffee"* or more specific to related context such as *"subsidiary"*, *"frappucino"*. Each path through the DAG, defined by a H-T pair (total of $K_1 K_2$ pairs), has a binomial $\zeta_{K_1 K_2}$ over which

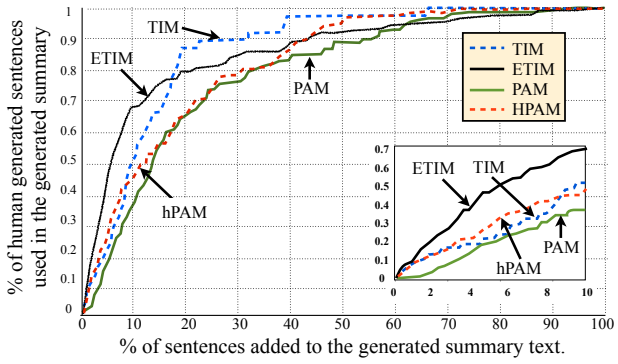


Figure 4: Average saliency performance of four systems over 45 different DUC models. The area under each curve is shown in legend. Inseam is the magnified view of top-ranked 10% of sentences in corpus.

level of the path outputs sampled word. If the word is a specific type, $x = 0$, then it is sampled from the background word distribution θ, a document specific multinomial. Once the level and conditional path is drawn (see level generation for ETTM above) the rest of the generative model is same as TTM.

5.1 Learning and Inference for ETTM

For each word, x is sampled from a sentence specific binomial ψ, just like TTM. If the word is related to the query $x = 1$, we sample a high and low-level topic pair $H - T$ as well as an additional level $\boldsymbol{L}$ is sampled to determine which level of topics the word should be sampled from. L is a corpus specific binomial one for all $H - T$ pairs. If $L = 1$, the word is one of corpus general words and sampled from the high-level topic, otherwise ($L = 2$) the word is corpus specific and sampled from a the low-level topic. The optimum hyper-parameters are set based on training performance via cross validation.

The conditional probabilities are similar to TTM, but with additional random variables, which determine the level of generality of words as follows:

$$p_{\text{ETTM}}(T_{k_1}, T_{k_2}, L|\mathbf{w}, \mathbf{T}_{-k_1}, \mathbf{T}_{-k_2}, L_{)} \propto$$
$$p_{\text{TTM}}(T_{k_1}, T_{k_2}, x = 1|.) * \frac{\gamma + N^L_{k_1 k_2}}{2\gamma + n_{k_1 k_2}}$$

5.2 Summary Generation with ETTM

For ETTM models, we extend the TTM sentence score to be able to include the effect of the general words in sentences (as word sequences in language models) using probabilities of K_1 high-level topic distributions, $\phi^w_{H_{k=1..K_1}}$, as:

$$score^{\text{ETTM}}(s_i) \propto \#\left[w_{id} \in s_j, x_{id} = 1\right] / nw_j * \frac{1}{K_1} \sum_{k=1..K_1} \prod_{w \in s_i} p(w|T_k)$$

where $p(w|T_k)$ is the probability of a word in s_i being generated from high-level topic H^k. Using this score, we re-rank the sentences in documents of the synthetic experiment. We compare the results of ETTM to a structurally similar probabilistic model, entitled hierarchical PAM (Mimno et al., 2007), which is designed to capture topics on a hierarchy of two layers, i.e., super topics and subtopics, where super-topics are distributions over abstract words. In Fig. 4 out of 45 models ETTM has the best performance in ranking the human generated sentences at the top, better than the TTM model. Thus; ETTM is capable of capturing focused sentences with general words related to the main concepts of the documents and much less redundant sentences containing concepts specific to user query.

6 Final Experiments

In this section, we qualitatively compare our models against state-of-the art models and later apply an intrinsic evaluation of generated summaries on topical coherence and informativeness.

For a qualitative comparison with the previous state-of-the models, we use the standard summarization datasets on this task. We train our models on the datasets provided by DUC2005 task and validate the results on DUC 2006 task, which consist of a total of 100 document clusters. We evaluate the performance of our models on DUC2007 datasets, which comprise of 45 document clusters, each containing 25 news articles. The task is to create max. 250 word long summary for each document cluster.

6.1. ROUGE Evaluations: We train each document cluster as a separate corpus to find the optimum parameters of each model and evaluate on test document clusters. ROUGE is a commonly used measure, a standard DUC evaluation metric, which computes recall over various n-grams statistics from a model generated summary against a set of human generated summaries. We report results in R-1 (recall against unigrams), R-2 (recall against bigrams), and R-SU4

ROUGE	w/o stop words			w/ stop words		
	R-1	R-2	R-4	R-1	R-2	R-4
PYTHY	**35.7**	8.9	**12.1**	42.6	**11.9**	**16.8**
HIERSUM	33.8	**9.3**	11.6	42.4	11.8	16.7
HybHSum	35.1	8.3	11.8	**45.6**	11.4	**17.2**
PAM	32.1	7.1	11.0	41.7	9.1	15.3
hPAM	31.9	7.0	11.1	41.2	8.9	15.2
TTM*	**34.0**	8.7	11.5	**44.7**	10.7	16.5
ETTM*	32.4	8.3	11.2	**44.1**	10.4	16.4

Table 1: ROUGE results of the best systems on DUC2007 dataset (best results are **bolded**.) * indicate our models.

(recall against skip-4 bigrams) ROUGE scores w/ and w/o stop words included.

For our models, we ran Gibbs samplers for 2000 iterations for each configuration throwing out first 500 samples as burn-in. We iterated different values for hyperparameters and measured the performance on validation dataset to capture the optimum values.

The following models are used as benchmark: (***i***) PYTHY (Toutanova et al., 2007): Utilizes human generated summaries to train a sentence ranking system using a classifier model; (***ii***) HIERSUM (Haghighi and Vanderwende, 2009): Based on hierarchical topic models. Using an approximation for inference, sentences are greedily added to a summary so long as they decrease KL-divergence of the generated summary concept distributions from document word-frequency distributions. (***iii***) HybHSum (Celikyilmaz and Hakkani-Tur, 2010): A semi-supervised model, which builds a hierarchial LDA to probabilistically score sentences in training dataset as summary or non-summary sentences. Using these probabilities as output variables, it learns a discriminative classifier model to infer the scores of new sentences in testing dataset. (***iv***) PAM (Li and McCallum, 2006) and hPAM (Mimno et al., 2007): Two hierarchical topic models to discover high and low-level concepts from documents, baselines for synthetic experiments in §4 & §5.

Results of our experiments are illustrated in Table 6. Our unsupervised TTM and ETTM systems yield a 44.1 R-1 (w/ stop-words) outperforming the rest of the models, except HybHSum. Because HybHSum uses the human generated summaries as supervision during model development and our systems do not, our performance is quite promising considering the generation is completely unsupervised without seeing any human generated summaries during training. However, the R-2 evaluation (as well as R-4) w/ stop-words does not outperform other models. This is because R-2 is a measure of bi-gram recall and neither of our models represent bi-grams whereas, for instance, PHTHY includes several bi-gram and higher order n-gram statistics. For topic models bi-grams tend to degenerate due to generating inconsistent bag of bi-grams (Wallach, 2006).

6.2. Manual Evaluations: A common DUC task is to manually evaluate models on the quality of generated summaries. We compare our best model ETTM to the results of PAM, our benchmark model in synthetic experiments, as well as hybrid hierarchical summarization model, hLDA (Celikyilmaz and Hakkani-Tur, 2010). Human annotators are given two sets of summary text for each document set, generated from either one of the two approaches: best ETTM and PAM or best ETTM and HybHSum models. The annotators are asked to mark the better summary according to five criteria: *non-redundancy* (which summary is less redundant), *coherence* (which summary is more coherent), *focus and readability* (content and no unnecessary details), *responsiveness* and *overall* performance.

We asked 3 annotators to rate DUC2007 predicted summaries (45 summary pairs per annotator). A total of 42 pairs are judged for ETTM vs. PAM models and 49 pairs for ETTM vs. HybHSum models. The evaluation results in frequencies are shown in Table 6. The participants rated ETTM generated summaries more coherent and focused compared to PAM, where the results are statistically significant (based on t-test on 95% confidence level) indicating that ETTM summaries are rated significantly better. The results of ETTM are slightly better than HybHSum. We consider our results promising because, being unsupervised, ETTM does not utilize human summaries for model development.

7 Conclusion

We introduce two new models for extracting topically coherent sentences from documents, an important property in extractive multi-document summarization systems. Our models combine approaches from the hierarchical topic models. We empha-

	PAM	ETTM	Tie	HybHSum	ETTM	Tie
Non-Redundancy	13	**26**	3	12	**18**	19
Coherence	13	**26**	3	15	**18**	16
Focus	14	**24**	4	14	**17**	18
Responsiveness	15	**24**	3	**19**	12	18
Overall	15	**25**	2	17	**22**	10

Table 2: Frequency results of manual evaluations. Tie indicates evaluations where two summaries are rated equal.

size capturing correlated semantic concepts in documents as well as characterizing general and specific words, in order to identify topically coherent sentences in documents. We showed empirically that a fully unsupervised model for extracting general sentences performs well at summarization task using datasets that were originally used in building automatic summarization system challenges. The success of our model can be traced to its capability of directly capturing coherent topics in documents, which makes it able to identify salient sentences.

Acknowledgments

The authors would like to thank Dr. Zhaleh Feizollahi for her useful comments and suggestions.

References

R. Barzilay and L. Lee. 2004. Catching the drift: Probabilistic content models with applications to generation and summarization. *In Proc. HLT-NAACL'04*.

R. Barzilay, K.R. McKeown, and M. Elhadad. 1999. Information fusion in the context of multi-document summarization. *Proc. 37th ACL*, pages 550–557.

D. Blei, A. Ng, and M. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*.

D. Blei, T. Griffiths, M. Jordan, and J. Tenenbaum. 2004. Hierarchical topic models and the nested chinese restaurant process. *In Neural Information Processing Systems [NIPS]*.

A. Celikyilmaz and D. Hakkani-Tur. 2010. A hybrid hierarchical model for multi-document summarization. *Proc. 48th ACL 2010*.

D. Chen, J. Tang, L. Yao, J. Li, and L. Zhou. 2000. Query-focused summarization by combining topic model and affinity propagation. *LNCS– Advances in Data and Web Development*.

J. Conroy, H. Schlesinger, and D. OLeary. 2006. Topic-focused multi-document summarization using an approximate oracle score. *Proc. ACL*.

H. Daumé-III and D. Marcu. 2006. Bayesian query focused summarization. *Proc. ACL-06*.

J. Eisenstein and R. Barzilay. 2008. Bayesian unsupervised topic segmentation. *Proc. EMNLP-SIGDAT*.

A. Haghighi and L. Vanderwende. 2009. Exploring content models for multi-document summarization. *NAACL HLT-09*.

S. Harabagiu, A. Hickl, and F. Lacatusu. 2007. Satisfying information needs with multi-document summaries. *Information Processing and Management*.

W. Li and A. McCallum. 2006. Pachinko allocation: Dag-structure mixture models of topic correlations. *Proc. ICML*.

W. Li, D. Blei, and A. McCallum. 2007. Nonparametric bayes pachinko allocation. *The 23rd Conference on Uncertainty in Artificial Intelligence*.

C.Y. Lin and E. Hovy. 2002. The automated acquisition of topic signatures fro text summarization. *Proc. CoLing*.

G. A. Miller. 1995. Wordnet: A lexical database for english. *ACM, Vol. 38, No. 11: 39-41*.

D. Mimno, W. Li, and A. McCallum. 2007. Mixtures of hierarchical topics with pachinko allocation. *Proc. ICML*.

A. Nenkova and L. Vanderwende. 2005a. Document summarization using conditional random fields. *Technical report, Microsoft Research*.

A. Nenkova and L. Vanderwende. 2005b. The impact of frequency on summarization. *Technical report, Microsoft Research*.

A. Nenkova, L. Vanderwende, and K. McKowen. 2006. A composition context sensitive multi-document summarizer. *Prof. SIGIR*.

D. R. Radev. 2004. Lexrank: graph-based centrality as salience in text summarization. *Jrnl. Artificial Intelligence Research*.

M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. 2004. The author-topic model for authors and documents. *UAI*.

J. Tang, L. Yao, and D. Chens. 2009. Multi-topic based query-oriented summarization. *SIAM International Conference Data Mining*.

K. Toutanova, C. Brockett, M. Gamon, J. Jagarlamudi, H. Suzuki, and L. Vanderwende. 2007. The phthy summarization system: Microsoft research at duc 2007. In *Proc. DUC*.

H. Wallach. 2006. Topic modeling: Beyond bag-of-words. *Proc. ICML 2006*.

X. Wan and J. Yang. 2006. Improved affinity graph based multi-document summarization. *HLT-NAACL*.

D. Wang, S. Zhu, T. Li, and Y. Gong. 2009. Multi-document summarization using sentence-based topic models. *Proc. ACL 2009*.

Coherent Citation-Based Summarization of Scientific Papers

Amjad Abu-Jbara
EECS Department
University of Michigan
Ann Arbor, MI, USA
amjbara@umich.edu

Dragomir Radev
EECS Department and
School of Information
University of Michigan
Ann Arbor, MI, USA
radev@umich.edu

Abstract

In citation-based summarization, text written by several researchers is leveraged to identify the important aspects of a target paper. Previous work on this problem focused almost exclusively on its extraction aspect (i.e. selecting a representative set of citation sentences that highlight the contribution of the target paper). Meanwhile, the fluency of the produced summaries has been mostly ignored. For example, diversity, readability, cohesion, and ordering of the sentences included in the summary have not been thoroughly considered. This resulted in noisy and confusing summaries. In this work, we present an approach for producing readable and cohesive citation-based summaries. Our experiments show that the proposed approach outperforms several baselines in terms of both extraction quality and fluency.

1 Introduction

Scientific research is a cumulative activity. The work of downstream researchers depends on access to upstream discoveries. The footnotes, end notes, or reference lists within research articles make this accumulation possible. When a reference appears in a scientific paper, it is often accompanied by a span of text describing the work being cited.

We name the sentence that contains an explicit reference to another paper *citation sentence*. Citation sentences usually highlight the most important aspects of the cited paper such as the research problem it addresses, the method it proposes, the good results it reports, and even its drawbacks and limitations.

By aggregating all the citation sentences that cite a paper, we have a rich source of information about it. This information is valuable because human experts have put their efforts to read the paper and summarize its important contributions.

One way to make use of these sentences is creating a summary of the target paper. This summary is different from the abstract or a summary generated from the paper itself. While the abstract represents the author's point of view, the citation summary is the summation of multiple scholars' viewpoints. The task of summarizing a scientific paper using its set of citation sentences is called citation-based summarization.

There has been previous work done on citation-based summarization (Nanba et al., 2000; Elkiss et al., 2008; Qazvinian and Radev, 2008; Mei and Zhai, 2008; Mohammad et al., 2009). Previous work focused on the extraction aspect; i.e. analyzing the collection of citation sentences and selecting a representative subset that covers the main aspects of the paper. The cohesion and the readability of the produced summaries have been mostly ignored. This resulted in noisy and confusing summaries.

In this work, we focus on the coherence and readability aspects of the problem. Our approach produces citation-based summaries in three stages: preprocessing, extraction, and postprocessing. Our experiments show that our approach produces better summaries than several baseline summarization systems.

The rest of this paper is organized as follows. After we examine previous work in Section 2, we outline the motivation of our approach in Section 3. Section 4 describes the three stages of our summarization system. The evaluation and the results are presented in Section 5. Section 6 concludes the paper.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 500–509,
Portland, Oregon, June 19-24, 2011.

2 Related Work

The idea of analyzing and utilizing citation information is far from new. The motivation for using information latent in citations has been explored tens of years back (Garfield et al., 1984; Hodges, 1972). Since then, there has been a large body of research done on citations.

Nanba and Okumura (2000) analyzed citation sentences and automatically categorized citations into three groups using 160 pre-defined phrase-based rules. They also used citation categorization to support a system for writing surveys (Nanba and Okumura, 1999). Newman (2001) analyzed the structure of the citation networks. Teufel et al. (2006) addressed the problem of classifying citations based on their function.

Siddharthan and Teufel (2007) proposed a method for determining the scientific attribution of an article by analyzing citation sentences. Teufel (2007) described a rhetorical classification task, in which sentences are labeled as one of Own, Other, Background, Textual, Aim, Basis, or Contrast according to their role in the authors argument. In parts of our approach, we were inspired by this work.

Elkiss et al. (2008) performed a study on citation summaries and their importance. They concluded that citation summaries are more focused and contain more information than abstracts. Mohammad et al. (2009) suggested using citation information to generate surveys of scientific paradigms.

Qazvinian and Radev (2008) proposed a method for summarizing scientific articles by building a similarity network of the citation sentences that cite the target paper, and then applying network analysis techniques to find a set of sentences that covers as much of the summarized paper facts as possible. We use this method as one of the baselines when we evaluate our approach. Qazvinian et al. (2010) proposed a citation-based summarization method that first extracts a number of important keyphrases from the set of citation sentences, and then finds the best subset of sentences that covers as many keyphrases as possible. Qazvinian and Radev (2010) addressed the problem of identifying the non-explicit citing sentences to aid citation-based summarization.

3 Motivation

The coherence and readability of citation-based summaries are impeded by several factors. First, many citation sentences cite multiple papers besides the target. For example, the following is a citation sentence that appeared in the NLP literature and talked about Resnik's (1999) work.

(1) Grefenstette and Nioche (2000) and Jones and Ghani (2000) use the web to generate corpora for languages where electronic resources are scarce, while ***Resnik (1999)*** *describes a method for mining the web for bilingual texts.*

The first fragment of this sentence describes different work other than Resnik's. The contribution of Resnik is mentioned in the underlined fragment. Including the irrelevant fragments in the summary causes several problems. First, the aim of the summarization task is to summarize the contribution of the target paper using minimal text. These fragments take space in the summary while being irrelevant and less important. Second, including these fragments in the summary breaks the context and, hence, degrades the readability and confuses the reader. Third, the existence of irrelevant fragments in a sentence makes the ranking algorithm assign a low weight to it although the relevant fragment may cover an aspect of the paper that no other sentence covers.

A second factor has to do with the ordering of the sentences included in the summary. For example, the following are two other citation sentences for Resnik (1999).

*(2) Mining the Web for bilingual text (****Resnik, 1999****) is not likely to provide sufficient quantities of high quality data.*

(3) ***Resnik (1999)*** *addressed the issue of language identification for finding Web pages in the languages of interest.*

If these two sentences are to be included in the summary, the reasonable ordering would be to put the second sentence first.

Thirdly, in some instances of citation sentences, the reference is not a syntactic constituent in the sen-

tence. It is added just to indicate the existence of citation. For example, in sentence (2) above, the reference could be safely removed from the sentence without hurting its grammaticality.

In other instances (e.g. sentence (3) above), the reference is a syntactic constituent of the sentence and removing it makes the sentence ungrammatical. However, in certain cases, the reference could be replaced with a suitable pronoun (i.e. he, she or they). This helps avoid the redundancy that results from repeating the author name(s) in every sentence.

Finally, a significant number of citation sentences are not suitable for summarization (Teufel et al., 2006) and should be filtered out. The following sentences are two examples.

(4) The two algorithms we employed in our dependency parsing model are the Eisner parsing ***(Eisner, 1996)*** *and Chu-Lius algorithm (Chu and Liu, 1965).*

(5) This type of model has been used by, among others, ***Eisner (1996)****.*

Sentence (4) appeared in a paper by Nguyen et al (2007). It does not describe any aspect of Eisner's work, rather it informs the reader that Nguyen et al. used Eisner's algorithm in their model. There is no value in adding this sentence to the summary of Eisner's paper. Teufel (2007) reported that a significant number of citation sentences (67% of the sentences in her dataset) were of this type.

Likewise, the comprehension of sentence (5) depends on knowing its context (i.e. its surrounding sentences). This sentence alone does not provide any valuable information about Eisner's paper and should not be added to the summary unless its context is extracted and included in the summary as well.

In our approach, we address these issues to achieve the goal of improving the coherence and the readability of citation-based summaries.

4 Approach

In this section we describe a system that takes a scientific paper and a set of citation sentences that cite it as input, and outputs a citation summary of the paper. Our system produces the summaries in three stages. In the first stage, the citation sentences are preprocessed to rule out the unsuitable sentences and the irrelevant fragments of sentences. In the second stage, a number of citation sentences that cover the various aspects of the paper are selected. In the last stage, the selected sentences are post-processed to enhance the readability of the summary. We describe the stages in the following three subsections.

4.1 Preprocessing

The aim of this stage is to determine which pieces of text (sentences or fragments of sentences) should be considered for selection in the next stage and which ones should be excluded. This stage involves three tasks: reference tagging, reference scope identification, and sentence filtering.

4.1.1 Reference Tagging

A citation sentence contains one or more references. At least one of these references corresponds to the target paper. When writing scientific articles, authors usually use standard patterns to include pointers to their references within the text. We use pattern matching to tag such references. The reference to the target is given a different tag than the references to other papers.

The following example shows a citation sentence with all the references tagged and the target reference given a different tag.

*In <**TREF**>**Resnik (1999)**</**TREF**>, <REF>Nie, Simard, and Foster (2001)</REF>, <REF>Ma and Liberman (1999)</REF>, and <REF>Resnik and Smith (2002)</REF>, the Web is harvested in search of pages that are available in two languages.*

4.1.2 Identifying the Reference Scope

In the previous section, we showed the importance of identifying the scope of the target reference; i.e. the fragment of the citation sentence that corresponds to the target paper. We define the scope of a reference as the shortest fragment of the citation sentence that contains the reference and could form a grammatical sentence if the rest of the sentence was removed.

To find such a fragment, we use a simple yet adequate heuristic. We start by parsing the sentence using the link grammar parser (Sleator and Temperley,

1991). Since the parser is not trained on citation sentences, we replace the references with placeholders before passing the sentence to the parser. Figure 1 shows a portion of the parse tree for Sentence (1) (from Section 1).

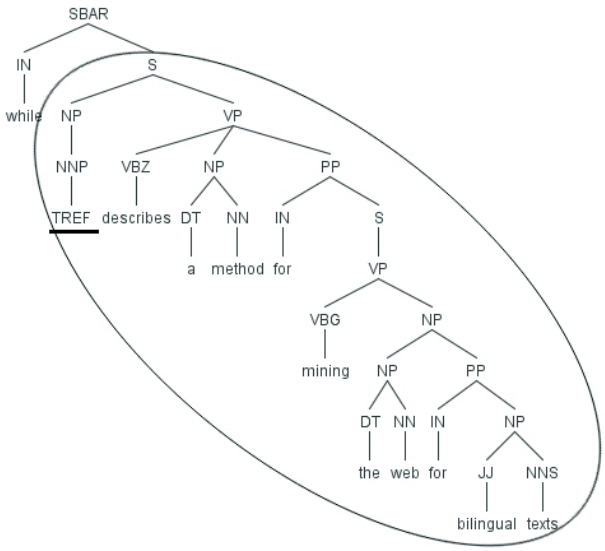


Figure 1: An example showing the scope of a target reference

We extract the scope of the reference from the parse tree as follows. We find the smallest subtree rooted at an S node (sentence clause node) and contains the target reference node. We extract the text that corresponds to this subtree if it is grammatical. Otherwise, we find the second smallest subtree rooted at an S node and so on. For example, the parse tree shown in Figure 1 suggests that the scope of the reference is:

Resnik (1999) *describes a method for mining the web for bilingual texts.*

4.1.3 Sentence Filtering

The task in this step is to detect and filter out unsuitable sentences; i.e., sentences that depend on their context (e.g. Sentence (5) above) or describe the own work of their authors, not the contribution of the target paper (e.g Sentence (4) above). Formally, we classify the citation sentences into two classes: suitable and unsuitable sentences. We use a machine learning technique for this purpose. We extract a number of features from each sentence and train a classification model using these features. The trained model is then used to classify the sentences. We use Support Vector Machines (SVM) with linear kernel as our classifier. The features that we use in this step and their descriptions are shown in Table 1.

4.2 Extraction

In the first stage, the sentences and sentence fragments that are not useful for our summarization task are ruled out. The input to this stage is a set of citation sentences that are believed to be suitable for the summary. From these sentences, we need to select a representative subset. The sentences are selected based on these three main properties:

First, they should cover diverse aspects of the paper. Second, the sentences that cover the same aspect should not contain redundant information. For example, if two sentences talk about the drawbacks of the target paper, one sentence can mention the computation inefficiency, while the other criticize the assumptions the paper makes. Third, the sentences should cover as many important facts about the target paper as possible using minimal text.

In this stage, the summary sentences are selected in three steps. In the first step, the sentences are classified into five functional categories: *Background, Problem Statement, Method, Results*, and *Limitations*. In the second step, we cluster the sentences within each category into clusters of similar sentences. In the third step, we compute the LexRank (Erkan and Radev, 2004) values for the sentences within each cluster. The summary sentences are selected based on the classification, the clustering, and the LexRank values.

4.2.1 Functional Category Classification

We classify the citation sentences into the five categories mentioned above using a machine learning technique. A classification model is trained on a number of features (Table 2) extracted from a labeled set of citation sentences. We use SVM with linear kernel as our classifier.

4.2.2 Sentence Clustering

In the previous step we determined the category of each citation sentence. It is very likely that sentences from the same category contain similar or overlapping information. For example, Sentences (6), (7), and (8) below appear in the set of citation

Feature	Description
Similarity to the target paper	The value of the cosine similarity (using TF-IDF vectors) between the citation sentence and the target paper.
Headlines	The section in which the citation sentence appeared in the citing paper. We recognize 10 section types such as *Introduction, Related Work, Approach, etc*.
Relative position	The relative position of the sentence in the section and the paragraph in which it appears
First person pronouns	This feature takes a value of 1 if the sentence contains a first person pronoun (I, we, our, us, etc.), and 0 otherwise.
Tense of the first verb	A sentence that contains a past tense verb near its beginning is more likely to be describing previous work.
Determiners	Demonstrative Determiners (this, that, these, those, and which) and Alternative Determiners (another, other). The value of this feature is the relative position of the first determiner (if one exists) in the sentence.

Table 1: The features used for sentence filtering

Feature	Description
Similarity to the sections of the target paper	The sections of the target paper are categorized into five categories: 1) *Introduction, Motivation, Problem Statement*. 2) *Background, Prior Work, Previous Work,* and *Related Work*. 3) *Experiments, Results,* and *Evaluation*. 4) *Discussion, Conclusion,* and *Future work*. 5) All other headlines. The value of this feature is the cosine similarity (using TF-IDF vectors) between the sentence and the text of the sections of each of the five section categories.
Headlines	This is the same feature that we used for sentence filtering in Section 4.1.3.
Number of references in the sentence	Sentences that contain multiple references are more likely to be *Background* sentences.
Verbs	We use all the verbs that their lemmatized form appears in at least three sentences that belong to the same category in the training set. Auxiliary verbs are excluded. In our annotated dataset, for example, the verb *propose* appeared in 67 sentences from the *Methodology* category, while the verbs *outperform* and *achieve* appeared in 33 *Result* sentences.

Table 2: The features used for sentence classification

sentences that cite Goldwater and Griffiths' (2007). These sentences belong to the same category (i.e *Method*). Both Sentences (6) and (7) convey the same information about Goldwater and Griffiths (2007) contribution. Sentence (8), however, describes a different aspect of the paper methodology.

(6) ***Goldwater and Griffiths (2007)*** *proposed an information-theoretic measure known as the Variation of Information (VI)*

(7) ***Goldwater and Griffiths (2007)*** *propose using the Variation of Information (VI) metric*

(8) A fully-Bayesian approach to unsupervised POS tagging has been developed by ***Goldwater and Griffiths (2007)*** *as a viable alternative to the traditional maximum likelihood-based HMM approach.*

Clustering divides the sentences of each category into groups of similar sentences. Following Qazvinian and Radev (2008), we build a cosine similarity graph out of the sentences of each category. This is an undirected graph in which nodes are sentences and edges represent similarity relations. Each edge is weighted by the value of the cosine similarity (using TF-IDF vectors) between the two sentences the edge connects. Once we have the similarity network constructed, we partition it into clusters using a community finding technique. We use the Clauset algorithm (Clauset et al., 2004), a hierarchical agglomerative community finding algorithm that runs in linear time.

4.2.3 Ranking

Although the sentences that belong to the same cluster are similar, they are not necessarily equally important. We rank the sentences within each cluster by computing their LexRank (Erkan and Radev, 2004). Sentences with higher rank are more important.

4.2.4 Sentence Selection

At this point we have determined (Figure 2), for each sentence, its category, its cluster, and its relative importance. Sentences are added to the summary in order based on their category, the size of their clusters, then their LexRank values. The categories are

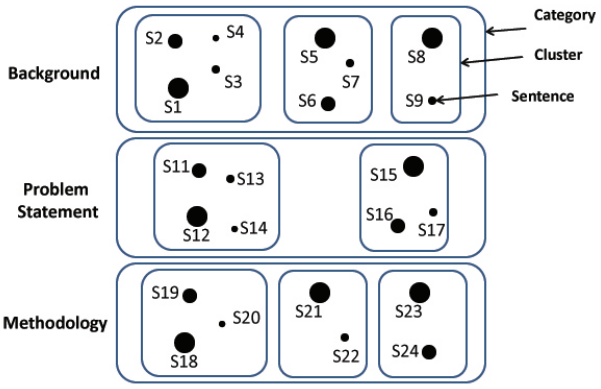


Figure 2: Example illustrating sentence selection

ordered as *Background, Problem, Method, Results*, then *Limitations*. Clusters within each category are ordered by the number of sentences in them whereas the sentences of each cluster are ordered by their LexRank values.

In the example shown in Figure 2, we have three categories. Each category contains several clusters. Each cluster contains several sentences with different LexRank values (illustrated by the sizes of the dots in the figure.) If the desired length of the summary is 3 sentences, the selected sentences will be in order S1, S12, then S18. If the desired length is 5, the selected sentences will be S1, S5, S12, S15, then S18.

4.3 Postprocessing

In this stage, we refine the sentences that we extracted in the previous stage. Each citation sentence will have the target reference (the author's names and the publication year) mentioned at least once. The reference could be either syntactically and semantically part of the sentence (e.g. Sentence (3) above) or not (e.g. Sentence (2)). The aim of this refinement step is to avoid repeating the author's names and the publication year in every sentence. We keep the author's names and the publication year only in the first sentence of the summary. In the following sentences, we either replace the reference with a suitable personal pronoun or remove it. The reference is replaced with a pronoun if it is part of the sentence and this replacement does not make the sentence ungrammatical. The reference is removed if it is not part of the sentence. If the sentence contains references for other papers, they are removed if this doesn't hurt the grammaticality of the sentence.

To determine whether a reference is part of the sentence or not, we again use a machine learning approach. We train a model on a set of labeled sentences. The features used in this step are listed in Table 3. The trained model is then used to classify the references that appear in a sentence into three classes: *keep*, *remove*, *replace*. If a reference is to be replaced, and the paper has one author, we use "*he/she*" (we do not know if the author is male or female). If the paper has two or more authors, we use "*they*".

5 Evaluation

We provide three levels of evaluation. First, we evaluate each of the components in our system separately. Then we evaluate the summaries that our system generate in terms of extraction quality. Finally, we evaluate the coherence and readability of the summaries.

5.1 Data

We use the ACL Anthology Network (AAN) (Radev et al., 2009) in our evaluation. AAN is a collection of more than 16000 papers from the Computational Linguistics journal, and the proceedings of the ACL conferences and workshops. AAN provides all citation information from within the network including the citation network, the citation sentences, and the citation context for each paper.

We used 55 papers from AAN as our data. The papers have a variable number of citation sentences, ranging from 15 to 348. The total number of citation sentences in the dataset is 4,335. We split the data randomly into two different sets; one for evaluating the components of the system, and the other for evaluating the extraction quality and the readability of the generated summaries. The first set (*dataset1*, henceforth) contained 2,284 sentences coming from 25 papers. We asked humans with good background in NLP (the area of the annotated papers) to provide two annotations for each sentence in this set: 1) label the sentence as *Background, Problem, Method, Result, Limitation,* or *Unsuitable*, 2) for each reference in the sentence, determine whether it could be *replaced* with a pronoun, *removed*, or should be *kept*.

Feature	Description
Part-of-speech (POS) tag	We consider the POS tags of the reference, the word before, and the word after. Before passing the sentence to the POS tagger, all the references in the sentence are replaced by placeholders.
Style of the reference	It is common practice in writing scientific papers to put the whole citation between parenthesis when the authors are not a constitutive part of the enclosing sentence, and to enclose just the year between parenthesis when the author's name is a syntactic constituent in the sentence.
Relative position of the reference	This feature takes one of three values: *first*, *last*, and *inside*.
Grammaticality	Grammaticality of the sentence if the reference is removed/replaced. Again, we use the Link Grammar parser (Sleator and Temperley, 1991) to check the grammaticality

Table 3: The features used for author name replacement

Each sentence was given to 3 different annotators. We used the majority vote labels.

We use Kappa coefficient (Krippendorff, 2003) to measure the inter-annotator agreement. Kappa coefficient is defined as:

$$Kappa = \frac{P(A) - P(E)}{1 - P(E)} \quad (1)$$

where $P(A)$ is the relative observed agreement among raters and $P(E)$ is the hypothetical probability of chance agreement.

The agreement among the three annotators on distinguishing the *unsuitable* sentences from the other five categories is 0.85. On Landis and Kochs(1977) scale, this value indicates an *almost perfect* agreement. The agreement on classifying the sentences into the five functional categories is 0.68. On the same scale this value indicates *substantial agreement*.

The second set (*dataset2*, henceforth) contained 30 papers (2051 sentences). We asked humans with a good background in NLP (the papers topic) to generate a readable, coherent summary for each paper in the set using its citation sentences as the source text. We asked them to fix the length of the summaries to 5 sentences. Each paper was assigned to two humans to summarize.

5.2 Component Evaluation

Reference Tagging and Reference Scope Identification Evaluation: We ran our reference tagging and scope identification components on the 2,284 sentences in *dataset1*. Then, we went through the tagged sentences and the extracted scopes, and counted the number of correctly/incorrectly tagged (extracted)/missed references (scopes). Our tagging component achieved 98.2% precision and 94.4% recall. The reference to the target paper was tagged correctly in all the sentences.

-	Bkgrnd	Prob	Method	Results	Limit.
Precision	64.62%	60.01%	88.66%	76.05%	33.53%
Recall	72.47%	59.30%	75.03%	82.29%	59.36%
F1	68.32%	59.65%	81.27%	79.04%	42.85%

Table 4: Precision and recall results achieved by our citation sentence classifier

Our scope identification component extracted the scope of target references with good precision (86.4%) but low recall (35.2%). In fact, extracting a useful scope for a reference requires more than just finding a grammatical substring. In future work, we plan to employ text regeneration techniques to improve the recall by generating grammatical sentences from ungrammatical fragments.

Sentence Filtering Evaluation: We used Support Vector Machines (SVM) with linear kernel as our classifier. We performed 10-fold cross validation on the labeled sentences (*unsuitable* vs *all other categories*) in *dataset1*. Our classifier achieved 80.3% accuracy.

Sentence Classification Evaluation: We used SVM in this step as well. We also performed 10-fold cross validation on the labeled sentences (the five functional categories). This classifier achieved 70.1% accuracy. The precision and recall for each category are given in Table 4

Author Name Replacement Evaluation: The classifier used in this task is also SVM. We performed 10-fold cross validation on the labeled sentences of *dataset1*. Our classifier achieved 77.41% accuracy.

Produced using our system
There has been a large number of studies in tagging and morphological disambiguation using various techniques such as statistical techniques, e.g. constraint-based techniques and transformation-based techniques. A thorough removal of ambiguity requires a syntactic process. A rule-based tagger described in Voutilainen (1995) was equipped with a set of guessing rules that had been hand-crafted using knowledge of English morphology and intuitions. The precision of rule-based taggers may exceed that of the probabilistic ones. The construction of a linguistic rule-based tagger, however, has been considered a difficult and time-consuming task.
Produced using Qazvinian and Radev (2008) system
Another approach is the rule-based or constraint-based approach, recently most prominently exemplified by the Constraint Grammar work (Karlsson et al. , 1995; Voutilainen, 1995b; Voutilainen et al. , 1992; Voutilainen and Tapanainen, 1993), where a large number of hand-crafted linguistic constraints are used to eliminate impossible tags or morphological parses for a given word in a given context. Some systems even perform the POS tagging as part of a syntactic analysis process (Voutilainen, 1995). A rule-based tagger described in (Voutilainen, 1995) is equipped with a set of guessing rules which has been hand-crafted using knowledge of English morphology and intuition. Older versions of EngCG (using about 1,150 constraints) are reported (butilainen et al. 1992; Voutilainen and HeikkiUi 1994; Tapanainen and Voutilainen 1994; Voutilainen 1995) to assign a correct analysis to about 99.7% of all words while each word in the output retains 1.04-1.09 alternative analyses on an average, i.e. some of the ambiguities remait unresolved. We evaluate the resulting disambiguated text by a number of metrics defined as follows (Voutilainen, 1995a).

Table 5: Sample Output

5.3 Extraction Evaluation

To evaluate the extraction quality, we use *dataset2* (that has never been used for training or tuning any of the system components). We use our system to generate summaries for each of the 30 papers in *dataset2*. We also generate summaries for the papers using a number of baseline systems (described in Section 5.3.1). All the generated summaries were 5 sentences long. We use the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) based on the longest common substrings (ROUGE-L) as our evaluation metric.

5.3.1 Baselines

We evaluate the extraction quality of our system (FL) against 7 different baselines. In the first baseline, the sentences are selected randomly from the set of citation sentences and added to the summary. The second baseline is the MEAD summarizer (Radev et al., 2004) with all its settings set to default. The third baseline is LexRank (Erkan and Radev, 2004) run on the entire set of citation sentences of the target paper. The forth baseline is Qazvinian and Radev (2008) citation-based summarizer (QR08) in which the citation sentences are first clustered then the sentences within each cluster are ranked using LexRank. The remaining baselines are variations of our system produced by removing one component from the pipeline at a time. In one variation (FL-1), we remove the *sentence filtering* component. In another variation (FL-2), we remove the *sentence classification* component; so, all the sentences are assumed to come from one category in the subsequent components. In a third variation (FL-3), the clustering component is removed. To make the comparison of the extraction quality to those baselines fair, we remove the *author name replacement* component from our system and all its variations.

5.3.2 Results

Table 6 shows the average ROUGE-L scores (with 95% confidence interval) for the summaries of the 30 papers in *dataset2* generated using our system and the different baselines. The two human summaries were used as models for comparison. The *Human* score reported in the table is the result of comparing the two human summaries to each others. Statistical significance was tested using a 2-tailed paired t-test. The results are statistically significant at the 0.05 level.

The results show that our approach outperforms all the baseline techniques. It achieves higher ROUGE-L score for most of the papers in our testing set. Comparing the score of FL-1 to the score of FL shows that sentence filtering has a significant impact on the results. It also shows that the classification and clustering components both improve the extraction quality.

5.4 Coherence and Readability Evaluation

We asked human judges (not including the authors) to rate the coherence and readability of a number of summaries for each of *dataset2* papers. For each paper we evaluated 3 summaries. The sum-

-	**Human**	**Random**	**MEAD**	**LexRank**	**QR08**
ROUGE-L	0.733	0.398	0.410	0.408	0.435
-	**FL-1**	**FL-2**	**FL-3**	**FL**	-
ROUGE-L	0.475	0.511	0.525	**0.539**	-

Table 6: Extraction Evaluation

Average Coherence Rating	Number of summaries		
	Human	FL	QV08
1≤ coherence <2	0	9	17
2≤ coherence <3	3	11	12
3≤ coherence <4	16	9	1
4≤ coherence ≤5	11	1	0

Table 7: Coherence Evaluation

mary that our system produced, the human summary, and a summary produced by Qazvinian and Radev (2008) summarizer (the best baseline - after our system and its variations - in terms of extraction quality as shown in the previous subsection.) The summaries were randomized and given to the judges without telling them how each summary was produced. The judges were not given access to the source text. They were asked to use a five point-scale to rate how coherent and readable the summaries are, where 1 means that the summary is totally incoherent and needs significant modifications to improve its readability, and 5 means that the summary is coherent and no modifications are needed to improve its readability. We gave each summary to 5 different judges and took the average of their ratings for each summary. We used Weighted Kappa with linear weights (Cohen, 1968) to measure the inter-rater agreement. The Weighted Kappa measure between the five groups of ratings was 0.72.

Table 7 shows the number of summaries in each rating range. The results show that our approach significantly improves the coherence of citation-based summarization. Table 5 shows two sample summaries (each 5 sentences long) for the Voutilainen (1995) paper. One summary was produced using our system and the other was produced using Qazvinian and Radev (2008) system.

6 Conclusions

In this paper, we presented a new approach for citation-based summarization of scientific papers that produces readable summaries. Our approach involves three stages. The first stage preprocesses the set of citation sentences to filter out the irrelevant sentences or fragments of sentences. In the second stage, a representative set of sentences are extracted and added to the summary in a reasonable order. In the last stage, the summary sentences are refined to improve their readability. The results of our experiments confirmed that our system outperforms several baseline systems.

Acknowledgments

This work is in part supported by the National Science Foundation grant "iOPENER: A Flexible Framework to Support Rapid Learning in Unfamiliar Research Domains", jointly awarded to University of Michigan and University of Maryland as IIS 0705832, and in part by the NIH Grant U54 DA021519 to the National Center for Integrative Biomedical Informatics.

Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the supporters.

References

Aaron Clauset, M. E. J. Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Phys. Rev. E*, 70(6):066111, Dec.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70(4):213 – 220.

Aaron Elkiss, Siwei Shen, Anthony Fader, Güneş Erkan, David States, and Dragomir Radev. 2008. Blind men and elephants: What do citation summaries tell us about a research article? *J. Am. Soc. Inf. Sci. Technol.*, 59(1):51–62.

Gunes Erkan and Dragomir R. Radev. 2004. Lexrank: graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479.

E. Garfield, Irving H. Sher, and R. J. Torpie. 1984. *The Use of Citation Data in Writing the History of Science*. Institute for Scientific Information Inc., Philadelphia, Pennsylvania, USA.

T. L. Hodges. 1972. Citation indexing-its theory and application in science, technology, and humanities. *Ph.D. thesis, University of California at Berkeley.Ph.D. thesis, University of California at Berkeley.*

Klaus H. Krippendorff. 2003. *Content Analysis: An Introduction to Its Methodology*. Sage Publications, Inc, 2nd edition, December.

J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1):159–174, March.

Qiaozhu Mei and ChengXiang Zhai. 2008. Generating impact-based summaries for scientific literature. In *Proceedings of ACL-08: HLT*, pages 816–824, Columbus, Ohio, June. Association for Computational Linguistics.

Saif Mohammad, Bonnie Dorr, Melissa Egan, Ahmed Hassan, Pradeep Muthukrishan, Vahed Qazvinian, Dragomir Radev, and David Zajic. 2009. Using citations to generate surveys of scientific paradigms. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 584–592, Boulder, Colorado, June. Association for Computational Linguistics.

Hidetsugu Nanba and Manabu Okumura. 1999. Towards multi-paper summarization using reference information. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 926–931, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Hidetsugu Nanba, Noriko Kando, Manabu Okumura, and Of Information Science. 2000. Classification of research papers using citation links and citation types: Towards automatic review article generation.

M. E. J. Newman. 2001. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences of the United States of America*, 98(2):404–409, January.

Vahed Qazvinian and Dragomir R. Radev. 2008. Scientific paper summarization using citation summary networks. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 689–696, Manchester, UK, August.

Vahed Qazvinian and Dragomir R. Radev. 2010. Identifying non-explicit citing sentences for citation-based summarization. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 555–564, Uppsala, Sweden, July. Association for Computational Linguistics.

Vahed Qazvinian, Dragomir R. Radev, and Arzucan Ozgur. 2010. Citation summarization through keyphrase extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 895–903, Beijing, China, August. Coling 2010 Organizing Committee.

Dragomir Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Çelebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Michael Topper, Adam Winkel, and Zhu Zhang. 2004. MEAD - a platform for multidocument multilingual text summarization. In *LREC 2004*, Lisbon, Portugal, May.

Dragomir R. Radev, Pradeep Muthukrishnan, and Vahed Qazvinian. 2009. The acl anthology network corpus. In *NLPIR4DL '09: Proceedings of the 2009 Workshop on Text and Citation Analysis for Scholarly Digital Libraries*, pages 54–61, Morristown, NJ, USA. Association for Computational Linguistics.

Advaith Siddharthan and Simone Teufel. 2007. Whose idea was this, and why does it matter? attributing scientific work to citations. In *In Proceedings of NAACL/HLT-07*.

Daniel D. K. Sleator and Davy Temperley. 1991. Parsing english with a link grammar. In *In Third International Workshop on Parsing Technologies*.

Simone Teufel, Advaith Siddharthan, and Dan Tidhar. 2006. Automatic classification of citation function. In *In Proc. of EMNLP-06*.

Simone Teufel. 2007. Argumentative zoning for improved citation indexing. computing attitude and affect in text. In *Theory and Applications, pages 159170*.

A Class of Submodular Functions for Document Summarization

Hui Lin
Dept. of Electrical Engineering
University of Washington
Seattle, WA 98195, USA
hlin@ee.washington.edu

Jeff Bilmes
Dept. of Electrical Engineering
University of Washington
Seattle, WA 98195, USA
bilmes@ee.washington.edu

Abstract

We design a class of submodular functions meant for document summarization tasks. These functions each combine two terms, one which encourages the summary to be representative of the corpus, and the other which positively rewards diversity. Critically, our functions are monotone nondecreasing and submodular, which means that an efficient scalable greedy optimization scheme has a constant factor guarantee of optimality. When evaluated on DUC 2004-2007 corpora, we obtain better than existing state-of-art results in both generic and query-focused document summarization. Lastly, we show that several well-established methods for document summarization correspond, in fact, to submodular function optimization, adding further evidence that submodular functions are a natural fit for document summarization.

1 Introduction

In this paper, we address the problem of generic and query-based extractive summarization from collections of related documents, a task commonly known as *multi-document summarization*. We treat this task as monotone submodular function maximization (to be defined in Section 2). This has a number of critical benefits. On the one hand, there exists a simple greedy algorithm for monotone submodular function maximization where the summary solution obtained (say $\hat{S}$) is guaranteed to be almost as good as the best possible solution (say S_{opt}) according to an objective $\mathcal{F}$. More precisely, the greedy algorithm is a constant factor approximation to the cardinality constrained version of the problem, so that $\mathcal{F}(\hat{S}) \geq (1 - 1/e)\mathcal{F}(S_{\text{opt}}) \approx 0.632\mathcal{F}(S_{\text{opt}})$. This is particularly attractive since the quality of the solution does not depend on the size of the problem, so even very large size problems do well. It is also important to note that this is a worst case bound, and in most cases the quality of the solution obtained will be much better than this bound suggests.

Of course, none of this is useful if the objective function $\mathcal{F}$ is inappropriate for the summarization task. In this paper, we argue that monotone nondecreasing submodular functions $\mathcal{F}$ are an ideal class of functions to investigate for document summarization. We show, in fact, that many well-established methods for summarization (Carbonell and Goldstein, 1998; Filatova and Hatzivassiloglou, 2004; Takamura and Okumura, 2009; Riedhammer et al., 2010; Shen and Li, 2010) correspond to submodular function optimization, a property not explicitly mentioned in these publications. We take this fact, however, as testament to the value of submodular functions for summarization: if summarization algorithms are repeatedly developed that, by chance, happen to be an instance of a submodular function optimization, this suggests that submodular functions are a natural fit. On the other hand, other authors have started realizing explicitly the value of submodular functions for summarization (Lin and Bilmes, 2010; Qazvinian et al., 2010).

Submodular functions share many properties in common with convex functions, one of which is that they are closed under a number of common combination operations (summation, certain compositions, restrictions, and so on). These operations give us the tools necessary to design a powerful submodular objective for submodular document summarization that extends beyond any previous work. We demonstrate this by carefully crafting a class of submodular func-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 510–520,
Portland, Oregon, June 19-24, 2011.

tions we feel are ideal for extractive summarization tasks, both generic and query-focused. In doing so, we demonstrate better than existing state-of-the-art performance on a number of standard summarization evaluation tasks, namely DUC-04 through to DUC-07. We believe our work, moreover, might act as a springboard for researchers in summarization to consider the problem of "how to design a submodular function" for the summarization task.

In Section 2, we provide a brief background on submodular functions and their optimization. Section 3 describes how the task of extractive summarization can be viewed as a problem of submodular function maximization. We also in this section show that many standard methods for summarization are, in fact, already performing submodular function optimization. In Section 4, we present our own submodular functions. Section 5 presents results on both generic and query-focused summarization tasks, showing as far as we know the best known ROUGE results for DUC-04 through DUC-06, and the best known precision results for DUC-07, and the best recall DUC-07 results among those that do not use a web search engine. Section 6 discusses implications for future work.

2 Background on Submodularity

We are given a set of objects $V = \{v_1, \ldots, v_n\}$ and a function $\mathcal{F} : 2^V \rightarrow \mathbb{R}$ that returns a real value for any subset $S \subseteq V$. We are interested in finding the subset of bounded size $|S| \leq k$ that maximizes the function, e.g., $\operatorname{argmax}_{S \subseteq V} \mathcal{F}(S)$. In general, this operation is hopelessly intractable, an unfortunate fact since the optimization coincides with many important applications. For example, $\mathcal{F}$ might correspond to the value or coverage of a set of sensor locations in an environment, and the goal is to find the best locations for a fixed number of sensors (Krause et al., 2008).

If the function $\mathcal{F}$ is monotone submodular then the maximization is still NP complete, but it was shown in (Nemhauser et al., 1978) that a greedy algorithm finds an approximate solution guaranteed to be within $\frac{e-1}{e} \sim 0.63$ of the optimal solution, as mentioned in Section 1. A version of this algorithm (Minoux, 1978), moreover, scales to very large data sets. Submodular functions are those that satisfy the property of *diminishing returns*: for any $A \subseteq B \subseteq V \backslash v$, a *submodular* function $\mathcal{F}$ must satisfy $\mathcal{F}(A+v) - \mathcal{F}(A) \geq \mathcal{F}(B+v) - \mathcal{F}(B)$. That is, the incremental "value" of v decreases as the context in which v is considered grows from A to B. An equivalent definition, useful mathematically, is that for any $A, B \subseteq V$, we must have that $\mathcal{F}(A) + \mathcal{F}(B) \geq \mathcal{F}(A \cup B) + \mathcal{F}(A \cap B)$. If this is satisfied everywhere with equality, then the function $\mathcal{F}$ is called *modular*, and in such case $\mathcal{F}(A) = c + \sum_{a \in A} \vec{f}_a$ for a sized $|V|$ vector $\vec{f}$ of real values and constant c. A set function $\mathcal{F}$ is *monotone nondecreasing* if $\forall A \subseteq B$, $\mathcal{F}(A) \leq \mathcal{F}(B)$. As shorthand, in this paper, monotone nondecreasing submodular functions will simply be referred to as *monotone submodular*.

Historically, submodular functions have their roots in economics, game theory, combinatorial optimization, and operations research. More recently, submodular functions have started receiving attention in the machine learning and computer vision community (Kempe et al., 2003; Narasimhan and Bilmes, 2005; Krause and Guestrin, 2005; Narasimhan and Bilmes, 2007; Krause et al., 2008; Kolmogorov and Zabin, 2004) and have recently been introduced to natural language processing for the tasks of document summarization (Lin and Bilmes, 2010) and word alignment (Lin and Bilmes, 2011).

Submodular functions share a number of properties in common with convex and concave functions (Lovász, 1983), including their wide applicability, their generality, their multiple options for their representation, and their closure under a number of common operators (including mixtures, truncation, complementation, and certain convolutions). For example, if a collection of functions $\{\mathcal{F}_i\}_i$ is submodular, then so is their weighted sum $\mathcal{F} = \sum_i \alpha_i \mathcal{F}_i$ where α_i are nonnegative weights. It is not hard to show that submodular functions also have the following composition property with concave functions:

Theorem 1. *Given functions $\mathcal{F} : 2^V \rightarrow \mathbb{R}$ and $f : \mathbb{R} \rightarrow \mathbb{R}$, the composition $\mathcal{F}' = f \circ \mathcal{F} : 2^V \rightarrow \mathbb{R}$ (i.e., $\mathcal{F}'(S) = f(\mathcal{F}(S))$) is nondecreasing submodular, if f is non-decreasing concave and $\mathcal{F}$ is nondecreasing submodular.*

This property will be quite useful when defining submodular functions for document summarization.

3 Submodularity in Summarization

3.1 Summarization with knapsack constraint

Let the *ground set* V represents all the sentences (or other linguistic units) in a document (or document collection, in the multi-document summarization case). The task of extractive document summarization is to select a subset $S \subseteq V$ to represent the entirety (ground set V). There are typically constraints on S, however. Obviously, we should have $|S| < |V| = N$ as it is a summary and should be small. In standard summarization tasks (e.g., DUC evaluations), the summary is usually required to be length-limited. Therefore, constraints on S can naturally be modeled as *knapsack constraints*: $\sum_{i \in S} c_i \leq b$, where c_i is the non-negative cost of selecting unit i (e.g., the number of words in the sentence) and b is our *budget*. If we use a set function $\mathcal{F} : 2^V \rightarrow \mathbb{R}$ to measure the quality of the summary set S, the summarization problem can then be formalized as the following combinatorial optimization problem:

Problem 1. *Find*

$$S^* \in \operatorname*{argmax}_{S \subseteq V} \mathcal{F}(S) \textit{ subject to: } \sum_{i \in S} c_i \leq b.$$

Since this is a generalization of the cardinality constraint (where $c_i = 1, \forall i$), this also constitutes a (well-known) NP-hard problem. In this case as well, however, a modified greedy algorithm with partial enumeration can solve Problem 1 near-optimally with $(1-1/e)$-approximation factor if $\mathcal{F}$ is monotone submodular (Sviridenko, 2004). The partial enumeration, however, is too computationally expensive for real world applications. In (Lin and Bilmes, 2010), we generalize the work by Khuller et al. (1999) on the budgeted maximum cover problem to the general submodular framework, and show a practical greedy algorithm with a $(1 - 1/\sqrt{e})$-approximation factor, where each greedy step adds the unit with the largest ratio of objective function gain to scaled cost, while not violating the budget constraint (see (Lin and Bilmes, 2010) for details). Note that in all cases, submodularity and monotonicity are two necessary ingredients to guarantee that the greedy algorithm gives near-optimal solutions.

In fact, greedy-like algorithms have been widely used in summarization. One of the more popular approaches is *maximum marginal relevance* (MMR) (Carbonell and Goldstein, 1998), where a greedy algorithm selects the most relevant sentences, and at the same time avoids redundancy by removing sentences that are too similar to ones already selected. Interestingly, the gain function defined in the original MMR paper (Carbonell and Goldstein, 1998) satisfies diminishing returns, a fact apparently unnoticed until now. In particular, Carbonell and Goldstein (1998) define an objective function gain of adding element k to set S ($k \notin S$) as:

$$\lambda Sim_1(s_k, q) - (1 - \lambda) \max_{i \in S} Sim_2(s_i, s_k), \quad (1)$$

where $Sim_1(s_k, q)$ measures the similarity between unit s_k to a query q, $Sim_2(s_i, s_k)$ measures the similarity between unit s_i and unit s_k, and $0 \leq \lambda \leq 1$ is a trade-off coefficient. We have:

Theorem 2. *Given an expression for $\mathcal{F}_{MMR}$ such that $\mathcal{F}_{MMR}(S \cup \{k\}) - \mathcal{F}_{MMR}(S)$ is equal to Eq. 1, $\mathcal{F}_{MMR}$ is non-monotone submodular.*

Obviously, diminishing-returns hold since

$$\max_{i \in S} Sim_2(s_i, s_k) \leq \max_{i \in R} Sim_2(s_i, s_k)$$

for all $S \subseteq R$, and therefore $\mathcal{F}_{\text{MMR}}$ is submodular. On the other hand, $\mathcal{F}_{\text{MMR}}$, would not be monotone, so the greedy algorithm's constant-factor approximation guarantee does not apply in this case.

When scoring a summary at the sub-sentence level, submodularity naturally arises. Concept-based summarization (Filatova and Hatzivassiloglou, 2004; Takamura and Okumura, 2009; Riedhammer et al., 2010; Qazvinian et al., 2010) usually maximizes the weighted credit of concepts covered by the summary. Although the authors may not have noticed, their objective functions are also submodular, adding more evidence suggesting that submodularity is natural for summarization tasks. Indeed, let S be a subset of sentences in the document and denote $\Gamma(S)$ as the set of concepts contained in S. The total credit of the concepts covered by S is then

$$\mathcal{F}_{\text{concept}}(S) \triangleq \sum_{i \in \Gamma(S)} c_i,$$

where c_i is the credit of concept i. This function is known to be submodular (Narayanan, 1997).

Similar to the MMR approach, in (Lin and Bilmes, 2010), a submodular graph based objective function is proposed where a graph cut function, measuring the similarity of the summary to the rest of document, is combined with a subtracted redundancy penalty function. The objective function is submodular but again, non-monotone. We theoretically justify that the performance guarantee of the greedy algorithm holds for this objective function with high probability (Lin and Bilmes, 2010). Our justification, however, is shown to be applicable only to certain particular non-monotone submodular functions, under certain reasonable assumptions about the probability distribution over weights of the graph.

3.2 Summarization with covering constraint

Another perspective is to treat the summarization problem as finding a low-cost subset of the document under the constraint that a summary should cover all (or a sufficient amount of) the information in the document. Formally, this can be expressed as

Problem 2. *Find*

$$S^* \in \operatorname*{argmin}_{S \subseteq V} \sum_{i \in S} c_i \textit{ subject to: } \mathcal{F}(S) \geq \alpha,$$

where c_i are the element costs, and set function $\mathcal{F}(S)$ measure the information covered by S. When $\mathcal{F}$ is submodular, the constraint $\mathcal{F}(S) \geq \alpha$ is called a *submodular cover* constraint. When $\mathcal{F}$ is monotone submodular, a greedy algorithm that iteratively selects k with minimum $c_k/(\mathcal{F}(S \cup \{k\}) - \mathcal{F}(S))$ has approximation guarantees (Wolsey, 1982). Recent work (Shen and Li, 2010) proposes to model document summarization as finding a minimum dominating set and a greedy algorithm is used to solve the problem. The dominating set constraint is also a submodular cover constraint. Define $\delta(S)$ be the set of elements that is either in S or is adjacent to some element in S. Then S is a dominating set if $|\delta(S)| = |V|$. Note that

$$\mathcal{F}_{\text{dom}}(S) \triangleq |\delta(S)|$$

is monotone submodular. The dominating set constraint is then also a submodular cover constraint, and therefore the approaches in (Shen and Li, 2010) are special cases of Problem 2. The solutions found in this framework, however, do not necessarily satisfy a summary's budget constraint. Consequently, a subset of the solution found by solving Problem 2 has to be constructed as the final summary, and the near-optimality is no longer guaranteed. Therefore, solving Problem 1 for document summarization appears to be a better framework regarding global optimality. In the present paper, our framework is that of Problem 1.

3.3 Automatic summarization evaluation

Automatic evaluation of summary quality is important for the research of document summarization as it avoids the labor-intensive and potentially inconsistent human evaluation. ROUGE (Lin, 2004) is widely used for summarization evaluation and it has been shown that ROUGE-N scores are highly correlated with human evaluation (Lin, 2004). Interestingly, ROUGE-N is monotone submodular, adding further evidence that monotone submodular functions are natural for document summarization.

Theorem 3. *ROUGE-N is monotone submodular.*

Proof. By definition (Lin, 2004), ROUGE-N is the n-gram recall between a candidate summary and a set of reference summaries. Precisely, let S be the candidate summary (a set of sentences extracted from the ground set V), $c_e : 2^V \rightarrow \mathbb{Z}_+$ be the number of times n-gram e occurs in summary S, and R_i be the set of n-grams contained in the reference summary i (suppose we have K reference summaries, i.e., $i = 1, \cdots, K$). Then ROUGE-N can be written as the following set function:

$$\mathcal{F}_{\text{ROUGE-N}}(S) \triangleq \frac{\sum_{i=1}^{K} \sum_{e \in R_i} \min(c_e(S), r_{e,i})}{\sum_{i=1}^{K} \sum_{e \in R_i} r_{e,i}},$$

where $r_{e,i}$ is the number of times n-gram e occurs in reference summary i. Since $c_e(S)$ is monotone modular and $\min(x, a)$ is a concave non-decreasing function of x, $\min(c_e(S), r_{e,i})$ is monotone submodular by Theorem 1. Since summation preserves submodularity, and the denominator is constant, we see that $\mathcal{F}_{\text{ROUGE-N}}$ is monotone submodular. □

Since the reference summaries are unknown, it is of course impossible to optimize $\mathcal{F}_{\text{ROUGE-N}}$ directly. Therefore, some approaches (Filatova and Hatzivassiloglou, 2004; Takamura and Okumura, 2009; Riedhammer et al., 2010) instead define "concepts". Alter-

natively, we herein propose a class of monotone submodular functions that naturally models the quality of a summary while not depending on an explicit notion of concepts, as we will see in the following section.

4 Monotone Submodular Objectives

Two properties of a good summary are *relevance* and *non-redundancy*. Objective functions for extractive summarization usually measure these two separately and then mix them together trading off encouraging relevance and penalizing redundancy. The redundancy penalty usually violates the monotonicity of the objective functions (Carbonell and Goldstein, 1998; Lin and Bilmes, 2010). We therefore propose to positively *reward diversity* instead of negatively penalizing redundancy. In particular, we model the summary quality as

$$\mathcal{F}(S) = \mathcal{L}(S) + \lambda \mathcal{R}(S), \tag{2}$$

where $\mathcal{L}(S)$ measures the coverage, or "fidelity", of summary set S to the document, $\mathcal{R}(S)$ rewards diversity in S, and $\lambda \geq 0$ is a trade-off coefficient. Note that the above is analogous to the objectives widely used in machine learning, where a loss function that measures the training set error (we measure the coverage of summary to a document), is combined with a regularization term encouraging certain desirable (e.g., sparsity) properties (in our case, we "regularize" the solution to be more diverse). In the following, we discuss how both $\mathcal{L}(S)$ and $\mathcal{R}(S)$ are naturally monotone submodular.

4.1 Coverage function

$\mathcal{L}(S)$ can be interpreted either as a set function that measures the similarity of summary set S to the document to be summarized, or as a function representing some form of "coverage" of V by S. Most naturally, $\mathcal{L}(S)$ should be monotone, as coverage improves with a larger summary. $\mathcal{L}(S)$ should also be submodular: consider adding a new sentence into two summary sets, one a subset of the other. Intuitively, the increment when adding a new sentence to the small summary set should be larger than the increment when adding it to the larger set, as the information carried by the new sentence might have already been covered by those sentences that are in the larger summary but not in the smaller summary. This is exactly the property of diminishing returns. Indeed, Shannon entropy, as the measurement of information, is another well-known monotone submodular function.

There are several ways to define $\mathcal{L}(S)$ in our context. For instance, we could use $\mathcal{L}(S) = \sum_{i \in V, j \in S} w_{i,j}$ where $w_{i,j}$ represents the similarity between i and j. $\mathcal{L}(S)$ could also be facility location objective, i.e., $\mathcal{L}(S) = \sum_{i \in V} \max_{j \in S} w_{i,j}$, as used in (Lin et al., 2009). We could also use $\mathcal{L}(S) = \sum_{i \in \Gamma(S)} c_i$ as used in concept-based summarization, where the definition of "concept" and the mechanism to extract these concepts become important. All of these are monotone submodular.

Alternatively, in this paper we propose the following objective that does not reply on concepts. Let

$$\mathcal{L}(S) = \sum_{i \in V} \min \left\{ \mathcal{C}_i(S), \alpha\, \mathcal{C}_i(V) \right\}, \tag{3}$$

where $\mathcal{C}_i : 2^V \rightarrow \mathbb{R}$ is a monotone submodular function and $0 \leq \alpha \leq 1$ is a threshold co-efficient. Firstly, $\mathcal{L}(S)$ as defined in Eqn. 3 is a monotone submodular function. The monotonicity is immediate. To see that $\mathcal{L}(S)$ is submodular, consider the fact that $f(x) = \min(x, a)$ where $a \geq 0$ is a concave non-decreasing function, and by Theorem 1, each summand in Eqn. 3 is a submodular function, and as summation preserves submodularity, $\mathcal{L}(S)$ is submodular.

Next, we explain the intuition behind Eqn. 3. Basically, $\mathcal{C}_i(S)$ measures how similar S is to element i, or how much of i is "covered" by S. Then $\mathcal{C}_i(V)$ is just the largest value that $\mathcal{C}_i(S)$ can achieve. We call i "saturated" by S when $\min\{\mathcal{C}_i(S), \alpha\mathcal{C}_i(V)\} = \alpha\mathcal{C}_i(V)$. When i is already saturated in this way, any new sentence j can not further improve the coverage of i even if it is very similar to i (i.e., $\mathcal{C}_i(S \cup \{j\}) - \mathcal{C}_i(S)$ is large). This will give other sentences that are not yet saturated a higher chance of being better covered, and therefore the resulting summary tends to better cover the entire document.

One simple way to define $\mathcal{C}_i(S)$ is just to use

$$\mathcal{C}_i(S) = \sum_{j \in S} w_{i,j} \tag{4}$$

where $w_{i,j} \geq 0$ measures the similarity between i and j. In this case, when $\alpha = 1$, Eqn. 3 reduces to the case where $\mathcal{L}(S) = \sum_{i \in V, j \in S} w_{i,j}$. As we will see in Section 5, having an α that is less than

1 significantly improves the performance compared to the case when $\alpha = 1$, which coincides with our intuition that using a truncation threshold improves the final summary's coverage.

4.2 Diversity reward function

Instead of penalizing redundancy by subtracting from the objective, we propose to reward diversity by adding the following to the objective:

$$\mathcal{R}(S) = \sum_{i=1}^{K} \sqrt{\sum_{j \in P_i \cap S} r_j}. \quad (5)$$

where $P_i, i = 1, \cdots K$ is a partition of the ground set V (i.e., $\bigcup_i P_i = V$ and the P_is are disjoint) into separate clusters, and $r_i \geq 0$ indicates the *singleton reward* of i (i.e., the reward of adding i into the empty set). The value r_i estimates the importance of i to the summary. The function $\mathcal{R}(S)$ rewards diversity in that there is usually more benefit to selecting a sentence from a cluster not yet having one of its elements already chosen. As soon as an element is selected from a cluster, other elements from the same cluster start having diminishing gain, thanks to the square root function. For instance, consider the case where $k_1, k_2 \in P_1$, $k_3 \in P_2$, and $r_{k_1} = 4$, $r_{k_2} = 9$, and $r_{k_3} = 4$. Assume k_1 is already in the summary set S. Greedily selecting the next element will choose k_3 rather than k_2 since $\sqrt{13} < 2 + 2$. In other words, adding k_3 achieves a greater reward as it increases the diversity of the summary (by choosing from a different cluster). Note, $\mathcal{R}(S)$ is distinct from $\mathcal{L}(S)$ in that $\mathcal{R}(S)$ might wish to include certain outlier material that $\mathcal{L}(S)$ could ignore.

It is easy to show that $\mathcal{R}(S)$ is submodular by using the composition rule from Theorem 1. The square root is non-decreasing concave function. Inside each square root lies a modular function with non-negative weights (and thus is monotone). Applying the square root to such a monotone submodular function yields a submodular function, and summing them all together retains submodularity, as mentioned in Section 2. The monotonicity of $\mathcal{R}(S)$ is straightforward. Note, the form of Eqn. 5 is similar to structured group norms (e.g., (Zhao et al., 2009)), recently shown to be related to submodularity (Bach, 2010; Jegelka and Bilmes, 2011).

Several extensions to Eqn. 5 are discussed next: First, instead of using a ground set partition, intersecting clusters can be used. Second, the square root function in Eqn. 5 can be replaced with any other non-decreasing concave functions (e.g., $f(x) = log(1 + x)$) while preserving the desired property of $\mathcal{R}(S)$, and the curvature of the concave function then determines the rate that the reward diminishes. Last, multi-resolution clustering (or partitions) with different sizes (K) can be used, i.e., we can use a mixture of components, each of which has the structure of Eqn. 5. A mixture can better represent the core structure of the ground set (e.g., the hierarchical structure in the documents (Celikyilmaz and Hakkani-tür, 2010)). All such extensions preserve both monotonicity and submodularity.

5 Experiments

The document understanding conference (DUC) (`http://duc.nist.org`) was the main forum providing benchmarks for researchers working on document summarization. The tasks in DUC evolved from single-document summarization to multi-document summarization, and from generic summarization (2001–2004) to query-focused summarization (2005–2007). As ROUGE (Lin, 2004) has been officially adopted for DUC evaluations since 2004, we also take it as our main evaluation criterion. We evaluated our approaches on DUC data 2003-2007, and demonstrate results on both generic and query-focused summarization. In all experiments, the modified greedy algorithm (Lin and Bilmes, 2010) was used for summary generation.

5.1 Generic summarization

Summarization tasks in DUC-03 and DUC-04 are multi-document summarization on English news articles. In each task, 50 document clusters are given, each of which consists of 10 documents. For each document cluster, the system generated summary may not be longer than 665 bytes including spaces and punctuation. We used DUC-03 as our development set, and tested on DUC-04 data. We show ROUGE-1 scores[1] as it was the main evaluation criterion for DUC-03, 04 evaluations.

[1]ROUGE version 1.5.5 with options: -a -c 95 -b 665 -m -n 4 -w 1.2

Documents were pre-processed by segmenting sentences and stemming words using the Porter Stemmer. Each sentence was represented using a bag-of-terms vector, where we used context terms up to bi-grams. Similarity between sentence i and sentence j, i.e., $w_{i,j}$, was computed using cosine similarity:

$$w_{i,j} = \frac{\sum_{w \in s_i} \text{tf}_{w,i} \times \text{tf}_{w,j} \times \text{idf}_w^2}{\sqrt{\sum_{w \in s_i} \text{tf}_{w,s_i}^2 \text{idf}_w^2} \sqrt{\sum_{w \in s_j} \text{tf}_{w,j}^2 \text{idf}_w^2}},$$

where $\text{tf}_{w,i}$ and $\text{tf}_{w,j}$ are the numbers of times that w appears in s_i and sentence s_j respectively, and idf_w is the inverse document frequency (IDF) of term w (up to bigram), which was calculated as the logarithm of the ratio of the number of articles that w appears over the total number of all articles in the document cluster.

Table 1: ROUGE-1 recall (R) and F-measure (F) results (%) on DUC-04. DUC-03 was used as development set.

DUC-04	R	F
$\sum_{i \in V} \sum_{j \in S} w_{i,j}$	33.59	32.44
$\mathcal{L}_1(S)$	39.03	38.65
$\mathcal{R}_1(S)$	38.23	37.81
$\mathcal{L}_1(S) + \lambda \mathcal{R}_1(S)$	**39.35**	**38.90**
Takamura and Okumura (2009)	38.50	-
Wang et al. (2009)	39.07	-
Lin and Bilmes (2010)	-	38.39
Best system in DUC-04 (peer 65)	38.28	37.94

We first tested our coverage and diversity reward objectives separately. For coverage, we use a modular $\mathcal{C}_i(S) = \sum_{j \in S} w_{i,j}$ for each sentence i, i.e.,

$$\mathcal{L}_1(S) = \sum_{i \in V} \min \left\{ \sum_{j \in S} w_{i,j}, \alpha \sum_{k \in V} w_{i,k} \right\}. \quad (6)$$

When $\alpha = 1$, $\mathcal{L}_1(S)$ reduces to $\sum_{i \in V, j \in S} w_{i,j}$, which measures the overall similarity of summary set S to ground set V. As mentioned in Section 4.1, using such similarity measurement could possibly over-concentrate on a small portion of the document and result in a poor coverage of the whole document. As shown in Table 1, optimizing this objective function gives a ROUGE-1 F-measure score 32.44%. On the other hand, when using $\mathcal{L}_1(S)$ with an $\alpha < 1$ (the value of α was determined on DUC-03 using a grid search), a ROUGE-1 F-measure score 38.65% is achieved, which is already better than the best performing system in DUC-04.

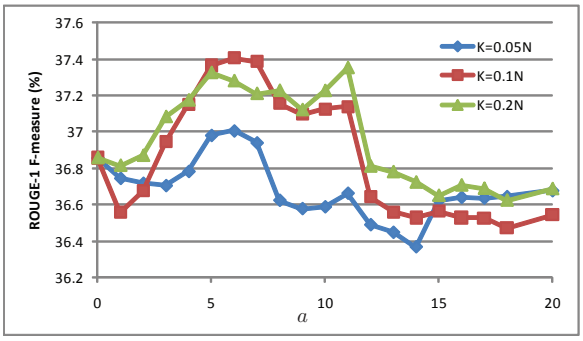


Figure 1: ROUGE-1 F-measure scores on DUC-03 when α and K vary in objective function $\mathcal{L}_1(S) + \lambda \mathcal{R}_1(S)$, where $\lambda = 6$ and $\alpha = \frac{a}{N}$.

As for the diversity reward objective, we define the singleton reward as $r_i = \frac{1}{N} \sum_j w_{i,j}$, which is the average similarity of sentence i to the rest of the document. It basically states that the more similar to the whole document a sentence is, the more reward there will be by adding this sentence to an empty summary set. By using this singleton reward, we have the following diversity reward function:

$$\mathcal{R}_1(S) = \sum_{k=1}^{K} \sqrt{\sum_{j \in S \cap P_k} \frac{1}{N} \sum_{i \in V} w_{i,j}}. \quad (7)$$

In order to generate $P_k, k = 1, \cdots K$, we used CLUTO[2] to cluster the sentences, where the IDF-weighted term vector was used as feature vector, and a direct K-mean clustering algorithm was used. In this experiment, we set $K = 0.2N$. In other words, there are 5 sentences in each cluster on average. And as we can see in Table 1, optimizing the diversity reward function alone achieves comparable performance to the DUC-04 best system.

Combining $\mathcal{L}_1(S)$ and $\mathcal{R}_1(S)$, our system outperforms the best system in DUC-04 significantly, and it also outperforms several recent systems, including a concept-based summarization approach (Takamura and Okumura, 2009), a sentence topic model based system (Wang et al., 2009), and our MMR-styled submodular system (Lin and Bilmes, 2010). Figure 1 illustrates how ROUGE-1 scores change when α and K vary on the development set (DUC-03).

[2]http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview

Table 2: ROUGE-2 recall (R) and F-measure (F) results (%) on DUC-05, where DUC-05 was used as training set.

DUC-05	R	F
$\mathcal{L}_1(S) + \lambda\mathcal{R}_Q(S)$	**8.38**	**8.31**
Daumé III and Marcu (2006)	7.62	-
Extr, Daumé et al. (2009)	7.67	-
Vine, Daumé et al. (2009)	8.24	-

Table 3: ROUGE-2 recall (R) and F-measure (F) results on DUC-05 (%). We used DUC-06 as training set.

DUC-05	R	F
$\mathcal{L}_1(S) + \lambda\mathcal{R}_Q(S)$	**7.82**	**7.72**
Daumé III and Marcu (2006)	6.98	-
Best system in DUC-05 (peer 15)	7.44	7.43

5.2 Query-focused summarization

We evaluated our approach on the task of query-focused summarization using DUC 05-07 data. In DUC-05 and DUC-06, participants were given 50 document clusters, where each cluster contains 25 news articles related to the same topic. Participants were asked to generate summaries of at most 250 words for each cluster. For each cluster, a title and a narrative describing a user's information need are provided. The narrative is usually composed of a set of questions or a multi-sentence task description. The main task in DUC-07 is the same as in DUC-06.

In DUC 05-07, ROUGE-2 was the primary criterion for evaluation, and thus we also report ROUGE-2[3] (both recall R, and precision F). Documents were processed as in Section 5.1. We used both the title and the narrative as query, where stop words, including some function words (e.g., "describe") that appear frequently in the query, were removed. All queries were then stemmed using the Porter Stemmer.

Note that there are several ways to incorporate query-focused information into both the coverage and diversity reward objectives. For instance, $\mathcal{C}_i(S)$ could be query-dependent in how it measures how much query-dependent information in i is covered by S. Also, the coefficient α could be query and sentence dependent, where it takes larger value when a sentence is more relevant to query (i.e., a larger value of α means later truncation, and therefore more possible coverage). Similarly, sentence clustering and singleton rewards in the diversity function can also

[3]ROUGE version 1.5.5 was used with option -n 2 -x -m -2 4 -u -c 95 -r 1000 -f A -p 0.5 -t 0 -d -l 250

Table 4: ROUGE-2 recall (R) and F-measure (F) results (%) on DUC-06, where DUC-05 was used as training set.

DUC-06	R	F
$\mathcal{L}_1(S) + \lambda\mathcal{R}_Q(S)$	**9.75**	**9.77**
Celikyilmaz and Hakkani-tür (2010)	9.10	-
Shen and Li (2010)	9.30	-
Best system in DUC-06 (peer 24)	9.51	9.51

Table 5: ROUGE-2 recall (R) and F-measure (F) results (%) on DUC-07. DUC-05 was used as training set for objective $\mathcal{L}_1(S) + \lambda\mathcal{R}_Q(S)$. DUC-05 and DUC-06 were used as training sets for objective $\mathcal{L}_1(S) + \sum_\kappa \lambda_\kappa \mathcal{R}_{Q,\kappa}(S)$.

DUC-07	R	F
$\mathcal{L}_1(S) + \lambda\mathcal{R}_Q(S)$	12.18	12.13
$\mathcal{L}_1(S) + \sum_{\kappa=1}^{3} \lambda_\kappa \mathcal{R}_{Q,\kappa}(S)$	12.38	**12.33**
Toutanova et al. (2007)	11.89	11.89
Haghighi and Vanderwende (2009)	11.80	-
Celikyilmaz and Hakkani-tür (2010)	11.40	-
Best system in DUC-07 (peer 15)	12.45	12.29

be query-dependent. In this experiment, we explore an objective with a query-independent coverage function ($\mathcal{R}_1(S)$), indicating prior importance, combined with a query-dependent diversity reward function, where the latter is defined as:

$$\mathcal{R}_Q(S) = \sum_{k=1}^{K} \sqrt{\sum_{j \in S \cap P_k} \left(\frac{\beta}{N} \sum_{i \in V} w_{i,j} + (1-\beta) r_{j,Q} \right)},$$

where $0 \leq \beta \leq 1$, and $r_{j,Q}$ represents the relevance between sentence j to query Q. This query-dependent reward function is derived by using a singleton reward that is expressed as a convex combination of the query-independent score ($\frac{1}{N}\sum_{i \in V} w_{i,j}$) and the query-dependent score ($r_{j,Q}$) of a sentence. We simply used the number of terms (up to a bi-gram) that sentence j overlaps the query Q as $r_{j,Q}$, where the IDF weighting is not used (i.e., every term in the query, after stop word removal, was treated as equally important). Both query-independent and query-dependent scores were then normalized by their largest value respectively such that they had roughly the same dynamic range.

To better estimate of the relevance between query and sentences, we further expanded sentences with synonyms and hypernyms of its constituent words. In particular, part-of-speech tags were obtained for each sentence using the maximum entropy part-of-speech tagger (Ratnaparkhi, 1996), and all nouns were then

expanded with their synonyms and hypernyms using WordNet (Fellbaum, 1998). Note that these expanded documents were only used in the estimation $r_{j,Q}$, and we plan to further explore whether there is benefit to use the expanded documents either in sentence similarity estimation or in sentence clustering in our future work. We also tried to expand the query with synonyms and observed a performance decrease, presumably due to noisy information in a query expression.

While it is possible to use an approach that is similar to (Toutanova et al., 2007) to learn the coefficients in our objective function, we trained all coefficients to maximize ROUGE-2 F-measure score using the Nelder-Mead (derivative-free) method. Using $\mathcal{L}_1(S)+\lambda\mathcal{R}_Q(S)$ as the objective and with the same sentence clustering algorithm as in the generic summarization experiment ($K = 0.2N$), our system, when both trained and tested on DUC-05 (results in Table 2), outperforms the Bayesian query-focused summarization approach and the search-based structured prediction approach, which were also trained and tested on DUC-05 (Daumé et al., 2009). Note that the system in (Daumé et al., 2009) that achieves its best performance (8.24% in ROUGE-2 recall) is a so called "vine-growth" system, which can be seen as an abstractive approach, whereas our system is *purely* an extractive system. Comparing to the extractive system in (Daumé et al., 2009), our system performs much better (8.38% v.s. 7.67%). More importantly, when trained only on DUC-06 and tested on DUC-05 (results in Table 3), our approach outperforms the best system in DUC-05 significantly.

We further tested the system trained on DUC-05 on both DUC-06 and DUC-07. The results on DUC-06 are shown in Table. 4. Our system outperforms the best system in DUC-06, as well as two recent approaches (Shen and Li, 2010; Celikyilmaz and Hakkani-tür, 2010). On DUC-07, in terms of ROUGE-2 score, our system outperforms PYTHY (Toutanova et al., 2007), a state-of-the-art supervised summarization system, as well as two recent systems including a generative summarization system based on topic models (Haghighi and Vanderwende, 2009), and a hybrid hierarchical summarization system (Celikyilmaz and Hakkani-tür, 2010). It also achieves comparable performance to the best DUC-07 system. Note that in the best DUC-07 system (Pingali et al., 2007; Jagarlamudi et al., 2006), an external web search engine (Yahoo!) was used to estimate a language model for query relevance. In our system, no such web search expansion was used.

To further improve the performance of our system, we used both DUC-05 and DUC-06 as a training set, and introduced three diversity reward terms into the objective where three different sentence clusterings with different resolutions were produced (with sizes $0.3N, 0.15N$ and $0.05N$). Denoting a diversity reward corresponding to clustering κ as $\mathcal{R}_{Q,\kappa}(S)$, we model the summary quality as $\mathcal{L}_1(S) + \sum_{\kappa=1}^{3} \lambda_\kappa \mathcal{R}_{Q,\kappa}(S)$. As shown in Table 5, using this objective function with multi-resolution diversity rewards improves our results further, and outperforms the best system in DUC-07 in terms of ROUGE-2 F-measure score.

6 Conclusion and discussion

In this paper, we show that submodularity naturally arises in document summarization. Not only do many existing automatic summarization methods correspond to submodular function optimization, but also the widely used ROUGE evaluation is closely related to submodular functions. As the corresponding submodular optimization problem can be solved efficiently and effectively, the remaining question is then how to design a submodular objective that best models the task. To address this problem, we introduce a powerful class of monotone submodular functions that are well suited to document summarization by modeling two important properties of a summary, fidelity and diversity. While more advanced NLP techniques could be easily incorporated into our functions (e.g., language models could define a better $\mathcal{C}_i(S)$, more advanced relevance estimations for the singleton rewards r_i, and better and/or overlapping clustering algorithms for our diversity reward), we already show top results on standard benchmark evaluations using fairly basic NLP methods (e.g., term weighting and WordNet expansion), all, we believe, thanks to the power and generality of submodular functions. As information retrieval and web search are closely related to query-focused summarization, our approach might be beneficial in those areas as well.

References

F. Bach. 2010. Structured sparsity-inducing norms through submodular functions. *Advances in Neural Information Processing Systems*.

J. Carbonell and J. Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proc. of SIGIR*.

A. Celikyilmaz and D. Hakkani-tür. 2010. A hybrid hierarchical model for multi-document summarization. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 815–824, Uppsala, Sweden, July. Association for Computational Linguistics.

H. Daumé, J. Langford, and D. Marcu. 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325.

H. Daumé III and D. Marcu. 2006. Bayesian query-focused summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, page 312.

C. Fellbaum. 1998. *WordNet: An electronic lexical database*. The MIT press.

E. Filatova and V. Hatzivassiloglou. 2004. Event-based extractive summarization. In *Proceedings of ACL Workshop on Summarization*, volume 111.

A. Haghighi and L. Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370, Boulder, Colorado, June. Association for Computational Linguistics.

J. Jagarlamudi, P. Pingali, and V. Varma. 2006. Query independent sentence scoring approach to DUC 2006. In *DUC 2006*.

S. Jegelka and J. A. Bilmes. 2011. Submodularity beyond submodular energies: coupling edges in graph cuts. In *Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, CO, June.

D. Kempe, J. Kleinberg, and E. Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the 9th Conference on SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

S. Khuller, A. Moss, and J. Naor. 1999. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45.

V. Kolmogorov and R. Zabin. 2004. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159.

A. Krause and C. Guestrin. 2005. Near-optimal nonmyopic value of information in graphical models. In *Proc. of Uncertainty in AI*.

A. Krause, H.B. McMahan, C. Guestrin, and A. Gupta. 2008. Robust submodular observation selection. *Journal of Machine Learning Research*, 9:2761–2801.

H. Lin and J. Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *North American chapter of the Association for Computational Linguistics/Human Language Technology Conference (NAACL/HLT-2010)*, Los Angeles, CA, June.

H. Lin and J. Bilmes. 2011. Word alignment via submodular maximization over matroids. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, Portland, OR, June.

H. Lin, J. Bilmes, and S. Xie. 2009. Graph-based submodular selection for extractive summarization. In *Proc. IEEE Automatic Speech Recognition and Understanding (ASRU)*, Merano, Italy, December.

C.-Y. Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*.

L. Lovász. 1983. Submodular functions and convexity. *Mathematical programming-The state of the art,(eds. A. Bachem, M. Grotschel and B. Korte) Springer*, pages 235–257.

M. Minoux. 1978. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques*, pages 234–243.

M. Narasimhan and J. Bilmes. 2005. A submodular-supermodular procedure with applications to discriminative structure learning. In *Proc. Conf. Uncertainty in Artifical Intelligence*, Edinburgh, Scotland, July. Morgan Kaufmann Publishers.

M. Narasimhan and J. Bilmes. 2007. Local search for balanced submodular clusterings. In *Twentieth International Joint Conference on Artificial Intelligence (IJCAI07)*, Hyderabad, India, January.

H. Narayanan. 1997. *Submodular functions and electrical networks*. North-Holland.

G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions I. *Mathematical Programming*, 14(1):265–294.

P. Pingali, K. Rahul, and V. Varma. 2007. IIIT Hyderabad at DUC 2007. *Proceedings of DUC 2007*.

V. Qazvinian, D.R. Radev, and A. Ozgür. 2010. Citation Summarization Through Keyphrase Extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 895–903.

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *EMNLP*, volume 1, pages 133–142.

K. Riedhammer, B. Favre, and D. Hakkani-Tür. 2010. Long story short-Global unsupervised models for keyphrase based meeting summarization. *Speech Communication*.

C. Shen and T. Li. 2010. Multi-document summarization via the minimum dominating set. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 984–992, Beijing, China, August. Coling 2010 Organizing Committee.

M. Sviridenko. 2004. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43.

H. Takamura and M. Okumura. 2009. Text summarization model based on maximum coverage problem and its variant. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 781–789. Association for Computational Linguistics.

K. Toutanova, C. Brockett, M. Gamon, J. Jagarlamudi, H. Suzuki, and L. Vanderwende. 2007. The PYTHY summarization system: Microsoft research at DUC 2007. In *the proceedings of Document Understanding Conference*.

D. Wang, S. Zhu, T. Li, and Y. Gong. 2009. Multi-document summarization using sentence-based topic models. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 297–300, Suntec, Singapore, August. Association for Computational Linguistics.

L.A. Wolsey. 1982. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393.

P. Zhao, G. Rocha, and B. Yu. 2009. Grouped and hierarchical model selection through composite absolute penalties. *Annals of Statistics*, 37(6A):3468–3497.

Semi-supervised Relation Extraction with Large-scale Word Clustering

Ang Sun **Ralph Grishman** **Satoshi Sekine**

Computer Science Department
New York University
{asun,grishman,sekine}@cs.nyu.edu

Abstract

We present a simple semi-supervised relation extraction system with large-scale word clustering. We focus on systematically exploring the effectiveness of different cluster-based features. We also propose several statistical methods for selecting clusters at an appropriate level of granularity. When training on different sizes of data, our semi-supervised approach consistently outperformed a state-of-the-art supervised baseline system.

1 Introduction

Relation extraction is an important information extraction task in natural language processing (NLP), with many practical applications. The goal of relation extraction is to detect and characterize semantic relations between pairs of entities in text. For example, a relation extraction system needs to be able to extract an *Employment* relation between the entities *US soldier* and *US* in the phrase *US soldier*.

Current supervised approaches for tackling this problem, in general, fall into two categories: feature based and kernel based. Given an entity pair and a sentence containing the pair, both approaches usually start with multiple level analyses of the sentence such as tokenization, partial or full syntactic parsing, and dependency parsing. Then the feature based method explicitly extracts a variety of lexical, syntactic and semantic features for statistical learning, either generative or discriminative (Miller et al., 2000; Kambhatla, 2004; Boschee et al., 2005; Grishman et al., 2005; Zhou et al., 2005; Jiang and Zhai, 2007). In contrast, the kernel based method does not explicitly extract features; it designs kernel functions over the structured sentence representations (sequence, dependency or parse tree) to capture the similarities between different relation instances (Zelenko et al., 2003; Bunescu and Mooney, 2005a; Bunescu and Mooney, 2005b; Zhao and Grishman, 2005; Zhang et al., 2006; Zhou et al., 2007; Qian et al., 2008). Both lines of work depend on effective features, either explicitly or implicitly.

The performance of a supervised relation extraction system is usually degraded by the sparsity of lexical features. For example, unless the example *US soldier* has previously been seen in the training data, it would be difficult for both the feature based and the kernel based systems to detect whether there is an *Employment* relation or not. Because the syntactic feature of the phrase *US soldier* is simply a noun-noun compound which is quite general, the words in it are crucial for extracting the relation.

This motivates our work to use word clusters as additional features for relation extraction. The assumption is that even if the word *soldier* may never have been seen in the annotated *Employment* relation instances, other words which share the same cluster membership with *soldier* such as *president* and *ambassador* may have been observed in the *Employment* instances. The absence of lexical features can be compensated by

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 521–529,
Portland, Oregon, June 19-24, 2011.

the cluster features. Moreover, word clusters may implicitly correspond to different relation classes. For example, the cluster of *president* may be related to the *Employment* relation as in *US president* while the cluster of *businessman* may be related to the *Affiliation* relation as in *US businessman*.

The main contributions of this paper are: we explore the cluster-based features in a systematic way and propose several statistical methods for selecting effective clusters. We study the impact of the size of training data on cluster features and analyze the performance improvements through an extensive experimental study.

The rest of this paper is organized as follows: Section 2 presents related work and Section 3 provides the background of the relation extraction task and the word clustering algorithm. Section 4 describes in detail a state-of-the-art supervised baseline system. Section 5 describes the cluster-based features and the cluster selection methods. We present experimental results in Section 6 and conclude in Section 7.

2 Related Work

The idea of using word clusters as features in discriminative learning was pioneered by Miller et al. (2004), who augmented name tagging training data with hierarchical word clusters generated by the Brown clustering algorithm (Brown et al., 1992) from a large unlabeled corpus. They used different thresholds to cut the word hierarchy to obtain clusters of various granularities for feature decoding. Ratinov and Roth (2009) and Turian et al. (2010) also explored this approach for name tagging. Though all of them used the same hierarchical word clustering algorithm for the task of name tagging and reported improvements, we noticed that the clusters used by Miller et al. (2004) were quite different from that of Ratinov and Roth (2009) and Turian et al. (2010). To our knowledge, there has not been work on selecting clusters in a principled way. We move a step further to explore several methods in choosing effective clusters. A second difference between this work and the above ones is that we utilize word clusters in the task of relation extraction which is very different from sequence labeling tasks such as name tagging and chunking.

Though Boschee et al. (2005) and Chan and Roth (2010) used word clusters in relation extraction, they shared the same limitation as the above approaches in choosing clusters. For example, Boschee et al. (2005) chose clusters of different granularities and Chan and Roth (2010) simply used a single threshold for cutting the word hierarchy. Moreover, Boschee et al. (2005) only augmented the *predicate* (typically a verb or a noun of the most importance in a relation in their definition) with word clusters while Chan and Roth (2010) performed this for any lexical feature consisting of a single word. In this paper, we systematically explore the effectiveness of adding word clusters to different lexical features.

3 Background

3.1 Relation Extraction

One of the well defined relation extraction tasks is the Automatic Content Extraction[1] (ACE) program sponsored by the U.S. government. ACE 2004 defined 7 major entity types: PER (Person), ORG (Organization), FAC (Facility), GPE (Geo-Political Entity: countries, cities, etc.), LOC (Location), WEA (Weapon) and VEH (Vehicle). An entity has three types of mention: NAM (proper name), NOM (nominal) or PRO (pronoun). A relation was defined over a pair of entity mentions within a single sentence. The 7 major relation types with examples are shown in Table 1. ACE 2004 also defined 23 relation subtypes. Following most of the previous work, this paper only focuses on relation extraction of major types.

Given a relation instance $x = (s, m_i, m_j)$, where m_i and m_j are a pair of mentions and s is the sentence containing the pair, the goal is to learn a function which maps the instance x to a type c, where c is one of the 7 defined relation types or the type *Nil* (no relation exists). There are two commonly used learning paradigms for relation extraction:

Flat: This strategy performs relation detection and classification at the same time. One multi-class classifier is trained to discriminate among the 7 relation types plus the *Nil* type.

Hierarchical: This one separates relation detection from relation classification. One binary

[1] Task definition: http://www.itl.nist.gov/iad/894.01/tests/ace/
ACE guidelines: http://projects.ldc.upenn.edu/ace/

classifier is trained first to distinguish between relation instances and non-relation instances. This can be done by grouping all the instances of the 7 relation types into a positive class and the instances of *Nil* into a negative class. Then the thresholded output of this binary classifier is used as training data for learning a multi-class classifier for the 7 relation types (Bunescu and Mooney, 2005b).

Type	Example
EMP-ORG	*US president*
PHYS	*a military base in Germany*
GPE-AFF	*U.S. businessman*
PER-SOC	*a spokesman for the senator*
DISC	*each of whom*
ART	*US helicopters*
OTHER-AFF	*Cuban-American people*

Table 1: ACE relation types and examples from the annotation guideline[2]. The heads of the two entity mentions are marked. Types are listed in decreasing order of frequency of occurrence in the ACE corpus.

3.2 Brown Word Clustering

The Brown algorithm is a hierarchical clustering algorithm which initially assigns each word to its own cluster and then repeatedly merges the two clusters which cause the least loss in average mutual information between adjacent clusters based on bigram statistics. By tracing the pairwise merging steps, one can obtain a word hierarchy which can be represented as a binary tree. A word can be compactly represented as a bit string by following the path from the root to itself in the tree, assigning a 0 for each left branch, and a 1 for each right branch. A cluster is just a branch of that tree. A high branch may correspond to more general concepts while the lower branches it includes might correspond to more specific ones.

Brown et al. (1992) described an efficient implementation based on a greedy algorithm which initially assigned only the most frequent words into distinct clusters. It is worth pointing out that in this implementation each word occupies a leaf in the hierarchy, but each leaf might contain more than one word as can be seen from Table 2. The lengths of the bit strings also vary among different words.

[2] http://projects.ldc.upenn.edu/ace/docs/EnglishRDCV4-3-2.PDF

Bit string	Examples
111011011100	***US*** *...*
1110110111011	***U.S.*** *...*
1110110110000	***American*** *...*
1110110111110110	***Cuban***, *Pakistani, Russian ...*
11111110010111	***Germany***, *Poland, Greece ...*
110111110100	***businessman***, *journalist, reporter*
1101111101111	***president,*** *governor, premier...*
1101111101100	***senator***, *soldier, ambassador ...*
11011101110	***spokesman***, *spokeswoman, ...*
11001100	***people***, *persons, miners, Haitians*
110110111011111	***base***, *compound, camps, camp ...*
110010111	***helicopters***, *tanks, Marines ...*

Table 2: An example of words and their bit string representations obtained in this paper. Words in bold are head words that appeared in Table 1.

4 Feature Based Relation Extraction

Given a pair of entity mentions $< m_i, m_j >$ and the sentence containing the pair, a feature based system extracts a feature vector v which contains diverse lexical, syntactic and semantic features. The goal is to learn a function which can estimate the conditional probability $p(c \mid v)$, the probability of a relation type c given the feature vector v. The type with the highest probability will be output as the *class label* for the mention pair.

We now describe a supervised baseline system with a very large set of features and its learning strategy.

4.1 Baseline Feature Set

We first adopted the full feature set from Zhou et al. (2005), a state-of-the-art feature based relation extraction system. For space reasons, we only show the lexical features as in Table 3 and refer the reader to the paper for the rest of the features.

At the lexical level, a relation instance can be seen as a sequence of tokens which form a five tuple *<Before, M1, Between, M2, After>*. Tokens of the five members and the interaction between the heads of the two mentions can be extracted as features as shown in Table 3.

In addition, we cherry-picked the following features which were not included in Zhou et al. (2005) but were shown to be quite effective for relation extraction.

Bigram of the words between the two mentions: This was extracted by both Zhao and Grishman (2005) and Jiang and Zhai (2007), aiming to

provide more order information of the tokens between the two mentions.

Patterns: There are three types of patterns: 1) the sequence of the tokens between the two mentions as used in Boschee et al. (2005); 2) the sequence of the heads of the constituents between the two mentions as used by Grishman et al. (2005); 3) the shortest dependency path between the two mentions in a dependency tree as adopted by Bunescu and Mooney (2005a). These patterns can provide more structured information of how the two mentions are connected.

Title list: This is tailored for the EMP-ORG type of relations as the head of one of the mentions is usually a title. The features are decoded in a way similar to that of Sun (2009).

Position	Feature	Description
Before	BM1F	*first word before* M1
	BM1L	*second word before* M1
M1	WM1	*bag-of-words in* M1
	HM1	*head*[3] *word of* M1
Between	WBNULL	*when no word in between*
	WBFL	*the only word in between when only one word in between*
	WBF	*first word in between when at least two words in between*
	WBL	*last word in between when at least two words in between*
	WBO	*other words in between except first and last words when at least three words in between*
M2	WM2	*bag-of-words in* M2
	HM2	*head word of* M2
M12	HM12	*combination of* HM1 *and* HM2
After	AM2F	*first word after* M2
	AM2L	*second word after* M2

Table 3: Lexical features for relation extraction.

4.2 Baseline Learning Strategy

We employ a simple learning framework that is similar to the *hierarchical* learning strategy as described in Section 3.1. Specifically, we first train a binary classifier to distinguish between relation instances and non-relation instances. Then rather than using the thresholded output of this binary classifier as training data, we use only the annotated relation instances to train a multi-class classifier for the 7 relation types. In the test phase, given a test instance x, we first apply the binary classifier to it for relation detection; if it is detected as a relation instance we then apply the multi-class relation classifier to classify it[4].

5 Cluster Feature Selection

The selection of cluster features aims to answer the following two questions: which lexical features should be augmented with word clusters to improve generalization accuracy? How to select clusters at an appropriate level of granularity? We will describe our solutions in Section 5.1 and 5.2.

5.1 Cluster Feature Decoding

While each one of the lexical features in Table 3 used by the baseline can potentially be augmented with word clusters, we believe the effectiveness of a lexical feature with augmentation of word clusters should be tested either individually or incrementally according to a rank of its *importance* as shown in Table 4. We will show the effectiveness of each cluster feature in the experiment section.

Importance	Lexical Feature	Description of lexical feature	Cluster Feature
1	HM	*HM1, HM2 and HM12*	HM1_WC, HM2_WC, HM12_WC
2	BagWM	*WM1 and WM2*	BagWM_WC
3	HC	*a head*[5] *of a chunk in context*	HC_WC
4	BagWC	*word of context*	BagWC_WC

Table 4: Cluster features ordered by *importance*.

The *importance* is based on linguistic intuitions and observations of the contributions of different lexical features from various feature based systems. Table 4 simplifies a relation instance as a three tuple <*Context*, *M1*, *M2*> where the *Context* includes the *Before*, *Between* and *After* from the

[3] The head word of a mention is normally set as the last word of the mention as in Zhou et al. (2005).

[4] Both the binary and multi-class classifiers output normalized probabilities in the range [0,1]. When the binary classifier's prediction probability is greater than 0.5, we take the prediction with the highest probability of the multi-class classifier as the final class label. When it is in the range [0.3,0.5], we only consider as the final class label the prediction of the multi-class classifier with a probability which is greater than 0.9. All other cases are taken as non-relation instances.

[5] The head of a chunk is defined as the last word in the chunk.

five tuple representation. As a relation in ACE is usually short, the words of the two entity mentions can provide more critical indications for relation classification than the words from the context. Within the two entity mentions, the head word of each mention is usually more important than other words of the mention; the conjunction of the two heads can provide an additional clue. And in general words other than the chunk head in the context do not contribute to establishing a relationship between the two entity mentions.

The cluster based semi-supervised system works by adding an additional layer of lexical features that incorporate word clusters as shown in column 4 of Table 4. Take the *US soldier* as an example, if we decide to use a length of 10 as a threshold to cut the Brown word hierarchy to generate word clusters, we will extract a cluster feature *HM1_WC10=1101111101* in addition to the lexical feature *HM1=soldier* given that the full bit string of *soldier* is *1101111101100* in Table 2. (Note that the cluster feature is a nominal feature, not to be confused with an integer feature.)

5.2 Selection of Clusters

Given the bit string representations of all the words in a vocabulary, researchers usually use prefixes of different lengths of the bit strings to produce word clusters of various granularities. However, how to choose the set of prefix lengths in a principled way? This has not been answered by prior work.

Our main idea is to learn the best set of prefix lengths, perhaps through the validation of their effectiveness on a development set of data. To our knowledge, previous research simply uses ad-hoc prefix lengths and lacks this training procedure. The training procedure can be extremely slow for reasons to be explained below.

Formally, let l be the set of available prefix lengths ranging from 1 bit to the length of the longest bit string in the Brown word hierarchy and let m be the set of prefix lengths we want to use in decoding cluster features, then the problem of selecting effective clusters transforms to finding a $|m|$-combination of the set l which maximizes system performance. The training procedure can be extremely time consuming if we enumerate every possible $|m|$-combination of l, given that $|m|$ can range from 1 to the size of l and the size of l equals the length of the longest bit string which is usually 20 when inducing 1,000 clusters using the Brown algorithm.

One way to achieve better efficiency is to consider only a subset of l instead of the full set. In addition, we limit ourselves to use sizes 3 and 4 for m for matching prior work. This keeps the cluster features to a manageable size considering that every word in your vocabulary could contribute to a lexical feature. For picking a subset of l, we propose below two statistical measures for computing the *importance* of a certain prefix length.

Information Gain (IG): IG measures the quality or *importance* of a feature *f* by computing the difference between the prior entropy of classes *C* and the posterior entropy, given values *V* of the feature *f* (Hunt et al., 1966; Quinlan, 1986). For our purpose, *C* is the set of relation types, *f* is a cluster-based feature with a certain prefix length such as *HM1_WC** where * means the prefix length and a value *v* is the prefix of the bit string representation of *HM1*. More formally, the IG of *f* is computed as follows:

$$IG(f) = -\sum_{c \in C} p(c) \log p(c) - \sum_{v \in V} (-p(v) \times \sum_{c \in C} p(c \mid v) \log p(c \mid v)) \quad (1)$$

where the first and second terms refer to the prior and posterior entropies respectively.

For each prefix length in the set l, we can compute its IG for a type of cluster feature and then rank the prefix lengths based on their IGs for that cluster feature. For simplicity, we rank the prefix lengths for a group of cluster features (a group is a row from column 4 in Table 4) by collapsing the individual cluster features into a single cluster feature. For example, we collapse the 3 types: *HM1_WC*, *HM2_WC* and *HM12_WC* into a single type *HM_WC* for computing the IG.

Prefix Coverage (PC): If we use a short prefix then the clusters produced correspond to the high branches in the word hierarchy and would be very general. The cluster features may not provide more informative information than the words themselves. Similarly, if we use a long prefix such as the length of the longest bit string, then maybe only a few of the lexical features can be *covered* by clusters. To capture this intuition, we define the PC of a prefix length *i* as below:

$$PC(i) = \frac{count(f_{c_i})}{count(f_l)} \quad (2)$$

where f_l stands for a lexical feature such as *HM1* and f_{c_i} a cluster feature with prefix length *i* such as *HM1_WCi*, $count(*)$ is the number of occurrences of that feature in training data.

Similar to IG, we compute PC for a group of cluster features, not for each individual feature.

In our experiments, the top 10 ranked prefix lengths based on IG and prefix lengths with PC values in the range [0.4, 0.9] were used.

In addition to the above two statistical measures, for comparison, we introduce another two simple but extreme measures for the selection of clusters.

Use All Prefixes (UA): UA produces a cluster feature at every available bit length with the hope that the underlying supervised system can *learn* proper weights of different cluster features during training. For example, if the full bit representation of "Apple" is "000", UA would produce three cluster features: *prefix1=0*, *prefix2=00* and *prefix3=000*. Because this method does not need validation on the development set, it is the laziest but the fastest method for *selecting* clusters.

Exhaustive Search (ES): ES works by trying every possible combination of the set *l* and picking the one that works the best for the development set. This is the most cautious and the slowest method for selecting clusters.

6 Experiments

In this section, we first present details of our unsupervised word clusters, the relation extraction data set and its preprocessing. We then present a series of experiments coupled with result analyses.

We used the English portion of the TDT5 corpora (LDC2006T18) as our unlabeled data for inducing word clusters. It contains roughly 83 million words in 3.4 million sentences with a vocabulary size of 450K. We left case intact in the corpora. Following previous work, we used Liang's implementation of the Brown clustering algorithm (Liang, 2005). We induced 1,000 word clusters for words that appeared at least twice in the corpora. The reduced vocabulary contains 255K unique words. The clusters are available at http://www.cs.nyu.edu/~asun/data/TDT5_BrownWC.tar.gz.

For relation extraction, we used the benchmark ACE 2004 training data. Following most of the previous research, we used in experiments the *nwire* (newswire) and *bnews* (broadcast news) genres of the data containing 348 documents and 4374 relation instances. We extracted an instance for every pair of mentions in the same sentence which were separated by no more than two other mentions. The non-relation instances generated were about 8 times more than the relation instances.

Preprocessing of the ACE documents: We used the Stanford parser[6] for syntactic and dependency parsing. We used chunklink[7] to derive chunking information from the Stanford parsing. Because some *bnews* documents are in lower case, we recover the case for the head of a mention if its type is NAM by making the first character into its upper case. This is for better matching between the words in ACE and the words in the unsupervised word clusters.

We used the OpenNLP [8] maximum entropy (maxent) package as our machine learning tool. We choose to work with maxent because the training is fast and it has a good support for multi-class classification.

6.1 Baseline Performance

Following previous work, we did 5-fold cross-validation on the 348 documents with hand-annotated entity mentions. Our results are shown in Table 5 which also lists the results of another three state-of-the-art feature based systems. For this and the following experiments, all the results were computed at the relation mention level.

System	P(%)	R(%)	F(%)
Zhou et al. (2007)[9]	78.2	63.4	70.1
Zhao and Grishman (2005)[10]	69.2	71.5	70.4
Our Baseline	73.4	67.7	70.4
Jiang and Zhai (2007) [11]	72.4	70.2	71.3

Table 5: Performance comparison on the ACE 2004 data over the 7 relation types.

[6] http://nlp.stanford.edu/software/lex-parser.shtml
[7] http://ilk.uvt.nl/team/sabine/chunklink/README.html
[8] http://opennlp.sourceforge.net/
[9] Zhou et al. (2005) tested their system on the ACE 2003 data; Zhou et al. (2007) tested their system on the ACE 2004 data.
[10] The paper gives a recall value of 70.5, which is not consistent with the given values of P and F. An examination of the correspondence in preparing this paper indicates that the correct recall value is 71.5.
[11] The result is from using the *All* features in Jiang and Zhai (2007). It is not quite clear from the paper that whether they used the 348 documents or the whole 2004 training data.

Note that although all the 4 systems did 5-fold cross-validation on the ACE 2004 data, the detailed data partition might be different. Also, we were doing cross-validation at the document level which we believe was more natural than the instance level. Nonetheless, we believe our baseline system has achieved very competitive performance.

6.2 The Effectiveness of Cluster Selection Methods

We investigated the tradeoff between performance and training time of each proposed method in selecting clusters. In this experiment, we randomly selected 70 documents from the 348 documents as test data which roughly equaled the size of 1 fold in the baseline in Section 6.1. For the baseline in this section, all the rest of the documents were used as training data. For the semi-supervised system, 70 percent of the rest of the documents were randomly selected as training data and 30 percent as development data. The set of prefix lengths that worked the best for the development set was chosen to select clusters. We only used the cluster feature *HM_WC* in this experiment.

System	F	△	Training Time (in minute)
Baseline	70.70		1
UA	71.19	+0.49	1.5
PC3	71.65	+0.95	30
PC4	71.72	+1.02	46
IG3	71.65	+0.95	45
IG4	71.68	+0.98	78
ES3	71.66	+0.96	465
ES4	71.60	+0.90	1678

Table 6: The tradeoff between performance and training time of each method in selecting clusters. PC3 means using 3 prefixes with the PC method. △ in this paper means the difference between a system and the baseline.

Table 6 shows that all the 4 proposed methods improved baseline performance, with UA as the fastest and ES as the slowest. It was interesting that ES did not always outperform the two statistical methods which might be because of its overfitting to the development set. In general, both PC and IG had good balances between performance and training time. There was no dramatic difference in performance between using 3 and 4 prefix lengths.

For the rest of this paper, we will only use PC4 as our method in selecting clusters.

6.3 The Effectiveness of Cluster Features

The baseline here is the same one used in Section 6.1. For the semi-supervised system, each test fold was the same one used in the baseline and the other 4 folds were further split into a training set and a development set in a ratio of 7:3 for selecting clusters. We first added the cluster features individually into the baseline and then added them incrementally according to the order specified in Table 4.

System		F	△
1	Baseline	70.4	
2	1 + HM_WC	71.5	+ 1.1
3	1 + BagWM_WC	71.0	+ 0.6
4	1 + HC_WC	69.6	- 0.8
5	1 + BagWC_WC	46.1	- 24.3
6	2 + BagWM_WC	71.0	+ 0.6
7	6 + HC_WC	70.6	+ 0.2
8	7+ BagWC_WC	50.3	- 20.1

Table 7: Performance [12] of the baseline and using different cluster features with PC4 over the 7 types.

We found that adding clusters to the heads of the two mentions was the most effective way of introducing cluster features. Adding clusters to the words of the mentions can also help, though not as good as the heads. We were surprised that the heads of chunks in context did not help. This might be because ACE relations are usually short and the limited number of long relations is not sufficient in generalizing cluster features. Adding clusters to every word in context hurt the performance a lot. Because of the behavior of each individual feature, it was not surprising that adding them incrementally did not give more performance gain.

For the rest of this paper, we will only use *HM_WC* as cluster features.

6.4 The Impact of Training Size

We studied the impact of training data size on cluster features as shown in Table 8. The test data was always the same as the 5-fold used in the baseline in Section 6.1. no matter the size of the training data. The training documents for the

[12] All the improvements of F in Table 7, 8 and 9 were significant at confidence levels >= 95%.

# docs	F of Relation Classification			F of Relation Detection		
	Baseline	PC4 (△)	Prefix10(△)	Baseline	PC4(△)	Prefix10(△)
50	62.9	63.8(+ 0.9)	63.7(+0.8)	71.4	71.9(+ 0.5)	71.6(+0.2)
75	62.8	64.6(+ 1.8)	63.9(+1.1)	71.5	72.3(+ 0.8)	72.5(+1.0)
125	66.1	68.1(+ 2.0)	67.5(+1.4)	74.5	74.8(+ 0.3)	74.3(-0.2)
175	67.8	69.7(+ 1.9)	69.5(+1.7)	75.2	75.5(+ 0.3)	75.2(0.0)
225	68.9	70.1(+ 1.2)	69.6(+0.7)	75.6	75.9(+ 0.3)	75.3(-0.3)
≈280	70.4	71.5(+ 1.1)	70.7(+0.3)	76.4	76.9(+ 0.5)	76.3(-0.1)

Table 8: Performance over the 7 relation types with different sizes of training data. Prefix10 uses the single prefix length 10 to generate word clusters as used by Chan and Roth (2010).

Type	P		R		F	
	Baseline	PC4 (△)	Baseline	PC4 (△)	Baseline	PC4 (△)
EMP-ORG	75.4	77.2(+1.8)	79.8	81.5(+1.7)	77.6	79.3(+1.7)
PHYS	73.2	71.2(-2.0)	61.6	60.2(-1.4)	66.9	65.3(-1.7)
GPE-AFF	67.1	69.0(+1.9)	60.0	63.2(+3.2)	63.3	65.9(+2.6)
PER-SOC	88.2	83.9(-4.3)	58.4	61.0(+2.6)	70.3	70.7(+0.4)
DISC	79.4	80.6(+1.2)	42.9	46.0(+3.2)	55.7	58.6(+2.9)
ART	87.9	96.9(+9.0)	63.0	67.4(+4.4)	73.4	79.3(+5.9)
OTHER-AFF	70.6	80.0(+9.4)	41.4	41.4(0.0)	52.2	54.6(+2.4)

Table 9: Performance of each individual relation type based on 5-fold cross-validation.

current size setup were randomly selected and added to the previous size setup (if applicable). For example, we randomly selected another 25 documents and added them to the previous 50 documents to get 75 documents. We made sure that every document participated in this experiment. The training documents for each size setup were split into a real training set and a development set in a ratio of 7:3 for selecting clusters.

There are some clear trends in Table 8. Under each training size, PC4 consistently outperformed the baseline and the system Prefix10 for relation classification. For PC4, the gain for classification was more pronounced than detection. The mixed detection results of Prefix10 indicated that only using a single prefix may not be stable.

We did not observe the same trend in the reduction of annotation need with cluster-based features as in Koo et al. (2008) for dependency parsing. PC4 with sizes 50, 125, 175 outperformed the baseline with sizes 75, 175, 225 respectively. But this was not the case when PC4 was tested with sizes 75 and 225. This might due to the complexity of the relation extraction task.

6.5 Analysis

There were on average 69 cross-type errors in the baseline in Section 6.1 which were reduced to 56 by using PC4. Table 9 showed that most of the improvements involved EMP-ORG, GPE-AFF, DISC, ART and OTHER-AFF. The performance gain for PER-SOC was not as pronounced as the other five types. The five types of relations are ambiguous as they share the same entity type GPE while the PER-SOC relation only holds between PER and PER. This reflects that word clusters can help to distinguish between ambiguous relation types.

As mentioned earlier the gain of relation detection was not as pronounced as classification as shown in Table 8. The unbalanced distribution of relation instances and non-relation instances remains as an obstacle for pushing the performance of relation extraction to the next level.

7 Conclusion and Future Work

We have described a semi-supervised relation extraction system with large-scale word clustering. We have systematically explored the effectiveness of different cluster-based features. We have also demonstrated that the two proposed statistical methods are both effective and efficient in selecting clusters at an appropriate level of granularity through an extensive experimental study.

Based on the experimental results, we plan to investigate additional ways to improve the performance of relation detection. Moreover, extending word clustering to phrase clustering (Lin and Wu, 2009) and pattern clustering (Sun and Grishman, 2010) is worth future investigation for relation extraction.

References

Rie K. Ando and Tong Zhang. 2005 A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. *Journal of Machine Learning Research*, Vol 6:1817-1853.

Elizabeth Boschee, Ralph Weischedel, and Alex Zamanian. 2005. Automatic information extraction. In *Proceedings of the International Conference on Intelligence Analysis*.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Razvan C. Bunescu and Raymond J. Mooney. 2005a. A shortest path dependency kenrel for relation extraction. In *Proceedings of HLT/EMNLP*.

Razvan C. Bunescu and Raymond J. Mooney. 2005b. Subsequence kernels for relation extraction. In *Proceedings* of *NIPS*.

Yee Seng Chan and Dan Roth. 2010. Exploiting background knowledge for relation extraction. In *Proc. of COLING*.

Ralph Grishman, David Westbrook and Adam Meyers. 2005. NYU's English ACE 2005 System Description. *ACE 2005 Evaluation Workshop.*

Earl B. Hunt, Philip J. Stone and Janet Marin. 1966. *Experiments in Induction*. New York: Academic Press, 1966.

Jing Jiang and ChengXiang Zhai. 2007. A systematic exploration of the feature space for relation extraction. In *Proceedings of HLT-NAACL-07*.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *Proceedings of ACL-04*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple Semi-supervised Dependency Parsing. In *Proceedings of ACL-08: HLT*.

Percy Liang. 2005. Semi-Supervised Learning for Natural Language. Master's thesis, Massachusetts Institute of Technology.

Dekang Lin and Xiaoyun Wu. 2009. Phrase Clustering for Discriminative Learning. In *Proc. of ACL-09.*

Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *Proc. of NAACL.*

Scott Miller, Jethran Guinness and Alex Zamanian. 2004. Name Tagging with Word Clusters and Discriminative Training. In *Proc. of HLT-NAACL.*

Longhua Qian, Guodong Zhou, Qiaoming Zhu and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction . In *Proc. of COLING.*

John Ross Quinlan. 1986. Induction of decision trees. *Machine Learning*, 1(1), 81-106.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of CoNLL-09.*

Ang Sun. 2009. A Two-stage Bootstrapping Algorithm for Relation Extraction. In *RANLP-09.*

Ang Sun and Ralph Grishman. 2010. Semi-supervised Semantic Pattern Discovery with Guidance from Unsupervised Pattern Clusters. In *Proc. of COLING.*

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL.*

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106.

Zhu Zhang. 2004. Weakly supervised relation classification for information extraction. In *Proc. of CIKM'2004*.

Min Zhang, Jie Zhang, Jian Su, and GuoDong Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of COLING-ACL-06*.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proceedings of ACL.*

Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of ACL-05.*

Guodong Zhou, Min Zhang, DongHong Ji, and QiaoMing Zhu. 2007. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of EMNLPCoNLL-07.*

In-domain Relation Discovery with Meta-constraints via Posterior Regularization

Harr Chen, Edward Benson, Tahira Naseem, and **Regina Barzilay**
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{harr, eob, tahira, regina} @csail.mit.edu

Abstract

We present a novel approach to discovering relations and their instantiations from a collection of documents in a single domain. Our approach learns relation types by exploiting *meta-constraints* that characterize the general qualities of a good relation in any domain. These constraints state that instances of a single relation should exhibit regularities at multiple levels of linguistic structure, including lexicography, syntax, and document-level context. We capture these regularities via the structure of our probabilistic model as well as a set of declaratively-specified constraints enforced during posterior inference. Across two domains our approach successfully recovers hidden relation structure, comparable to or outperforming previous state-of-the-art approaches. Furthermore, we find that a small set of constraints is applicable across the domains, and that using domain-specific constraints can further improve performance. [1]

1 Introduction

In this paper, we introduce a novel approach for the unsupervised learning of relations and their instantiations from a set of in-domain documents. Given a collection of news articles about earthquakes, for example, our method discovers relations such as the earthquake's location and resulting damage, and extracts phrases representing the relations' instantiations. Clusters of similar in-domain documents are increasingly available in forms such as Wikipedia article categories, financial reports, and biographies.

A strong earthquake rocked the Philippine island of Mindoro early Tuesday, [destroying]$_{ind}$ [some homes]$_{arg}$ ...
A strong earthquake hit the China-Burma border early Wednesday ... The official Xinhua News Agency said [some houses]$_{arg}$ were [damaged]$_{ind}$ ...
A strong earthquake with a preliminary magnitude of 6.6 shook northwestern Greece on Saturday, ... [destroying]$_{ind}$ [hundreds of old houses]$_{arg}$ ...

Figure 1: Excerpts from newswire articles about earthquakes. The indicator and argument words for the *damage* relation are highlighted.

In contrast to previous work, our approach learns from *domain-independent meta-constraints* on relation expression, rather than supervision specific to particular relations and their instances. In particular, we leverage the linguistic intuition that documents in a single domain exhibit regularities in how they express their relations. These regularities occur both in the relations' lexical and syntactic realizations as well as at the level of document structure. For instance, consider the *damage* relation excerpted from earthquake articles in Figure 1. Lexically, we observe similar words in the instances and their contexts, such as "destroying" and "houses." Syntactically, in two instances the relation instantiation is the dependency child of the word "destroying." On the discourse level, these instances appear toward the beginning of their respective documents. In general, valid relations in many domains are characterized by these coherence properties.

We capture these regularities using a Bayesian model where the underlying relations are repre-

[1] The source code for this work is available at: http://groups.csail.mit.edu/rbg/code/relation_extraction/

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 530–540,
Portland, Oregon, June 19-24, 2011.

sented as latent variables. The model takes as input a constituent-parsed corpus and explains how the constituents arise from the latent variables. Each relation instantiation is encoded by the variables as a relation-evoking *indicator* word (*e.g.*, "destroying") and corresponding *argument* constituent (*e.g.*, "some homes").[2] Our approach capitalizes on relation regularity in two ways. First, the model's generative process encourages coherence in the local features and placement of relation instances. Second, we apply posterior regularization (Graça et al., 2007) during inference to enforce higher-level declarative constraints, such as requiring indicators and arguments to be syntactically linked.

We evaluate our approach on two domains previously studied for high-level document structure analysis, news articles about earthquakes and financial markets. Our results demonstrate that we can successfully identify domain-relevant relations. We also study the importance and effectiveness of the declaratively-specified constraints. In particular, we find that a small set of declarative constraints are effective across domains, while additional domain-specific constraints yield further benefits.

2 Related Work

Extraction with Reduced Supervision Recent research in information extraction has taken large steps toward reducing the need for labeled data. Examples include using bootstrapping to amplify small seed sets of example outputs (Agichtein and Gravano, 2000; Yangarber et al., 2000; Bunescu and Mooney, 2007; Zhu et al., 2009), leveraging existing databases that overlap with the text (Mintz et al., 2009; Yao et al., 2010), and learning general domain-independent knowledge bases by exploiting redundancies in large web and news corpora (Hasegawa et al., 2004; Shinyama and Sekine, 2006; Banko et al., 2007; Yates and Etzioni, 2009).

Our approach is distinct in both the supervision and data we operate over. First, in contrast to bootstrapping and database matching approaches, we learn from meta-qualities, such as low variability in syntactic patterns, that characterize a good relation. We hypothesize that these properties hold across relations in different domains. Second, in contrast to work that builds general relation databases from heterogeneous corpora, our focus is on learning the relations salient in a single domain. Our setup is more germane to specialized domains expressing information not broadly available on the web.

Earlier work in unsupervised information extraction has also leveraged meta-knowledge independent of specific relation types, such as declaratively-specified syntactic patterns (Riloff, 1996), frequent dependency subtree patterns (Sudo et al., 2003), and automatic clusterings of syntactic patterns (Lin and Pantel, 2001; Zhang et al., 2005) and contexts (Chen et al., 2005; Rosenfeld and Feldman, 2007). Our approach incorporates a broader range of constraints and balances constraints with underlying patterns learned from the data, thereby requiring more sophisticated machinery for modeling and inference.

Extraction with Constraints Previous work has recognized the appeal of applying declarative constraints to extraction. In a supervised setting, Roth and Yih (2004) induce relations by using linear programming to impose global declarative constraints on the output from a set of classifiers trained on local features. Chang et al. (2007) propose an objective function for semi-supervised extraction that balances likelihood of labeled instances and constraint violation on unlabeled instances. Recent work has also explored how certain kinds of supervision can be formulated as constraints on model posteriors. Such constraints are not declarative, but instead based on annotations of words' majority relation labels (Mann and McCallum, 2008) and pre-existing databases with the desired output schema (Bellare and McCallum, 2009). In contrast to previous work, our approach explores a different class of constraints that does not rely on supervision that is specific to particular relation types and their instances.

3 Model

Our work performs in-domain relation discovery by leveraging regularities in relation expression at the lexical, syntactic, and discourse levels. These regularities are captured via two components: a probabilistic model that explains how documents are generated from latent relation variables and a technique

[2] We do not use the word "argument" in the syntactic sense—a relation's argument may or may not be the syntactic dependency argument of its indicator.

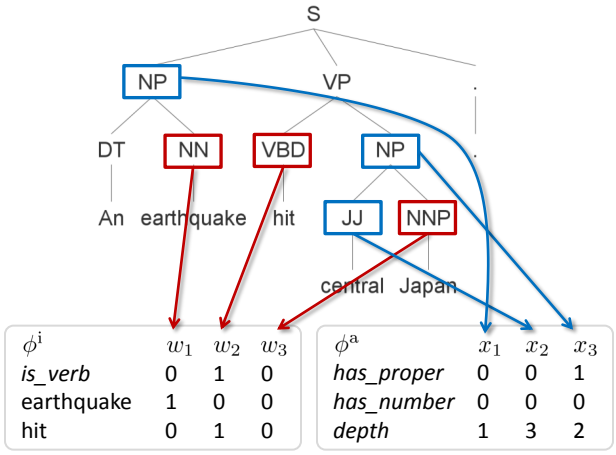


Figure 2: Words w and constituents x of syntactic parses are represented with indicator features ϕ^{i} and argument features ϕ^{a} respectively. A single relation instantiation is a pair of indicator w and argument x; we filter w to be nouns and verbs and x to be noun phrases and adjectives.

for biasing inference to adhere to declaratively-specified constraints on relation expression. This section describes the generative process, while Sections 4 and 5 discuss declarative constraints.

3.1 Problem Formulation

Our input is a corpus of constituent-parsed documents and a number K of relation types. The output is K clusters of semantically related relation instantiations. We represent these instantiations as a pair of *indicator* word and *argument* sequence from the same sentence. The indicator's role is to anchor a relation and identify its type. We only allow nouns or verbs to be indicators. For instance, in the earthquake domain a likely indicator for *damage* would be "destroyed." The argument is the actual relation value, *e.g.*, "some homes," and corresponds to a noun phrase or adjective.[3]

Along with the document parse trees, we utilize a set of features $\phi^{\text{i}}(w)$ and $\phi^{\text{a}}(x)$ describing each potential indicator word w and argument constituent x, respectively. An example feature representation is shown in Figure 2. These features can encode words, part-of-speech tags, context, and so on. Indicator and argument feature definitions need not be the same (*e.g.*, *has_number* is important for arguments but irrelevant for indicators).[4]

3.2 Generative Process

Our model associates each relation type k with a set of *feature distributions* θ_k and a *location distribution* λ_k. Each instantiation's indicator and argument, and its position within a document, are drawn from these distributions. By sharing distributions within each relation, the model places high probability mass on clusters of instantiations that are coherent in features and position. Furthermore, we allow at most one instantiation per document and relation, so as to target relations that are relevant to the entire document.

There are three steps to the generative process. First, we draw feature and location distributions for each relation. Second, an instantiation is selected for every pair of document d and relation k. Third, the indicator features of each word and argument features of each constituent are generated based on the relation parameters and instantiations. Figure 3 presents a reference for the generative process.

Generating Relation Parameters Each relation k is associated with four feature distribution parameter vectors: θ^{i}_k for indicator words, θ^{bi}_k for non-indicator words, θ^{a}_k for argument constituents, and θ^{ba}_k for non-argument constituents. Each of these is a set of multinomial parameters per feature drawn from a symmetric Dirichlet prior. A likely indicator word should have features that are highly probable according to θ^{i}_k, and likewise for arguments and θ^{a}_k. Parameters θ^{bi}_k and θ^{ba}_k represent *background* distributions for non-relation words and constituents, similar in spirit to other uses of background distributions that filter out irrelevant words (Che, 2006).[5] By drawing each instance from these distributions, we encourage the relation to be coherent in local lexical and syntactic properties.

Each relation type k is also associated with a parameter vector λ_k over document segments drawn from a symmetric Dirichlet prior. Documents are divided into L equal-length segments; λ_k states how likely relation k is for each segment, with one null outcome for the relation not occurring in the document. Because λ_k is shared within a relation, its

[3] In this paper we focus on unary relations; binary relations can be modeled with extensions of the hidden variables and constraints.

[4] We consider only categorical features here, though the extension to continuous or ordinal features is straightforward.

[5] We use separate background distributions for each relation to make inference more tractable.

For each relation type k:

- For each indicator feature ϕ^{i} draw feature distributions $\theta^{\text{i}}_{k,\phi^{\text{i}}}, \theta^{\text{bi}}_{k,\phi^{\text{i}}} \sim \text{Dir}(\theta_0)$
- For each argument feature ϕ^{a} draw feature distributions $\theta^{\text{a}}_{k,\phi^{\text{a}}}, \theta^{\text{ba}}_{k,\phi^{\text{a}}} \sim \text{Dir}(\theta_0)$
- Draw location distribution $\lambda_k \sim \text{Dir}(\lambda_0)$

For each relation type k and document d:

- Select document segment $s_{d,k} \sim \text{Mult}(\lambda_k)$
- Select sentence $z_{d,k}$ uniformly from segment $s_{d,k}$, and indicator $i_{d,k}$ and argument $a_{d,k}$ uniformly from sentence $z_{d,k}$

For each word w in every document d:

- Draw each indicator feature $\phi^{\text{i}}(w) \sim \text{Mult}\left(\frac{1}{Z}\prod_{k=1}^{K}\theta_{k,\phi^{\text{i}}}\right)$, where $\theta_{k,\phi^{\text{i}}}$ is $\theta^{\text{i}}_{k,\phi^{\text{i}}}$ if $i_{d,k} = w$ and $\theta^{\text{bi}}_{k,\phi^{\text{i}}}$ otherwise

For each constituent x in every document d:

- Draw each argument feature $\phi^{\text{a}}(x) \sim \text{Mult}\left(\frac{1}{Z}\prod_{k=1}^{K}\theta_{k,\phi^{\text{a}}}\right)$, where $\theta_{k,\phi^{\text{a}}}$ is $\theta^{\text{a}}_{k,\phi^{\text{a}}}$ if $a_{d,k} = x$ and $\theta^{\text{ba}}_{k,\phi^{\text{a}}}$ otherwise

Figure 3: The generative process for model parameters and features. In the above Dir and Mult refer respectively to the Dirichlet distribution and multinomial distribution. Fixed hyperparameters are subscripted with zero.

instances will tend to occur in the same relative positions across documents. The model can learn, for example, that a particular relation typically occurs in the first quarter of a document (if $L = 4$).

Generating Relation Instantiations For every relation type k and document d, we first choose which portion of the document (if any) contains the instantiation by drawing a document segment $s_{d,k}$ from λ_k. Our model only draws one instantiation per pair of k and d, so each discovered instantiation within a document is a separate relation. We then choose the specific sentence $z_{d,k}$ uniformly from within the segment, and the indicator word $i_{d,k}$ and argument constituent $a_{d,k}$ uniformly from within that sentence.

Generating Text Finally, we draw the feature values. We make a Naïve Bayes assumption between features, drawing each independently conditioned on relation structure. For a word w, we want all relations to be able to influence its generation. Toward this end, we compute the element-wise product of feature parameters across relations $k = 1, \ldots, K$, using indicator parameters θ^{i}_k if relation k selected w as an indicator word (if $i_{d,k} = w$) and background parameters θ^{bi}_k otherwise. The result is then normalized to form a valid multinomial that produces word w's features. Constituents are drawn similarly from every relations' argument distributions.

4 Inference with Constraints

The model presented above leverages relation regularities in local features and document placement. However, it is unable to specify global syntactic preferences about relation expression, such as indicators and arguments being in the same clause. Another issue with this model is that different relations could overlap in their indicators and arguments.[6]

To overcome these obstacles, we apply declarative constraints by imposing inequality constraints on expectations of the posterior during inference using *posterior regularization* (Graça et al., 2007). In this section we present the technical details of the approach; Section 5 explains the specific linguistically-motivated constraints we consider.

4.1 Inference with Posterior Regularization

We first review how posterior regularization impacts the variational inference procedure in general. Let θ, z, and x denote the parameters, hidden structure, and observations of an arbitrary model. We are interested in estimating the posterior distribution $p(\theta, z \mid x)$ by finding a distribution $q(\theta, z) \in \mathcal{Q}$ that is minimal in KL-divergence to the true posterior:

$$\begin{aligned}&\text{KL}(q(\theta, z) \parallel p(\theta, z \mid x)) \\ &= \int q(\theta, z) \log \frac{q(\theta, z)}{p(\theta, z, x)} d\theta dz + \log p(x). \quad (1)\end{aligned}$$

For tractability, variational inference typically makes a mean-field assumption that restricts the set $\mathcal{Q}$ to distributions where θ and z are independent, *i.e.*, $q(\theta, z) = q(\theta)q(z)$. We then optimize equation 1 by coordinate-wise descent on $q(\theta)$ and $q(z)$.

To incorporate constraints into inference, we further restrict $\mathcal{Q}$ to distributions that satisfy a given

[6] In fact, a true *maximum a posteriori* estimate of the model parameters would find the same most salient relation over and over again for every k, rather than finding K different relations.

set of inequality constraints, each of the form $\mathbb{E}_q[f(z)] \leq b$. Here, $f(z)$ is a deterministic function of z and b is a user-specified threshold. Inequalities in the opposite direction simply require negating $f(z)$ and b. For example, we could apply a syntactic constraint of the form $\mathbb{E}_q[f(z)] \geq b$, where $f(z)$ counts the number of indicator/argument pairs that are syntactically connected in a pre-specified manner (*e.g.*, the indicator and argument modify the same verb), and b is a fixed threshold.

Given a set $\mathcal{C}$ of constraints with functions $f_c(z)$ and thresholds b_c, the updates for $q(\theta)$ and $q(z)$ from equation 1 are as follows:

$$q(\theta) = \underset{q(\theta)}{\operatorname{argmin}} \operatorname{KL}\big(q(\theta) \parallel q'(\theta)\big), \tag{2}$$

where $q'(\theta) \propto \exp \mathbb{E}_{q(z)}[\log p(\theta, z, x)]$, and

$$q(z) = \underset{q(z)}{\operatorname{argmin}} \operatorname{KL}\big(q(z) \parallel q'(z)\big) \quad s.t. \quad \mathbb{E}_{q(z)}[f_c(z)] \leq b_c, \ \ \forall c \in \mathcal{C}, \tag{3}$$

where $q'(z) \propto \exp \mathbb{E}_{q(\theta)}[\log p(\theta, z, x)]$. Equation 2 is not affected by the posterior constraints and is updated by setting $q(\theta)$ to $q'(\theta)$. We solve equation 3 in its dual form (Graça et al., 2007):

$$\underset{\kappa}{\operatorname{argmin}} \sum_{c \in \mathcal{C}} \kappa_c b_c + \log \sum_z q'(z) e^{-\sum_{c \in \mathcal{C}} \kappa_c f_c(z)} \quad s.t. \quad \kappa_c \geq 0, \ \ \forall c \in \mathcal{C}. \tag{4}$$

With the box constraints of equation 4, a numerical optimization procedure such as L-BFGS-B (Byrd et al., 1995) can be used to find optimal dual parameters κ^*. The original $q(z)$ is then updated to $q'(z) \exp\big(-\sum_{c \in \mathcal{C}} \kappa_c^* f_c(z)\big)$ and renormalized.

4.2 Updates for our Model

Our model uses this mean-field factorization:

$$q(\theta, \lambda, z, a, i) = \prod_{k=1}^{K} q(\lambda_k; \hat{\lambda}_k) q(\theta_k^{\text{i}}; \hat{\theta}_k^{\text{i}}) q(\theta_k^{\text{bi}}; \hat{\theta}_k^{\text{bi}}) q(\theta_k^{\text{a}}; \hat{\theta}_k^{\text{a}}) q(\theta_k^{\text{ba}}; \hat{\theta}_k^{\text{ba}}) \times \prod_d q(z_{d,k}, a_{d,k}, i_{d,k}; \hat{c}_{d,k}) \tag{5}$$

In the above, $\hat{\lambda}$ and $\hat{\theta}$ are Dirichlet distribution parameters, and $\hat{c}$ are multinomial parameters. Note that we do not factorize the distribution of z, i, and a for a single document and relation, instead representing their joint distribution with a single set of variational parameters $\hat{c}$. This is tractable because a single relation occurs only once per document, reducing the joint search space of z, i, and a. The factors in equation 5 are updated one at a time while holding the other factors fixed.

Updating $\hat{\theta}$ Due to the Naïve Bayes assumption between features, each feature's $q(\theta)$ distributions can be updated separately. However, the product between feature parameters of different relations introduces a nonconjugacy in the model, precluding a closed form update. Instead we numerically optimize equation 1 with respect to each $\hat{\theta}$, similarly to previous work (Boyd-Graber and Blei, 2008). For instance, $\hat{\theta}^{\text{i}}_{k,\phi}$ of relation k and feature ϕ is updated by finding the gradient of equation 1 with respect to $\hat{\theta}^{\text{i}}_{k,\phi}$ and applying L-BFGS. Parameters $\hat{\theta}^{\text{bi}}$, $\hat{\theta}^{\text{a}}$, and $\hat{\theta}^{\text{ba}}$ are updated analogously.

Updating $\hat{\lambda}$ This update follows the standard closed form for Dirichlet parameters:

$$\hat{\lambda}_{k,\ell} = \lambda_0 + \mathbb{E}_{q(z,a,i)}[C_\ell(z, a, i)], \tag{6}$$

where C_ℓ counts the number of times z falls into segment ℓ of a document.

Updating $\hat{c}$ Parameters $\hat{c}$ are updated by first computing an unconstrained update $q'(z, a, i; \hat{c}')$:

$$\hat{c}'_{d,k,(z,a,i)} \propto \exp\Bigg(\mathbb{E}_{q(\lambda_k)}[\log p(z, a, i \mid \lambda_k)] + \mathbb{E}_{q(\theta^{\text{i}}_k)}[\log p(i \mid \theta^{\text{i}}_k)] + \sum_{w \neq i} \mathbb{E}_{q(\theta^{\text{bi}}_k)}[\log p(w \mid \theta^{\text{bi}}_k)] + \mathbb{E}_{q(\theta^{\text{a}}_k)}[\log p(a \mid \theta^{\text{a}}_k)] + \sum_{x \neq a} \mathbb{E}_{q(\theta^{\text{ba}}_k)}[\log p(x \mid \theta^{\text{ba}}_k)] \Bigg)$$

We then perform the minimization on the dual in equation 4 under the provided constraints to derive a final update to the constrained $\hat{c}$.

Simplifying Approximation The update for $\hat{\theta}$ requires numerical optimization due to the nonconjugacy introduced by the point-wise product in feature generation. If instead we have every relation type separately generate a copy of the corpus, the $\hat{\theta}$

	Quantity	$f(z, a, i)$	$\leq$ or $\geq$	b
Syntax	$\forall k$	Counts i, a of relation k that match a pattern (see text)	$\geq$	$0.8D$
Prevalence	$\forall k$	Counts instantiations of relation k	$\geq$	$0.8D$
Separation (ind)	$\forall w$	Counts times w selected as i	$\leq$	2
Separation (arg)	$\forall w$	Counts times w selected as part of a	$\leq$	1

Table 1: Each constraint takes the form $\mathbb{E}_q[f(z, a, i)] \leq b$ or $\mathbb{E}_q[f(z, a, i)] \geq b$; D denotes the number of corpus documents, $\forall k$ means one constraint per relation type, and $\forall w$ means one constraint per token in the corpus.

updates becomes closed-form expressions similar to equation 6. This approximation yields similar parameter estimates as the true updates while vastly improving speed, so we use it in our experiments.

5 Declarative Constraints

We now have the machinery to incorporate a variety of declarative constraints during inference. The classes of domain-independent constraints we study are summarized in Table 1. For the proportion constraints we arbitrarily select a threshold of 80% without any tuning, in the spirit of building a domain-independent approach.

Syntax As previous work has observed, most relations are expressed using a limited number of common syntactic patterns (Riloff, 1996; Banko and Etzioni, 2008). Our syntactic constraint captures this insight by requiring that a certain proportion of the induced instantiations for each relation match one of these syntactic patterns:

- The indicator is a verb and the argument's headword is either the child or grandchild of the indicator word in the dependency tree.
- The indicator is a noun and the argument is a modifier or complement.
- The indicator is a noun in a verb's subject and the argument is in the corresponding object.

Prevalence For a relation to be domain-relevant, it should occur in numerous documents across the corpus, so we institute a constraint on the number of times a relation is instantiated. Note that the effect of this constraint could also be achieved by tuning the prior probability of a relation not occurring in a document. However, this prior would need to be adjusted every time the number of documents or feature selection changes; using a constraint is an appealing alternative that is portable across domains.

Separation The separation constraint encourages diversity in the discovered relation types by restricting the number of times a single word can serve as either an indicator or part of the argument of a relation instance. Specifically, we require that every token of the corpus occurs at most once as a word in a relation's argument in expectation. On the other hand, a single word can sometimes be evocative of multiple relations (*e.g.*, "occurred" signals both *date* and *time* in "occurred on Friday at 3pm"). Thus, we allow each word to serve as an indicator more than once, arbitrarily fixing the limit at two.

6 Experimental Setup

Datasets and Metrics We evaluate on two datasets, financial market reports and newswire articles about earthquakes, previously used in work on high-level content analysis (Barzilay and Lee, 2004; Lapata, 2006). *Finance* articles chronicle daily market movements of currencies and stock indexes, and *earthquake* articles document specific earthquakes. Constituent parses are obtained automatically using the Stanford parser (Klein and Manning, 2003) and then converted to dependency parses using the PennConvertor tool (Johansson and Nugues, 2007). We manually annotated relations for both corpora, selecting relation types that occurred frequently in each domain. We found 15 types for *finance* and 9 for *earthquake*. Corpus statistics are summarized below, and example relation types are shown in Table 2.

	Docs	Sent/Doc	Tok/Doc	Vocab
Finance	100	12.1	262.9	2918
Earthquake	200	9.3	210.3	3155

In our task, annotation conventions for desired output relations can greatly impact token-level performance, and the model cannot learn to fit a particular convention by looking at example data. For example, earthquakes times are frequently reported in both local and GMT, and either may be arbitrarily chosen as correct. Moreover, the baseline we

Finance	Bond	104.58 yen, 98.37 yen
	Dollar Change	up 0.52 yen, down 0.01 yen
	Tokyo Index Change	down 5.38 points or 0.41 percent, up 0.16 points, insignificant in percentage terms
Earthquake	Damage	about 10000 homes, some buildings, no information
	Epicenter	Patuca about 185 miles (300 kilometers) south of Quito, 110 kilometers (65 miles) from shore under the surface of the Flores sea in the Indonesian archipelago
	Magnitude	5.7, 6, magnitude-4

Table 2: Example relation types identified in the *finance* and *earthquake* datasets with example instance arguments.

compare against produces lambda calculus formulas rather than spans of text as output, so a token-level comparison requires transforming its output.

For these reasons, we evaluate on both *sentence-level* and *token-level* precision, recall, and F-score. Precision is measured by mapping every induced relation cluster to its closest gold relation and computing the proportion of predicted sentences or words that are correct. Conversely, for recall we map every gold relation to its closest predicted relation and find the proportion of gold sentences or words that are predicted. This mapping technique is based on the many-to-one scheme used for evaluating unsupervised part-of-speech induction (Johnson, 2007). Note that sentence-level scores are always at least as high as token-level scores, since it is possible to select a sentence correctly but none of its true relation tokens while the opposite is not possible.

Domain-specific Constraints On top of the cross-domain constraints from Section 5, we study whether imposing basic domain-specific constraints can be beneficial. The *finance* dataset is heavily quantitative, so we consider applying a single domain-specific constraint stating that most relation arguments should include a number. Likewise, *earthquake* articles are typically written with a majority of the relevant information toward the beginning of the document, so its domain-specific constraint is that most relations should occur in the first two sentences of a document. Note that these domain-specific constraints are not specific to individual relations or instances, but rather encode a preference across all relation types. In both cases, we again use an 80% threshold without tuning.

Features For indicators, we use the word, part of speech, and word stem. For arguments, we use the word, syntactic constituent label, the head word of the parent constituent, and the dependency label of the argument to its parent.

Baselines We compare against three alternative unsupervised approaches. Note that the first two only identify relation-bearing sentences, not the specific words that participate in the relation.

Clustering (CLUTO): A straightforward way of identifying sentences bearing the same relation is to simply cluster them. We implement a clustering baseline using the CLUTO toolkit with word and part-of-speech features. As with our model, we set the number of clusters K to the true number of relation types.

Mallows Topic Model (MTM): Another technique for grouping similar sentences is the Mallows-based topic model of Chen et al. (2009). The datasets we consider here exhibit high-level regularities in content organization, so we expect that a topic model with global constraints could identify plausible clusters of relation-bearing sentences. Again, K is set to the true number of relation types.

Unsupervised Semantic Parsing (USP): Our final unsupervised comparison is to USP, an unsupervised deep semantic parser introduced by Poon and Domingos (2009). USP induces a lambda calculus representation of an entire corpus and was shown to be competitive with open information extraction approaches (Lin and Pantel, 2001; Banko et al., 2007). We give USP the required Stanford dependency format as input (de Marneffe and Manning, 2008). We find that the results are sensitive to the cluster granularity prior, so we tune this parameter and report the best-performing runs.

We recognize that USP targets a different output representation than ours: a hierarchical semantic structure over the entirety of a dependency-parsed text. In contrast, we focus on discovering a limited number K of domain-relevant relations expressed as constituent phrases. Despite these differences, both

methods ultimately aim to capture domain-specific relations expressed with varying verbalizations, and both operate over in-domain input corpora supplemented with syntactic information. For these reasons, USP provides a clear and valuable point of comparison. For this comparison, we transform USP's lambda calculus formulas to relation spans as follows. First, we group lambda forms by a combination of core form, argument form, and the parent's core form.[7] We then filter to the K relations that appear in the most documents. For token-level evaluation we take the dependency tree fragment corresponding to the lambda form. For example, in the sentence "a strong earthquake rocked the Philippines island of Mindoro early Tuesday," USP learns that the word "Tuesday" has a core form corresponding to words {*Tuesday*, *Wednesday*, *Saturday*}, a parent form corresponding to words {*shook*, *rock*, *hit*, *jolt*}, and an argument form of TMOD; all phrases with this same combination are grouped as a relation.

Training Regimes and Hyperparameters For each run of our model we perform three random restarts to convergence and select the posterior with lowest final free energy. We fix K to the true number of annotated relation types for both our model and USP and L (the number of document segments) to five. Dirichlet hyperparameters are set to 0.1.

7 Results

Table 3's first two sections present the results of our main evaluation. For *earthquake*, the far more difficult domain, our base model with only the domain-independent constraints strongly outperforms all three baselines across both metrics. For *finance*, the CLUTO and USP baselines achieve performance comparable to or slightly better than our base model. Our approach, however, has the advantage of providing a formalism for seamlessly incorporating additional arbitrary domain-specific constraints. When we add such constraints (denoted as *model+DSC*), we achieve consistently higher performance than all baselines across both datasets and metrics, demonstrating that this approach provides a simple and effective framework for injecting domain knowledge into relation discovery.

The first two baselines correspond to a setup where the number of sentence clusters K is set to the true number of relation types. This has the effect of lowering precision because each sentence must be assigned a cluster. To mitigate this impact, we experimented with using $K + N$ clusters, with N ranging from 1 to 30. In each case, we then keep only the K largest clusters. For the *earthquake* dataset, increasing N improves performance until some point, after which performance degrades. However, the best F-Score corresponding to the optimal number of clusters is 42.2, still far below our model's 66.0 F-score. For the *finance* domain, increasing the number of clusters hurts performance.

Our results show a large gap in F-score between the sentence and token-level evaluations for both the USP baseline and our model. A qualitative analysis of the results indicates that our model often picks up on regularities that are difficult to distinguish without relation-specific supervision. For *earthquake*, a *location* may be annotated as "the Philippine island of Mindoro" while we predict just the word "Mindoro." For *finance*, an *index change* can be annotated as "30 points, or 0.8 percent," while our model identifies "30 points" and "0.8 percent" as separate relations. In practice, these outputs are all plausible discoveries, and a practitioner desiring specific outputs could impose additional constraints to guide relation discovery toward them.

The Impact of Constraints To understand the impact of the declarative constraints, we perform an ablation analysis on the constraint sets. We consider removing the constraints on syntactic patterns (*no-syn*) and the constraints disallowing relations to overlap (*no-sep*) from the full domain-independent model.[8] We also try a version with hard syntactic constraints (*hard-syn*), which requires that *every* extraction match one of the three syntactic patterns specified by the syntactic constraint.

Table 3's bottom section presents the results of this evaluation. The model's performance degrades when either of the two constraint sets are removed, demonstrating that the constraints are in fact beneficial for relation discovery. Additionally, in the *hard-syn* case, performance drops dramatically for *finance*

[7] This grouping mechanism yields better results than only grouping by core form.

[8] Prevalence constraints are always enforced, as otherwise the prior on not instantiating a relation would need to be tuned.

	Finance						Earthquake					
	Sentence-level			Token-level			Sentence-level			Token-level		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
Model	82.1	59.7	69.2	42.2	23.9	30.5	54.2	68.1	60.4	20.2	16.8	18.3
Model+DSC	87.3	81.6	**84.4**	51.8	30.0	**38.0**	66.4	65.6	**66.0**	22.6	23.1	**22.8**
CLUTO	56.3	92.7	70.0	—	—	—	19.8	58.0	29.5	—	—	—
MTM	40.4	99.3	57.5	—	—	—	18.6	74.6	29.7	—	—	—
USP	91.3	66.1	76.7	28.5	32.6	30.4	61.2	43.5	50.8	9.9	32.3	15.1
No-sep	97.8	35.4	52.0	86.1	8.7	15.9	42.2	21.9	28.8	16.1	4.6	7.1
No-syn	83.3	46.1	59.3	20.8	9.9	13.4	53.8	60.9	57.1	14.0	13.8	13.9
Hard-syn	47.7	39.0	42.9	11.6	7.0	8.7	55.0	66.2	60.1	20.1	17.3	18.6

Table 3: Top section: our model, with and without domain-specific constraints (DSC). Middle section: The three baselines. Bottom section: ablation analysis of constraint sets for our model. For all scores, higher is better.

while remaining almost unchanged for *earthquake*. This suggests that formulating constraints as soft inequalities on posterior expectations gives our model the flexibility to accommodate both the underlying signal in the data and the declarative constraints.

Comparison against Supervised CRF Our final set of experiments compares a *semi-supervised* version of our model against a conditional random field (CRF) model. The CRF model was trained using the same features as our model's argument features. To incorporate training examples in our model, we simply treat annotated relation instances as observed variables. For both the baselines and our model, we experiment with using up to 10 annotated documents. At each of those levels of supervision, we average results over 10 randomly drawn training sets.

At the sentence level, our model compares very favorably to the supervised CRF. For *finance*, it takes at least 10 annotated documents (corresponding to roughly 130 annotated relation instances) for the CRF to match the semi-supervised model's performance. For *earthquake*, using even 10 annotated documents (about 71 relation instances) is not sufficient to match our model's performance.

At the token level, the supervised CRF baseline is far more competitive. Using a single labeled document (13 relation instances) yields superior performance to either of our model variants for *finance*, while four labeled documents (29 relation instances) do the same for *earthquake*. This result is not surprising—our model makes strong domain-independent assumptions about how underlying patterns of regularities in the text connect to relation expression. Without domain-specific supervision such assumptions are necessary, but they can prevent the model from fully utilizing available labeled instances. Moreover, being able to annotate even a single document requires a broad understanding of every relation type germane to the domain, which can be infeasible when there are many unfamiliar, complex domains to process.

In light of our strong sentence-level performance, this suggests a possible human-assisted application: use our model to identify promising relation-bearing sentences in a new domain, then have a human annotate those sentences for use by a supervised approach to achieve optimal token-level extraction.

8 Conclusions

This paper has presented a constraint-based approach to in-domain relation discovery. We have shown that a generative model augmented with declarative constraints on the model posterior can successfully identify domain-relevant relations and their instantiations. Furthermore, we found that a single set of constraints can be used across divergent domains, and that tailoring constraints specific to a domain can yield further performance benefits.

Acknowledgements

The authors gratefully acknowledge the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0172. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government. Thanks also to Hoifung Poon and the members of the MIT NLP group for their suggestions and comments.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of DL*.

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Proceedings of ACL*.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of IJCAI*.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceedings of HLT/NAACL*.

Kedar Bellare and Andrew McCallum. 2009. Generalized expectation criteria for bootstrapping extractors using record-text alignment. In *Proceedings of EMNLP*.

Jordan Boyd-Graber and David M. Blei. 2008. Syntactic topic models. In *Advances in NIPS*.

Razvan C. Bunescu and Raymond J. Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of ACL*.

Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proceedings of ACL*.

2006. Modeling general and specific aspects of documents with a probabilistic topic model. In *Advances in NIPS*.

Jinxiu Chen, Dong-Hong Ji, Chew Lim Tan, and Zheng-Yu Niu. 2005. Automatic relation extraction with model order selection and discriminative label identification. In *Proceedings of IJCNLP*.

Harr Chen, S.R.K. Branavan, Regina Barzilay, and David R. Karger. 2009. Content modeling using latent permutations. *Journal of Artificial Intelligence Research*, 36:129–163.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *Proceedings of the COLING Workshop on Cross-framework and Cross-domain Parser Evaluation*.

João Graça, Kuzman Ganchev, and Ben Taskar. 2007. Expectation maximization and posterior constraints. In *Advances in NIPS*.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of ACL*.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for english. In *Proceedings of NODALIDA*.

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proceedings of EMNLP*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*.

Mirella Lapata. 2006. Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics*, 32(4):471–484.

Dekang Lin and Patrick Pantel. 2001. DIRT - discovery of inference rules from text. In *Proceedings of SIGKDD*.

Gideon S. Mann and Andrew McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proceedings of ACL*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL/IJCNLP*.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of EMNLP*.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged texts. In *Proceedings of AAAI*.

Benjamin Rosenfeld and Ronen Feldman. 2007. Clustering for unsupervised relation identification. In *Proceedings of CIKM*.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of CoNLL*.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of HLT/NAACL*.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic IE pattern acquisition. In *Proceedings of ACL*.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of COLING*.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Cross-document relation extraction without labelled data. In *Proceedings of EMNLP*.

Alexander Yates and Oren Etzioni. 2009. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34:255–296.

Min Zhang, Jian Su, Danmei Wang, Guodong Zhou, and Chew Lim Tan. 2005. Discovering relations between named entities from a large raw corpus using tree similarity-based clustering. In *Proceedings of IJCNLP*.

Jun Zhu, Zaiqing Nie, Xiaojing Liu, Bo Zhang, and Ji-Rong Wen. 2009. StatSnowball: a statistical approach to extracting entity relationships. In *Proceedings of WWW*.

Knowledge-Based Weak Supervision for Information Extraction of Overlapping Relations

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, Daniel S. Weld
Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA
{raphaelh,clzhang,xiaoling,lsz,weld}@cs.washington.edu

Abstract

Information extraction (IE) holds the promise of generating a large-scale knowledge base from the Web's natural language text. Knowledge-based weak supervision, using structured data to heuristically label a training corpus, works towards this goal by enabling the automated learning of a potentially unbounded number of relation extractors. Recently, researchers have developed multi-instance learning algorithms to combat the noisy training data that can come from heuristic labeling, but their models assume relations are *disjoint* — for example they cannot extract the pair `Founded(Jobs, Apple)` and `CEO-of(Jobs, Apple)`.

This paper presents a novel approach for multi-instance learning with overlapping relations that combines a sentence-level extraction model with a simple, corpus-level component for aggregating the individual facts. We apply our model to learn extractors for NY Times text using weak supervision from Freebase. Experiments show that the approach runs quickly and yields surprising gains in accuracy, at both the aggregate and sentence level.

1 Introduction

Information-extraction (IE), the process of generating relational data from natural-language text, continues to gain attention. Many researchers dream of creating a large repository of high-quality extracted tuples, arguing that such a knowledge base could benefit many important tasks such as question answering and summarization. Most approaches to IE use supervised learning of relation-specific examples, which can achieve high precision and recall. Unfortunately, however, fully supervised methods are limited by the availability of training data and are unlikely to scale to the thousands of relations found on the Web.

A more promising approach, often called "weak" or "distant" supervision, creates its own training data by heuristically matching the contents of a database to corresponding text (Craven and Kumlien, 1999). For example, suppose that $r(e_1, e_2) =$ `Founded(Jobs,Apple)` is a ground tuple in the database and $s =$"Steve Jobs founded Apple, Inc." is a sentence containing synonyms for both $e_1 =$ `Jobs` and $e_2 =$ `Apple`, then s may be a natural language expression of the fact that $r(e_1, e_2)$ holds and could be a useful training example.

While weak supervision works well when the textual corpus is tightly aligned to the database contents (*e.g.*, matching Wikipedia infoboxes to associated articles (Hoffmann et al., 2010)), Riedel *et al.* (2010) observe that the heuristic leads to noisy data and poor extraction performance when the method is applied more broadly (*e.g.*, matching Freebase records to NY Times articles). To fix this problem they cast weak supervision as a form of multi-instance learning, assuming only that *at least one* of the sentences containing e_1 and e_2 are expressing $r(e_1, e_2)$, and their method yields a substantial improvement in extraction performance.

However, Riedel *et al.*'s model (like that of previous systems (Mintz et al., 2009)) assumes that *relations do not overlap* — there cannot exist two facts $r(e_1, e_2)$ and $q(e_1, e_2)$ that are both true for any pair of entities, e_1 and e_2. Unfortunately, this assumption is often violated;

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 541–550,
Portland, Oregon, June 19-24, 2011.

for example both `Founded(Jobs, Apple)` and `CEO-of(Jobs, Apple)` are clearly true. Indeed, 18.3% of the weak supervision facts in Freebase that match sentences in the NY Times 2007 corpus have overlapping relations.

This paper presents MULTIR, a novel model of weak supervision that makes the following contributions:

- MULTIR introduces a probabilistic, graphical model of multi-instance learning which handles overlapping relations.
- MULTIR also produces accurate sentence-level predictions, decoding individual sentences as well as making corpus-level extractions.
- MULTIR is computationally tractable. Inference reduces to weighted set cover, for which it uses a greedy approximation with worst case running time $O(|\mathrm{R}| \cdot |\mathrm{S}|)$ where R is the set of possible relations and S is largest set of sentences for any entity pair. In practice, MULTIR runs very quickly.
- We present experiments showing that MULTIR outperforms a reimplementation of Riedel *et al.* (2010)'s approach on both aggregate (corpus as a whole) and sentential extractions. Additional experiments characterize aspects of MULTIR's performance.

2 Weak Supervision from a Database

Given a corpus of text, we seek to extract facts about *entities*, such as the company `Apple` or the city `Boston`. A *ground fact* (or *relation instance*), is an expression $r(\mathbf{e})$ where r is a relation name, for example `Founded` or `CEO-of`, and $\mathbf{e} = e_1, \ldots, e_n$ is a list of entities.

An *entity mention* is a contiguous sequence of textual tokens denoting an entity. In this paper we assume that there is an *oracle* which can identify all entity mentions in a corpus, but the oracle doesn't normalize or disambiguate these mentions. We use $e_i \in E$ to denote both an entity and its name (*i.e.*, the tokens in its mention).

A *relation mention* is a sequence of text (including one or more entity mentions) which states that some ground fact $r(\mathbf{e})$ is true. For example, "Steve Ballmer, CEO of Microsoft, spoke recently at CES." contains three entity mentions as well as a relation mention for `CEO-of(Steve Ballmer, Microsoft)`. In this paper we restrict our attention to binary relations. Furthermore, we assume that both entity mentions appear as noun phrases in a single sentence.

The task of *aggregate extraction* takes two inputs, Σ, a set of sentences comprising the corpus, and an extraction model; as output it should produce a set of ground facts, I, such that each fact $r(\mathbf{e}) \in I$ is expressed somewhere in the corpus.

Sentential extraction takes the same input and likewise produces I, but in addition it also produces a function, $\Gamma : I \rightarrow \mathcal{P}(\Sigma)$, which identifies, for each $r(\mathbf{e}) \in I$, the set of sentences in Σ that contain a mention describing $r(\mathbf{e})$. In general, the corpus-level extraction problem is easier, since it need only make aggregate predictions, perhaps using corpus-wide statistics. In contrast, sentence-level extraction must justify each extraction with *every* sentence which expresses the fact.

The *knowledge-based weakly supervised learning* problem takes as input (1) Σ, a training corpus, (2) E, a set of entities mentioned in that corpus, (3) R, a set of relation names, and (4), Δ, a set of ground facts of relations in R. As output the learner produces an extraction model.

3 Modeling Overlapping Relations

We define an undirected graphical model that allows joint reasoning about aggregate (corpus-level) and sentence-level extraction decisions. Figure 1(a) shows the model in plate form.

3.1 Random Variables

There exists a connected component for each pair of entities $\mathbf{e} = (e_1, e_2) \in \mathrm{E} \times \mathrm{E}$ that models all of the extraction decisions for this pair. There is one Boolean output variable Y^r for each relation name $r \in \mathrm{R}$, which represents whether the ground fact $r(\mathbf{e})$ is true. Including this set of binary random variables enables our model to extract overlapping relations.

Let $S_{(e_1,e_2)} \subset \Sigma$ be the set of sentences which contain mentions of both of the entities. For each sentence $x_i \in S_{(e_1,e_2)}$ there exists a latent variable Z_i which ranges over the relation names $r \in R$ and,

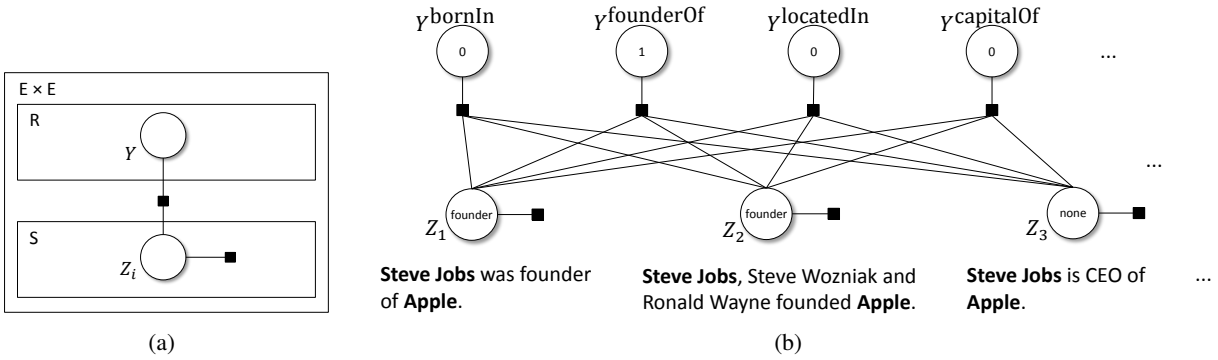


Figure 1: (a) Network structure depicted as plate model and (b) an example network instantiation for the pair of entities `Steve Jobs`, `Apple`.

importantly, also the distinct value `none`. Z_i should be assigned a value $r \in R$ only when x_i expresses the ground fact $r(\mathbf{e})$, thereby modeling sentence-level extraction.

Figure 1(b) shows an example instantiation of the model with four relation names and three sentences.

3.2 A Joint, Conditional Extraction Model

We use a conditional probability model that defines a joint distribution over all of the extraction random variables defined above. The model is undirected and includes repeated factors for making sentence level predictions as well as globals factors for aggregating these choices.

For each entity pair $\mathbf{e} = (e_1, e_2)$, define $\mathbf{x}$ to be a vector concatenating the individual sentences $x_i \in S_{(e_1,e_2)}$, $\mathbf{Y}$ to be vector of binary Y^r random variables, one for each $r \in R$, and $\mathbf{Z}$ to be the vector of Z_i variables, one for each sentence x_i. Our conditional extraction model is defined as follows:

$$p(\mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z}|\mathbf{x}; \theta) \stackrel{\text{def}}{=} \frac{1}{Z_x} \prod_r \Phi^{\text{join}}(y^r, \mathbf{z}) \prod_i \Phi^{\text{extract}}(z_i, x_i)$$

where the parameter vector θ is used, below, to define the factor Φ^{extract}.

The factors Φ^{join} are deterministic OR operators

$$\Phi^{\text{join}}(y^r, \mathbf{z}) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } y^r = true \wedge \exists i : z_i = r \\ 0 & \text{otherwise} \end{cases}$$

which are included to ensure that the ground fact $r(\mathbf{e})$ is predicted at the aggregate level for the assignment $Y^r = y^r$ only if at least one of the sentence level assignments $Z_i = z_i$ signals a mention of $r(\mathbf{e})$.

The extraction factors Φ^{extract} are given by

$$\Phi^{\text{extract}}(z_i, x_i) \stackrel{\text{def}}{=} \exp\left(\sum_j \theta_j \phi_j(z_i, x_i)\right)$$

where the features ϕ_j are sensitive to the relation name assigned to extraction variable z_i, if any, and cues from the sentence x_i. We will make use of the Mintz *et al.* (2009) sentence-level features in the expeiments, as described in Section 7.

3.3 Discussion

This model was designed to provide a joint approach where extraction decisions are almost entirely driven by sentence-level reasoning. However, defining the Y^r random variables and tying them to the sentence-level variables, Z_i, provides a direct method for modeling weak supervision. We can simply train the model so that the Y variables match the facts in the database, treating the Z_i as hidden variables that can take any value, as long as they produce the correct aggregate predictions.

This approach is related to the multi-instance learning approach of Riedel *et al.* (2010), in that both models include sentence-level and aggregate random variables. However, their sentence level variables are binary and they only have a single aggregate variable that takes values $r \in R \cup \{\texttt{none}\}$, thereby ruling out overlapping relations. Additionally, their aggregate decisions make use of Mintz-style aggregate features (Mintz et al., 2009), that collect evidence from multiple sentences, while we use

Inputs:
(1) Σ, a set of sentences,
(2) E, a set of entities mentioned in the sentences,
(3) R, a set of relation names, and
(4) Δ, a database of atomic facts of the form $r(e_1, e_2)$ for $r \in R$ and $e_i \in E$.

Definitions:
We define the training set $\{(\mathbf{x}_i, \mathbf{y}_i)|i = 1 \ldots n\}$, where i is an index corresponding to a particular entity pair (e_j, e_k) in Δ, $\mathbf{x}_i$ contains all of the sentences in Σ with mentions of this pair, and $\mathbf{y}_i =$ **relVector**(e_j, e_k).

Computation:
initialize parameter vector $\mathbf{\Theta} \leftarrow \mathbf{0}$
for $t = 1...T$ **do**
  for $i = 1...n$ **do**
    $(\mathbf{y}', \mathbf{z}') \leftarrow \arg\max_{\mathbf{y},\mathbf{z}} p(\mathbf{y}, \mathbf{z}|\mathbf{x_i}; \theta)$
    if $\mathbf{y}' \neq \mathbf{y_i}$ **then**
      $\mathbf{z}^* \leftarrow \arg\max_{\mathbf{z}} p(\mathbf{z}|\mathbf{x_i}, \mathbf{y_i}; \theta)$
      $\mathbf{\Theta} \leftarrow \mathbf{\Theta} + \phi(\mathbf{x_i}, \mathbf{z}^*) - \phi(\mathbf{x_i}, \mathbf{z}')$
    end if
  end for
end for
Return Θ

Figure 2: The MULTIR Learning Algorithm

only the deterministic OR nodes. Perhaps surprising, we are still able to improve performance at both the sentential and aggregate extraction tasks.

4 Learning

We now present a multi-instance learning algorithm for our weak-supervision model that treats the sentence-level extraction random variables Z_i as latent, and uses facts from a database (*e.g.*, Freebase) as supervision for the aggregate-level variables Y^r.

As input we have (1) Σ, a set of sentences, (2) E, a set of entities mentioned in the sentences, (3) R, a set of relation names, and (4) Δ, a database of atomic facts of the form $r(e_1, e_2)$ for $r \in R$ and $e_i \in E$. Since we are using weak learning, the Y^r variables in $\mathbf{Y}$ are not directly observed, but can be approximated from the database Δ. We use a procedure, **relVector**(e_1, e_2) to return a bit vector whose j^{th} bit is one if $r_j(e_1, e_2) \in \Delta$. The vector does *not* have a bit for the special *none* relation; if there is no relation between the two entities, all bits are zero.

Finally, we can now define the training set to be pairs $\{(\mathbf{x}_i, \mathbf{y}_i)|i = 1 \ldots n\}$, where i is an index corresponding to a particular entity pair (e_j, e_k), $\mathbf{x}_i$ contains all of the sentences with mentions of this pair, and $\mathbf{y}_i =$ **relVector**(e_j, e_k).

Given this form of supervision, we would like to find the setting for θ with the highest likelihood:

$$O(\theta) = \prod_i p(\mathbf{y}_i|\mathbf{x}_i; \theta) = \prod_i \sum_{\mathbf{z}} p(\mathbf{y}_i, \mathbf{z}|\mathbf{x}_i; \theta)$$

However, this objective would be difficult to optimize exactly, and algorithms for doing so would be unlikely to scale to data sets of the size we consider. Instead, we make two approximations, described below, leading to a Perceptron-style additive (Collins, 2002) parameter update scheme which has been modified to reason about hidden variables, similar in style to the approaches of (Liang et al., 2006; Zettlemoyer and Collins, 2007), but adapted for our specific model. This approximate algorithm is computationally efficient and, as we will see, works well in practice.

Our first modification is to do online learning instead of optimizing the full objective. Define the feature sums $\phi(\mathbf{x}, \mathbf{z}) = \sum_j \phi(x_j, z_j)$ which range over the sentences, as indexed by j. Now, we can define an update based on the gradient of the local log likelihood for example i:

$$\frac{\partial \log O_i(\theta)}{\partial \theta_j} = E_{p(\mathbf{z}|\mathbf{x}_i, \mathbf{y}_i; \theta)}[\phi_j(\mathbf{x}_i, \mathbf{z})] - E_{p(\mathbf{y}, \mathbf{z}|\mathbf{x}_i; \theta)}[\phi_j(\mathbf{x}_i, \mathbf{z})]$$

where the deterministic OR Φ^{join} factors ensure that the first expectation assigns positive probability only to assignments that produce the labeled facts $\mathbf{y}_i$ but that the second considers all valid sets of extractions.

Of course, these expectations themselves, especially the second one, would be difficult to compute exactly. Our second modification is to do a Viterbi approximation, by replacing the expectations with maximizations. Specifically, we compute the most likely sentence extractions for the label facts $\arg\max_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}_i, \mathbf{y}_i; \theta)$ and the most likely extraction for the input, without regard to the labels, $\arg\max_{\mathbf{y},\mathbf{z}} p(\mathbf{y}, \mathbf{z}|\mathbf{x}_i; \theta)$. We then compute the features for these assignments and do a simple additive update. The final algorithm is detailed in Figure 2.

5 Inference

To support learning, as described above, we need to compute assignments $\arg\max_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}, \mathbf{y}; \theta)$ and $\arg\max_{\mathbf{y},\mathbf{z}} p(\mathbf{y}, \mathbf{z}|\mathbf{x}; \theta)$. In this section, we describe algorithms for both cases that use the deterministic OR nodes to simplify the required computations.

Predicting the most likely joint extraction $\arg\max_{\mathbf{y},\mathbf{z}} p(\mathbf{y}, \mathbf{z}|\mathbf{x}; \theta)$ can be done efficiently given the structure of our model. In particular, we note that the factors Φ^{join} represent deterministic dependencies between $\mathbf{Z}$ and $\mathbf{Y}$, which when satisfied do not affect the probability of the solution. It is thus sufficient to independently compute an assignment for each sentence-level extraction variable Z_i, ignoring the deterministic dependencies. The optimal setting for the aggregate variables $\mathbf{Y}$ is then simply the assignment that is consistent with these extractions. The time complexity is $O(|\mathrm{R}| \cdot |\mathrm{S}|)$.

Predicting sentence level extractions given weak supervision facts, $\arg\max_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}, \mathbf{y}; \theta)$, is more challenging. We start by computing extraction scores $\Phi^{\text{extract}}(x_i, z_i)$ for each possible extraction assignment $Z_i = z_i$ at each sentence $x_i \in \mathrm{S}$, and storing the values in a dynamic programming table. Next, we must find the most likely assignment $\mathbf{z}$ that respects our output variables $\mathbf{y}$. It turns out that this problem is a variant of the weighted, edge-cover problem, for which there exist polynomial time optimal solutions.

Let $G = (\mathcal{E}, \mathcal{V} = \mathcal{V}^{\mathrm{S}} \cup \mathcal{V}^{\mathbf{y}})$ be a complete weighted bipartite graph with one node $v_i^{\mathrm{S}} \in \mathcal{V}^{\mathrm{S}}$ for each sentence $x_i \in \mathrm{S}$ and one node $v_r^{\mathbf{y}} \in \mathcal{V}^{\mathbf{y}}$ for each relation $r \in \mathrm{R}$ where $y^r = 1$. The edge weights are given by $c((v_i^{\mathrm{S}}, v_r^{\mathbf{y}})) \stackrel{\text{def}}{=} \Phi^{\text{extract}}(\mathbf{x}_i, z_i)$. Our goal is to select a subset of the edges which maximizes the sum of their weights, subject to each node $v_i^{\mathrm{S}} \in \mathcal{V}^{\mathrm{S}}$ being incident to exactly one edge, and each node $v_r^{\mathbf{y}} \in \mathcal{V}^{\mathbf{y}}$ being incident to at least one edge.

Exact Solution An exact solution can be obtained by first computing the maximum weighted bipartite matching, and adding edges to nodes which are not incident to an edge. This can be computed in time $O(|\mathcal{V}|(|\mathcal{E}| + |\mathcal{V}| \log |\mathcal{V}|))$, which we can rewrite as $O((|\mathrm{R}| + |\mathrm{S}|)(|\mathrm{R}||\mathrm{S}| + (|\mathrm{R}| + |\mathrm{S}|) \log(|\mathrm{R}| + |\mathrm{S}|)))$.

Approximate Solution An approximate solution can be obtained by iterating over the nodes in $\mathcal{V}^{\mathbf{y}}$, and each time adding the highest weight incident edge whose addition doesn't violate a constraint. The running time is $O(|\mathrm{R}||\mathrm{S}|)$. This greedy search guarantees each fact is extracted at least once and allows any additional extractions that increase the overall probability of the assignment. Given the computational advantage, we use it in all of the experimental evaluations.

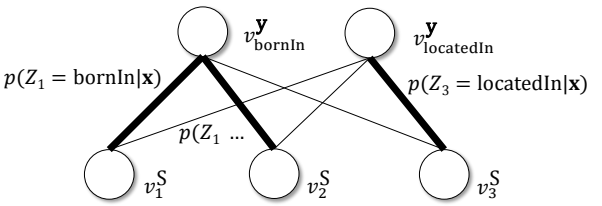


Figure 3: Inference of $\arg\max_{\mathbf{z}} p(\mathbf{Z} = \mathbf{z}|\mathbf{x}, \mathbf{y})$ requires solving a weighted, edge-cover problem.

6 Experimental Setup

We follow the approach of Riedel *et al.* (2010) for generating weak supervision data, computing features, and evaluating aggregate extraction. We also introduce new metrics for measuring sentential extraction performance, both relation-independent and relation-specific.

6.1 Data Generation

We used the same data sets as Riedel *et al.* (2010) for weak supervision. The data was first tagged with the Stanford NER system (Finkel et al., 2005) and then entity mentions were found by collecting each continuous phrase where words were tagged identically (*i.e.*, as a person, location, or organization). Finally, these phrases were matched to the names of Freebase entities.

Given the set of matches, define Σ to be set of NY Times sentences with two matched phrases, E to be the set of Freebase entities which were mentioned in one or more sentences, Δ to be the set of Freebase facts whose arguments, e_1 and e_2 were mentioned in a sentence in Σ, and R to be set of relations names used in the facts of Δ. These sets define the weak supervision data.

6.2 Features and Initialization

We use the set of sentence-level features described by Riedel *et al.* (2010), which were originally de-

veloped by Mintz *et al.* (2009). These include indicators for various lexical, part of speech, named entity, and dependency tree path properties of entity mentions in specific sentences, as computed with the Malt dependency parser (Nivre and Nilsson, 2004) and OpenNLP POS tagger[1]. However, unlike the previous work, we did not make use of any features that explicitly aggregate these properties across multiple mention instances.

The MULTIR algorithm has a single parameter T, the number of training iterations, that must be specified manually. We used $T = 50$ iterations, which performed best in development experiments.

6.3 Evaluation Metrics

Evaluation is challenging, since only a small percentage (approximately 3%) of sentences match facts in Freebase, and the number of matches is highly unbalanced across relations, as we will see in more detail later. We use the following metrics.

Aggregate Extraction Let Δ^e be the set of extracted relations for any of the systems; we compute aggregate precision and recall by comparing Δ^e with Δ. This metric is easily computed but underestimates extraction accuracy because Freebase is incomplete and some true relations in Δ^e will be marked wrong.

Sentential Extraction Let S^e be the sentences where some system extracted a relation and S^F be the sentences that match the arguments of a fact in Δ. We manually compute sentential extraction accuracy by sampling a set of 1000 sentences from $S^e \cup S^F$ and manually labeling the correct extraction decision, either a relation $r \in R$ or $none$. We then report precision and recall for each system on this set of sampled sentences. These results provide a good approximation to the true precision but can overestimate the actual recall, since we did not manually check the much larger set of sentences where no approach predicted extractions.

6.4 Precision / Recall Curves

To compute precision / recall curves for the tasks, we ranked the MULTIR extractions as follows. For sentence-level evaluations, we ordered according to the extraction factor score $\Phi^{\text{extract}}(z_i, x_i)$. For aggregate comparisons, we set the score for an extraction $Y^r = true$ to be the max of the extraction factor scores for the sentences where r was extracted.

[1] http://opennlp.sourceforge.net/

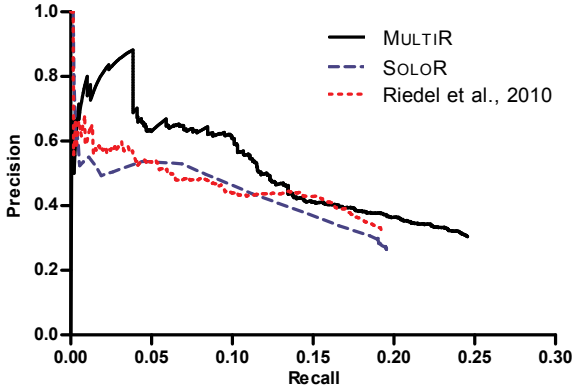


Figure 4: Aggregate extraction precision / recall curves for Riedel *et al.* (2010), a reimplementation of that approach (SOLOR), and our algorithm (MULTIR).

7 Experiments

To evaluate our algorithm, we first compare it to an existing approach for using multi-instance learning with weak supervision (Riedel et al., 2010), using the same data and features. We report both aggregate extraction and sentential extraction results. We then investigate relation-specific performance of our system. Finally, we report running time comparisons.

7.1 Aggregate Extraction

Figure 4 shows approximate precision / recall curves for three systems computed with aggregate metrics (Section 6.3) that test how closely the extractions match the facts in Freebase. The systems include the original results reported by Riedel *et al.* (2010) as well as our new model (MULTIR). We also compare with SOLOR, a reimplementation of their algorithm, which we built in Factorie (McCallum et al., 2009), and will use later to evaluate sentential extraction.

MULTIR achieves competitive or higher precision over all ranges of recall, with the exception of the very low recall range of approximately 0-1%. It also significantly extends the highest recall achieved, from 20% to 25%, with little loss in precision. To investigate the low precision in the 0-1% recall range, we manually checked the ten highest con-

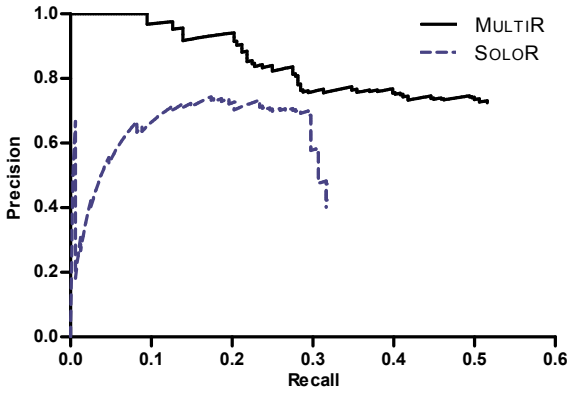


Figure 5: Sentential extraction precision / recall curves for MULTIR and SOLOR.

fidence extractions produced by MULTIR that were marked wrong. We found that all ten were true facts that were simply missing from Freebase. A manual evaluation, as we perform next for sentential extraction, would remove this dip.

7.2 Sentential Extraction

Although their model includes variables to model sentential extraction, Riedel *et al.* (2010) did not report sentence level performance. To generate the precision / recall curve we used the joint model assignment score for each of the sentences that contributed to the aggregate extraction decision.

Figure 4 shows approximate precision / recall curves for MULTIR and SOLOR computed against manually generated sentence labels, as defined in Section 6.3. MULTIR achieves significantly higher recall with a consistently high level of precision. At the highest recall point, MULTIR reaches 72.4% precision and 51.9% recall, for an F1 score of 60.5%.

7.3 Relation-Specific Performance

Since the data contains an unbalanced number of instances of each relation, we also report precision and recall for each of the ten most frequent relations. Let S_r^M be the sentences where MULTIR extracted an instance of relation $r \in R$, and let S_r^F be the sentences that match the arguments of a fact about relation r in Δ. For each r, we sample 100 sentences from both S_r^M and S_r^F and manually check accuracy. To estimate precision $\tilde{P}_r$ we compute the ratio of true relation mentions in S_r^M, and to estimate recall $\tilde{R}_r$ we take the ratio of true relation mentions in S_r^F which are returned by our system.

Table 1 presents this approximate precision and recall for MULTIR on each of the relations, along with statistics we computed to measure the quality of the weak supervision. Precision is high for the majority of relations but recall is consistently lower. We also see that the Freebase matches are highly skewed in quantity and can be low quality for some relations, with very few of them actually corresponding to true extractions. The approach generally performs best on the relations with a sufficiently large number of true matches, in many cases even achieving precision that outperforms the accuracy of the heuristic matches, at reasonable recall levels.

7.4 Overlapping Relations

Table 1 also highlights some of the effects of learning with overlapping relations. For example, in the data, almost all of the matches for the administrative_divisions relation overlap with the contains relation, because they both model relationships for a pair of locations. Since, in general, sentences are much more likely to describe a contains relation, this overlap leads to a situation were almost none of the administrate_division matches are true ones, and we cannot accurately learn an extractor. However, we can still learn to accurately extract the contains relation, despite the distracting matches. Similarly, the place_of_birth and place_of_death relations tend to overlap, since it is often the case that people are born and die in the same city. In both cases, the precision outperforms the labeling accuracy and the recall is relatively high.

To measure the impact of modeling overlapping relations, we also evaluated a simple, restricted baseline. Instead of labeling each entity pair with the set of all true Freebase facts, we created a dataset where each true relation was used to create a different training example. Training MULTIR on this data simulates effects of conflicting supervision that can come from not modeling overlaps. On average across relations, precision increases 12 points but recall drops 26 points, for an overall reduction in F1 score from 60.5% to 40.3%.

7.5 Running Time

One final advantage of our model is the modest running time. Our implementation of the

Relation	Freebase Matches		MULTIR	
	#sents	% true	$\tilde{P}$	$\tilde{R}$
/business/person/company	302	89.0	100.0	25.8
/people/person/place_lived	450	60.0	80.0	6.7
/location/location/contains	2793	51.0	100.0	56.0
/business/company/founders	95	48.4	71.4	10.9
/people/person/nationality	723	41.0	85.7	15.0
/location/neighborhood/neighborhood_of	68	39.7	100.0	11.1
/people/person/children	30	80.0	100.0	8.3
/people/deceased_person/place_of_death	68	22.1	100.0	20.0
/people/person/place_of_birth	162	12.0	100.0	33.0
/location/country/administrative_divisions	424	0.2	N/A	0.0

Table 1: Estimated precision and recall by relation, as well as the number of matched sentences (#sents) and accuracy (% true) of matches between sentences and facts in Freebase.

Riedel *et al.* (2010) approach required approximately 6 hours to train on NY Times 05-06 and 4 hours to test on the NY Times 07, each without preprocessing. Although they do sampling for inference, the global aggregation variables require reasoning about an exponentially large (in the number of sentences) sample space.

In contrast, our approach required approximately one minute to train and less than one second to test, on the same data. This advantage comes from the decomposition that is possible with the deterministic OR aggregation variables. For test, we simply consider each sentence in isolation and during training our approximation to the weighted assignment problem is linear in the number of sentences.

7.6 Discussion

The sentential extraction results demonstrates the advantages of learning a model that is primarily driven by sentence-level features. Although previous approaches have used more sophisticated features for aggregating the evidence from individual sentences, we demonstrate that aggregating strong sentence-level evidence with a simple deterministic OR that models overlapping relations is more effective, and also enables training of a sentence extractor that runs with no aggregate information.

While the Riedel *et al.* approach does include a model of which sentences express relations, it makes significant use of aggregate features that are primarily designed to do entity-level relation predictions and has a less detailed model of extractions at the individual sentence level. Perhaps surprisingly, our model is able to do better at both the sentential and aggregate levels.

8 Related Work

Supervised-learning approaches to IE were introduced in (Soderland et al., 1995) and are too numerous to summarize here. While they offer high precision and recall, these methods are unlikely to scale to the thousands of relations found in text on the Web. Open IE systems, which perform self-supervised learning of relation-independent extractors (*e.g.*, Preemptive IE (Shinyama and Sekine, 2006), TEXTRUNNER (Banko et al., 2007; Banko and Etzioni, 2008) and WOE (Wu and Weld, 2010)) can scale to millions of documents, but don't output canonicalized relations.

8.1 Weak Supervision

Weak supervision (also known as distant- or self supervision) refers to a broad class of methods, but we focus on the increasingly-popular idea of using a store of structured data to heuristicaly label a textual corpus. Craven and Kumlien (1999) introduced the idea by matching the Yeast Protein Database (YPD) to the abstracts of papers in PubMed and training a naive-Bayes extractor. Bellare and McCallum (2007) used a database of BibTex records to train a CRF extractor on 12 bibliographic relations. The KYLIN system aplied weak supervision to learn relations from Wikipedia, treating infoboxes as the associated database (Wu and Weld, 2007); Wu *et al.* (2008) extended the system to use smoothing over an automatically generated infobox taxon-

omy. Mintz *et al.* (2009) used Freebase facts to train 100 relational extractors on Wikipedia. Hoffmann *et al.* (2010) describe a system similar to KYLIN, but which dynamically generates lexicons in order to handle sparse data, learning over 5000 Infobox relations with an average F1 score of 61%. Yao *et al.* (2010) perform weak supervision, while using selectional preference constraints to a jointly reason about entity types.

The NELL system (Carlson et al., 2010) can also be viewed as performing weak supervision. Its initial knowledge consists of a selectional preference constraint and 20 ground fact seeds. NELL then matches entity pairs from the seeds to a Web corpus, but instead of learning a probabilistic model, it bootstraps a set of extraction patterns using semi-supervised methods for multitask learning.

8.2 Multi-Instance Learning

Multi-instance learning was introduced in order to combat the problem of ambiguously-labeled training data when predicting the activity of different drugs (Dietterich et al., 1997). Bunescu and Mooney (2007) connect weak supervision with multi-instance learning and extend their relational extraction kernel to this context.

Riedel *et al.* (2010), combine weak supervision and multi-instance learning in a more sophisticated manner, training a graphical model, which assumes only that *at least one* of the matches between the arguments of a Freebase fact and sentences in the corpus is a true relational mention. Our model may be seen as an extension of theirs, since both models include sentence-level and aggregate random variables. However, Riedel *et al.* have only a single aggregate variable that takes values $r \in R \cup \{\texttt{none}\}$, thereby ruling out overlapping relations. We have discussed the comparison in more detail throughout the paper, including in the model formulation section and experiments.

9 Conclusion

We argue that weak supervision is promising method for scaling information extraction to the level where it can handle the myriad, different relations on the Web. By using the contents of a database to heuristically label a training corpus, we may be able to automatically learn a nearly unbounded number of relational extractors. Since the processs of matching database tuples to sentences is inherently heuristic, researchers have proposed multi-instance learning algorithms as a means for coping with the resulting noisy data. Unfortunately, previous approaches assume that all relations are *disjoint* — for example they cannot extract the pair `Founded(Jobs, Apple)` and `CEO-of(Jobs, Apple)`, because two relations are not allowed to have the same arguments.

This paper presents a novel approach for multi-instance learning with overlapping relations that combines a sentence-level extraction model with a simple, corpus-level component for aggregating the individual facts. We apply our model to learn extractors for NY Times text using weak supervision from Freebase. Experiments show improvements for both sentential and aggregate (corpus level) extraction, and demonstrate that the approach is computationally efficient.

Our early progress suggests many interesting directions. By joining two or more Freebase tables, we can generate many more matches and learn more relations. We also wish to refine our model in order to improve precision. For example, we would like to add type reasoning about entities and selectional preference constraints for relations. Finally, we are also interested in applying the overall learning approaches to other tasks that could be modeled with weak supervision, such as coreference and named entity classification.

The source code of our system, its output, and all data annotations are available at `http://cs.uw.edu/homes/raphaelh/mr`.

Acknowledgments

We thank Sebastian Riedel and Limin Yao for sharing their data and providing valuable advice. This material is based upon work supported by a WRF / TJ Cable Professorship, a gift from Google and by the Air Force Research Laboratory (AFRL) under prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the Air Force Research Laboratory (AFRL).

References

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Proceedings of 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pages 28–36.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2670–2676.

Kedar Bellare and Andrew McCallum. 2007. Learning extractors from unlabeled text using relevant databases. In *Sixth International Workshop on Information Integration on the Web*.

Razvan Bunescu and Raymond Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-10)*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-2002)*.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 77–86.

Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. 1997. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, January.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pages 363–370.

Raphael Hoffmann, Congle Zhang, and Daniel S. Weld. 2010. Learning 5000 relational extractors. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, pages 286–295.

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

Andrew McCallum, Karl Schultz, and Sameer Singh. 2009. Factorie: Probabilistic programming via imperatively defined factor graphs. In *Neural Information Processing Systems Conference (NIPS)*.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-2009)*, pages 1003–1011.

Joakim Nivre and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of the Conference on Natural Language Learning (CoNLL-04)*, pages 49–56.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of the Sixteenth European Conference on Machine Learning (ECML-2010)*, pages 148–163.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computation Linguistics (HLT-NAACL-06)*.

Stephen Soderland, David Fisher, Jonathan Aseltine, and Wendy G. Lehnert. 1995. Crystal: Inducing a conceptual dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-1995)*, pages 1314–1321.

Fei Wu and Daniel S. Weld. 2007. Autonomously semantifying wikipedia. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM-2007)*, pages 41–50.

Fei Wu and Daniel S. Weld. 2008. Automatically refining the wikipedia infobox ontology. In *Proceedings of the 17th International Conference on World Wide Web (WWW-2008)*, pages 635–644.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using wikipedia. In *The Annual Meeting of the Association for Computational Linguistics (ACL-2010)*, pages 118–127.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Collective cross-document relation extraction without labelled data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-2010)*, pages 1013–1023.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-2007)*.

Exploiting Syntactico-Semantic Structures for Relation Extraction

Yee Seng Chan and **Dan Roth**
University of Illinois at Urbana-Champaign
{chanys,danr}@illinois.edu

Abstract

In this paper, we observe that there exists a second dimension to the relation extraction (RE) problem that is orthogonal to the relation type dimension. We show that most of these second dimensional structures are relatively constrained and not difficult to identify. We propose a novel algorithmic approach to RE that starts by first identifying these structures and then, within these, identifying the semantic type of the relation. In the real RE problem where relation arguments need to be identified, exploiting these structures also allows reducing pipelined propagated errors. We show that this RE framework provides significant improvement in RE performance.

1 Introduction

Relation extraction (RE) has been defined as the task of identifying a given set of semantic binary relations in text. For instance, given the span of text "... the Seattle zoo ...", one would like to extract the relation that "the Seattle zoo" is located-at "Seattle". RE has been frequently studied over the last few years as a supervised learning task, learning from spans of text that are annotated with a set of semantic relations of interest. However, most approaches to RE have assumed that the relations' arguments are given as input (Chan and Roth, 2010; Jiang and Zhai, 2007; Jiang, 2009; Zhou et al., 2005), and therefore offer only a partial solution to the problem.

Conceptually, this is a rather simple approach as *all* spans of texts are treated uniformly and are being mapped to one of several relation types of interest. However, these approaches to RE require a large amount of manually annotated training data to achieve good performance, making it difficult to expand the set of target relations. Moreover, as we show, these approaches become brittle when the relations' arguments are not given but rather need to be identified in the data too.

In this paper we build on the observation that there exists a second dimension to the relation extraction problem that is orthogonal to the *relation type* dimension: all relation types are expressed in one of several constrained syntactico-semantic structures. As we show, identifying where the text span is on the *syntactico-semantic structure* dimension first, can be leveraged in the RE process to yield improved performance. Moreover, working in the second dimension provides robustness to the *real* RE problem, that of identifying arguments along with the relations between them.

For example, in "the Seattle zoo", the entity mention "Seattle" modifies the noun "zoo". Thus, the two mentions "Seattle" and "the Seattle zoo", are involved in what we later call a *premodifier relation*, one of several syntactico-semantic structures we identify in Section 3.

We highlight that all relation types can be expressed in one of several syntactico-semantic structures – Premodifiers, Possessive, Preposition, Formulaic and Verbal. As it turns out, most of these structures are relatively constrained and are not difficult to identify. This suggests a novel algorithmic approach to RE that starts by first identifying these structures and then, within these, identifying the semantic type of the relation. Not only does this approach provide significantly improved RE perfor-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 551–560,
Portland, Oregon, June 19-24, 2011.

mance, it carries with it two additional advantages.

First, leveraging the syntactico-semantic structure is especially beneficial in the presence of small amounts of data. Second, and more important, is the fact that exploiting the syntactico-semantic dimension provides several new options for dealing with the *full* RE problem – incorporating the argument identification into the problem. We explore one of these possibilities, making use of the constrained structures as a way to aid in the identification of the relations' arguments. We show that this already provides significant gain, and discuss other possibilities that can be explored. The contributions of this paper are summarized below:

- We highlight that all relation types are expressed as one of several syntactico-semantic structures and show that most of these are relatively constrained and not difficult to identify. Consequently, working first in this structural dimension can be leveraged in the RE process to improve performance.
- We show that when one does not have a large number of training examples, exploiting the syntactico-semantic structures is crucial for RE performance.
- We show how to leverage these constrained structures to improve RE when the relations' arguments are not given. The constrained structures allow us to jointly entertain argument candidates and relations built with them as arguments. Specifically, we show that considering argument candidates which otherwise would have been discarded (provided they exist in syntactico-semantic structures), we reduce error propagation along a standard pipeline RE architecture, and that this joint inference process leads to improved RE performance.

In the next section, we describe our relation extraction framework that leverages the syntactico-semantic structures. We then present these structures in Section 3. We describe our mention entity typing system in Section 4 and features for the RE system in Section 5. We present our RE experiments in Section 6 and perform analysis in Section 7, before concluding in Section 8.

$\mathcal{S}$ = {premodifier, possessive, preposition, formulaic}
gold mentions in training data $\mathcal{M}_{train}$
$\mathcal{D}_g = \{(m_i, m_j) \in \mathcal{M}_{train} \times \mathcal{M}_{train} \mid$
  m_i in same sentence as $m_j \wedge i \neq j \wedge i < j\}$
RE_{base} = RE classifier trained on $\mathcal{D}_g$

$D_s = \emptyset$
for each $(m_i, m_j) \in \mathcal{D}_g$
do
  p = structure inference on (m_i, m_j) using patterns
  if $p \in \mathcal{S} \vee (m_i, m_j)$ was annotated with a $\mathcal{S}$ structure
    $\mathcal{D}_s = \mathcal{D}_s \cup (m_i, m_j)$
done
RE_s = RE classifier trained on $\mathcal{D}_s$

Output: RE_{base} and RE_s

Figure 1: Training a regular baseline RE classifier RE_{base} and a RE classifier leveraging syntactico-semantic structures RE_s.

2 Relation Extraction Framework

In Figure 1, we show the algorithm for training a typical baseline RE classifier (RE_{base}), and for training a RE classifier that leverages the syntactico-semantic structures (RE_s).

During evaluation and when the *gold* mentions are already annotated, we apply RE_s as follows. When given a test example mention pair (x_i,x_j), we perform structure inference on it using the patterns described in Section 3. If (x_i,x_j) is identified as having any of the four syntactico-semantic structures $\mathcal{S}$, apply RE_s to predict the relation label, else apply RE_{base}.

Next, we show in Figure 2 our joint inference algorithmic framework that leverages the syntactico-semantic structures for RE, when mentions need to be *predicted*. Since the structures are fairly constrained, we can use them to consider mention candidates that are originally predicted as non mentions. As shown in Figure 2, we conservatively include such mentions when forming mention pairs, provided their null labels are predicted with a low probability t[1].

[1] In this work, we arbitrary set t=0.2. After the experiments, and in our own analysis, we observe that t=0.25 achieves better performance. Besides using the probability of the 1-best prediction, one could also for instance, use the probability difference between the first and second best predictions. However, selecting an optimal t value is not the main focus of this work.

$\mathcal{S}$ = {premodifier, possessive, preposition, formulaic}
candidate mentions $\mathcal{M}_{cand}$
Let $L_m = \underset{y}{\text{argmax}} P_{MET}(y|m,\theta), m \in \mathcal{M}_{cand}$
selected mentions $\mathcal{M}_{sel} = \{m \in \mathcal{M}_{cand} \mid$
$L_m \neq null \vee P_{MET}(null|m,\theta) \leq t\}$
$\mathcal{Q}_{hasNull} = \{(m_i, m_j) \in \mathcal{M}_{sel} \times \mathcal{M}_{sel} \mid$
m_i in same sentence as $m_j \wedge i \neq j \wedge i < j \wedge$
$(L_{m_i} \neq null \vee L_{m_j} \neq null)\}$
Let pool of relation predictions $\mathcal{R} = \emptyset$

for each $(m_i, m_j) \in \mathcal{Q}_{hasNull}$
do
p = structure inference on (m_i, m_j) using patterns
if $p \in \mathcal{S}$
r = relation prediction for (m_i, m_j) using RE_s
$\mathcal{R} = \mathcal{R} \cup r$
else if $L_{m_i} \neq null \wedge L_{m_j} \neq null$
r = relation prediction for (m_i, m_j) using RE_{base}
$\mathcal{R} = \mathcal{R} \cup r$
done

Output: $\mathcal{R}$

Figure 2: RE using predicted mentions and patterns. Abbreviations: L_m: predicted entity label for mention m using the mention entity typing (MET) classifier described in Section 4; P_{MET}: prediction probability according to the MET classifier; t: used for thresholding.

There is a large body of work in using patterns to extract relations (Fundel et al., 2007; Greenwood and Stevenson, 2006; Zhu et al., 2009). However, these works operate along the first dimension, that of using patterns to mine for *relation type* examples. In contrast, in our RE framework, we apply patterns to identify the syntactico-semantic structure dimension first, and leverage this in the RE process. In (Roth and Yih, 2007), the authors used entity types to constrain the (first dimensional) relation types allowed among them. In our work, although a few of our patterns involve semantic type comparison, most of the patterns are syntactic in nature.

In this work, we performed RE evaluation on the NIST Automatic Content Extraction (ACE) corpus. Most prior RE evaluation on ACE data assumed that mentions are already pre-annotated and given as input (Chan and Roth, 2010; Jiang and Zhai, 2007; Zhou et al., 2005). An exception is the work of (Kambhatla, 2004), where the author evaluated on the ACE-2003 corpus. In that work, the author did not address the pipelined errors propagated from the mention identification process.

3 Syntactico-Semantic Structures

In this paper, we performed RE on the ACE-2004 corpus. In ACE-2004 when the annotators tagged a pair of mentions with a relation, they also specified the type of syntactico-semantic structure[2]. ACE-2004 identified five types of structures: premodifier, possessive, preposition, formulaic, and verbal. We are unaware of any previous computational approaches that recognize these structures automatically in text, as we do, and use it in the context of RE (or any other problem). In (Qian et al., 2008), the authors reported the recall scores of their RE system on the various syntactico-semantic structures. But they do not attempt to recognize nor leverage these structures.

In this work, we focus on detecting the first four structures. These four structures cover 80% of the mention pairs having valid semantic relations (we give the detailed breakdown in Section 7) and we show that they are relatively easy to identify using simple rules or patterns. In this section, we indicate mentions using square bracket pairs, and use m_i and m_j to represent a mention pair. We now describe the four structures.

Premodifier relations specify the proper adjective or proper noun premodifier and the following noun it modifies, e.g.: [the [Seattle] zoo]

Possessive indicates that the first mention is in a possessive case, e.g.: [[California] 's Governor]

Preposition indicates that the two mentions are semantically related via the existence of a preposition, e.g.: [officials] in [California]

Formulaic The ACE04 annotation guideline[3] indicates the annotation of several formulaic relations, including for example address: [Medford] , [Massachusetts]

[2] ACE-2004 termed it as lexical condition. We use the term syntactico-semantic structure in this paper as the mention pair exists in specific syntactic structures, and we use rules or patterns that are syntactically and semantically motivated to detect these structures.

[3] http://projects.ldc.upenn.edu/ace/docs/EnglishRDCV4-3-2.PDF

Structure type	Pattern
Premodifier	Basic pattern: [u* [v+] w+] , where u, v, w represent words Each w is a noun or adjective If u* is not empty, then u*: JJ+ ∨ JJ "and" JJ? ∨ CD JJ* ∨ RB DT JJ? ∨ RB CD JJ ∨ DT (RB|JJ|VBG|VBD|VBN|CD)? Let w_1 = first word in w+. $w_1 \neq$ "'s" and POS tag of $w_1 \neq$ POS Let v_l = last word in v+. POS tag of $v_l \neq$ PRP$ nor WP$
Possessive	Basic pattern: [u? [v+] w+] , where u, v, w represent words Let w_1 = first word in w+. If w_1 = "'s" ∨ POS tag of w_1 = POS, accept mention pair Let v_l = last word in v+. If POS tag of v_l = PRP$ or WP$, accept mention pair
Preposition	Basic pattern: [m_i] v* [m_j], where v represent words and number of prepositions in the text span v* between them = 0, 1, or 2 If satisfy pattern: IN [m_i][m_j], accept mention pair If satisfy pattern: [m_i] (IN|TO) [m_j], accept mention pair If all labels in $\mathcal{L}_d$ start with "prep", accept mention pair
Formulaic	If satisfy pattern: [m_i] / [m_j] ∧ $E_c(m_i)$ = PER ∧ $E_c(m_j)$ = ORG, accept mention pair If satisfy pattern: [m_i][m_j] If $E_c(m_i)$ = PER ∧ $E_c(m_j)$ = ORG ∨ GPE, accept mention pair

Table 1: Rules and patterns for the four syntactico-semantic structures. Regular expression notations: '*' matches the preceding element zero or more times; '+' matches the preceding element one or more times; '?' indicates that the preceding element is optional; '|' indicates or. Abbreviations: $E_c(m)$: coarse-grained entity type of mention m; $\mathcal{L}_d$: labels in dependency path between the headword of two mentions. We use square brackets '[' and ']' to denote mention boundaries. The '/' in the *Formulaic* row denotes the occurrence of a lexical '/' in text.

In this rest of this section, we present the rules/patterns for detecting the above four syntactico-semantic structure, giving an overview of them in Table 1. We plan to release all of the rules/patterns along with associated code[4]. Notice that the patterns are intuitive and mostly syntactic in nature.

3.1 Premodifier Structures

- We require that one of the mentions completely include the other mention. Thus, the basic pattern is [u* [v+] w+].
- If u* is not empty, we require that it satisfies any of the following POS tag sequences: JJ+ ∨ JJ and JJ? ∨ CD JJ*, etc. These are (optional) POS tag sequences that normally start a valid noun phrase.
- We use two patterns to differentiate between premodifier relations and possessive relations, by checking for the existence of POS tags PRP$, WP$, POS, and the word "'s".

[4]http://cogcomp.cs.illinois.edu/page/publications

3.2 Possessive Structures

- The basic pattern for possessive is similar to that for premodifier: [u? [v+] w+]
- If the word immediately following v+ is "'s" or its POS tag is "POS", we accept the mention pair. If the POS tag of the last word in v+ is either PRP$ or WP$, we accept the mention pair.

3.3 Preposition Structures

- We first require the two mentions to be non-overlapping, and check for the existence of patterns such as "IN [m_i] [m_j]" and "[m_i] (IN|TO) [m_j]".
- If the only dependency labels in the dependency path between the head words of m_i and m_j are "prep" (prepositional modifier), accept the mention pair.

3.4 Formulaic Structures

- The ACE-2004 annotator guidelines specify that several relations such as reporter signing off, addresses, etc. are often specified in standard structures. We check for the existence of patterns such as "[m_i] / [m_j]", "[m_i] [m_j]",

Category	Feature
For every word w_k in mention m_i	POS of w_k and offset from lw
	w_k and offset from lw
	POS of w_k, w_k, and offset from lw
	POS of w_k, offset from lw, and lw
	$Bc(w_k)$ and offset from lw
	POS of w_k, $Bc(w_k)$, and offset from lw
	POS of w_k, offset from lw, and $Bc(lw)$
Contextual	$C_{-1,-1}$ of m_i
	$C_{+1,+1}$ of m_i
	$P_{-1,-1}$ of m_i
	$P_{+1,+1}$ of m_i
NE tags	tag of NE, if lw of NE coincides with lw of m_i in the sentence
Syntactic parse	parse-label of parse tree constituent that exactly covers m_i
	parse-labels of parse tree constituents covering m_i

Table 2: Features used in our mention entity typing (MET) system. The abbreviations are as follows. lw: last word in the mention; $Bc(w)$: the brown cluster bit string representing w; NE: named entity

and whether they satisfy certain semantic entity type constraints.

4 Mention Extraction System

As part of our experiments, we perform RE using predicted mentions. We first describe the features (an overview is given in Table 2) and then describe how we extract candidate mentions from sentences during evaluation.

4.1 Mention Extraction Features

Features for every word in the mention For every word w_k in a mention m_i, we extract seven features. These are a combination of w_k itself, its POS tag, and its integer offset from the last word (lw) in the mention. For instance, given the mention "the operation room", the offsets for the three words in the mention are -2, -1, and 0 respectively. These features are meant to capture the word and POS tag sequences in mentions.

We also use word clusters which are automatically generated from unlabeled texts, using the Brown clustering (Bc) algorithm of (Brown et al., 1992). This algorithm outputs a binary tree where words are leaves in the tree. Each word (leaf) in the tree can be represented by its unique path from the root and this path can be represented as a simple bit string. As part of our features, we use the cluster bit string representation of w_k and lw.

Category	Feature
POS	POS of single word between m_1, m_2
	hw of m_i, m_j and $P_{-1,-1}$ of m_i, m_j
	hw of m_i, m_j and $P_{-1,-1}$ of m_i, m_j
	hw of m_i, m_j and $P_{+1,+1}$ of m_i, m_j
	hw of m_i, m_j and $P_{-2,-1}$ of m_i, m_j
	hw of m_i, m_j and $P_{-1,+1}$ of m_i, m_j
	hw of m_i, m_j and $P_{+1,+2}$ of m_i, m_j
Base chunk	any base phrase chunk between m_i, m_j

Table 3: Additional RE features.

Contextual We extract the word $C_{-1,-1}$ immediately before m_i, the word $C_{+1,+1}$ immediately after m_i, and their associated POS tags P.

NE tags We automatically annotate the sentences with named entity (NE) tags using the named entity tagger of (Ratinov and Roth, 2009). This tagger annotates *proper nouns* with the tags PER (person), ORG (organization), LOC (location), or MISC (miscellaneous). If the lw of m_i coincides (actual token offset) with the lw of any NE annotated by the NE tagger, we extract the NE tag as a feature.

Syntactic parse We parse the sentences using the syntactic parser of (Klein and Manning, 2003). We extract the label of the parse tree constituent (if it exists) that *exactly covers* the mention, and also labels of all constituents that *covers* the mention.

4.2 Extracting Candidate Mentions

From a sentence, we gather the following as candidate mentions: all nouns and possessive pronouns, all named entities annotated by the the NE tagger (Ratinov and Roth, 2009), all base noun phrase (NP) chunks, all chunks satisfying the pattern: NP (PP NP)+, all NP constituents in the syntactic parse tree, and from each of these constituents, all substrings consisting of two or more words, provided the substrings do not start nor end on punctuation marks. These mention candidates are then fed to our mention entity typing (MET) classifier for type prediction (more details in Section 6.3).

5 Relation Extraction System

We build a supervised RE system using sentences annotated with entity mentions and predefined target relations. During evaluation, when given a pair of mentions m_i, m_j, the system predicts whether any of the predefined target relation holds between the mention pair.

Most of our features are based on the work of (Zhou et al., 2005; Chan and Roth, 2010). Due to space limitations, we refer the reader to our prior work (Chan and Roth, 2010) for the lexical, structural, mention-level, entity type, and dependency features. Here, we only describe the features that were not used in that work.

As part of our RE system, we need to extract the head word (hw) of a mention (m), which we heuristically determine as follows: if m contains a preposition and a noun preceding the preposition, we use the noun as the hw. If there is no preposition in m, we use the last noun in m as the hw.

POS features If there is a single word between the two mentions, we extract its POS tag. Given the hw of m, $P_{i,j}$ refers to the sequence of POS tags in the immediate context of hw (we exclude the POS tag of hw). The offsets *i* and *j* denote the position (relative to hw) of the first and last POS tag respectively. For instance, $P_{-2,-1}$ denotes the sequence of two POS tags on the immediate left of hw, and $P_{-1,+1}$ denotes the POS tag on the immediate left of hw and the POS tag on the immediate right of hw.

Base phrase chunk We add a boolean feature to detect whether there is any base phrase chunk in the text span between the two mentions.

6 Experiments

We use the ACE-2004 dataset (catalog LDC2005T09 from the Linguistic Data Consortium) to conduct our experiments. Following prior work, we use the news wire (nwire) and broadcast news (bnews) corpora of ACE-2004 for our experiments, which consists of 345 documents.

To build our RE system, we use the LIBLINEAR (Fan et al., 2008) package, with its default settings of L2-loss SVM (dual) as the solver, and we use an epsilon of 0.1. To ensure that this baseline RE system based on the features in Section 5 is competitive, we compare against the state-of-the-art feature-based RE systems of (Jiang and Zhai, 2007) and (Chan and Roth, 2010). In these works, the authors reported performance on undirected coarse-grained RE. Performing 5-fold cross validation on the nwire and bnews corpora, (Jiang and Zhai, 2007) and (Chan and Roth, 2010) reported F-measures of 71.5 and 71.2, respectively. Using the same evaluation setting, our baseline RE system achieves a competitive 71.4 F-measure.

We build three RE classifiers: *binary*, *coarse*, *fine*. Lumping all the predefined target relations into a single label, we build a *binary* classifier to predict whether any of the predefined relations exists between a given mention pair.

In this work, we model the argument order of the mentions when performing RE, since relations are usually asymmetric in nature. For instance, we consider m_i:EMP-ORG:m_j and m_j:EMP-ORG:m_i to be distinct relation types. In our experiments, we extracted a total of 55,520 examples or mention pairs. Out of these, 4,011 are positive relation examples annotated with 6 coarse-grained relation types and 22 fine-grained relation types[5].

We build a *coarse-grained* classifier to disambiguate between 13 relation labels (two asymmetric labels for each of the 6 coarse-grained relation types and a null label). We similarly build a *fine-grained* classifier to disambiguate between 45 relation labels.

6.1 Evaluation Method

For our experiments, we adopt the experimental setting in our prior work (Chan and Roth, 2010) of ensuring that all examples from a single document are either all used for training, or all used for evaluation.

In that work, we also highlight that ACE annotators rarely duplicate a relation link for coreferent mentions. For instance, assume mentions m_i, m_j, and m_k are in the same sentence, mentions m_i and m_j are coreferent, and the annotators tag the mention pair m_j, m_k with a particular relation r. The annotators will rarely duplicate the same (implicit)

[5]We omit a single relation: Discourse (DISC). The ACE-2004 annotation guidelines states that the DISC relation is established only for the purposes of the discourse and does not reference an official entity relevant to world knowledge. In this work, we focus on semantically meaningful relations. Furthermore, the DISC relation is dropped in ACE-2005.

RE model	10 documents			5% of data			80% of data		
	Rec%	Pre%	F1%	Rec%	Pre%	F1%	Rec%	Pre%	F1%
Binary	58.0	80.3	67.4	64.4	80.6	71.6	73.2	84.0	78.2
Binary+Patterns	73.1	78.5	**75.7** (+8.3)	75.3	80.6	**77.9**	80.1	84.2	**82.1**
Coarse	33.5	62.5	43.6	42.4	66.2	51.7	62.1	75.5	68.1
Coarse+Patterns	44.2	59.6	**50.8** (+7.2)	51.2	64.2	**56.9**	68.0	75.4	**71.5**
Fine	18.1	47.0	26.1	26.3	51.6	34.9	51.6	68.4	58.8
Fine+Patterns	24.8	43.5	**31.6** (+5.5)	32.2	48.9	**38.9**	56.4	67.5	**61.5**

Table 4: Micro-averaged (across the 5 folds) RE results using gold mentions.

RE model	10 documents			5% of data			80% of data		
	Rec%	Pre%	F1%	Rec%	Pre%	F1%	Rec%	Pre%	F1%
Binary	32.2	46.6	38.1	35.5	48.9	41.1	40.1	52.7	45.5
Binary+Patterns	46.7	45.9	**46.3** (+8.2)	47.6	47.8	**47.2**	50.2	50.4	**50.3**
Coarse	18.6	41.1	25.6	22.4	40.9	28.9	32.3	47.5	38.5
Coarse+Patterns	26.8	34.7	**30.2** (+4.6)	30.3	37.0	**33.3**	38.9	42.9	**40.8**
Fine	10.7	32.2	16.1	14.6	33.4	20.3	26.9	44.3	33.5
Fine+Patterns	15.7	26.3	**19.7** (+3.6)	19.4	29.2	**23.3**	31.7	38.3	**34.7**

Table 5: Micro-averaged (across the 5 folds) RE results using predicted mentions.

relation r between m_i and m_k, thus leaving the gold relation label as null. Whether this is correct or not is debatable. However, to avoid being penalized when our RE system actually correctly predicts the label of an implicit relation, we take the following approach.

During evaluation, if our system correctly predicts an implicit label, we simply switch its prediction to the null label. Since the RE recall scores only take into account non-null relation labels, this scoring method does not change the recall, but could marginally increase the precision scores by decreasing the count of RE predictions. In our experiments, we observe that both the usual and our scoring method give very similar RE results and the experimental trends remain the same. Of course, using this scoring method requires coreference information, which is available in the ACE data.

6.2 RE Evaluation Using Gold Mentions

To perform our experiments, we split the 345 documents into 5 equal sets. In each of the 5 folds, 4 sets (276 documents) are reserved for drawing training examples, while the remaining set (69 documents) is used as evaluation data. In the experiments described in this section, we use the gold mentions available in the data.

When one only has a small amount of training data, it is crucial to take advantage of external knowledge such as the syntactico-semantic structures. To simulate this setting, in each fold, we randomly selected 10 documents from the fold's available training documents (about 3% of the total 345 documents) as training data. We built one binary, one coarse-grained, and one fine-grained classifier for each fold.

In Section 2, we described how we trained a baseline RE classifier (RE_{base}) and a RE classifier using the syntactico-semantic patterns (RE_s).

We first apply RE_{base} on each test example mention pair (m_i,m_j) to obtain the RE baseline results, showing these in Table 4 under the column "10 documents", and in the rows "Binary", "Coarse", and "Fine". We then applied RE_s on the test examples as described in Section 2, showing the results in the rows "Binary+Patterns", "Coarse+Patterns", and "Fine+Patterns". The results show that by using syntactico-semantic structures, we obtain significant F-measure improvements of **8.3**, **7.2**, and **5.5** for binary, coarse-grained, and fine-grained relation predictions respectively.

6.3 RE Evaluation Using Predicted Mentions

Next, we perform our experiments using predicted mentions. ACE-2004 defines 7 coarse-grained *entity* types, each of which are then refined into 43 fine-

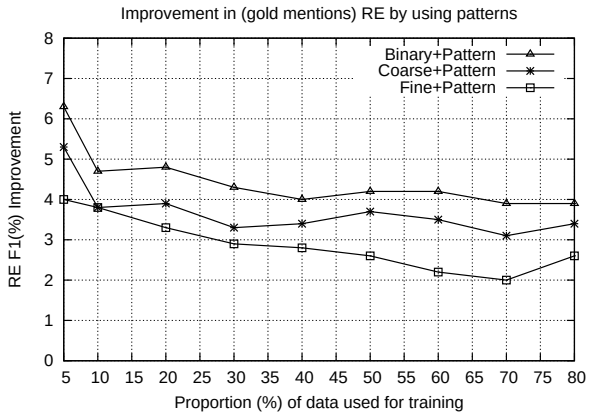


Figure 3: Improvement in (gold mention) RE.

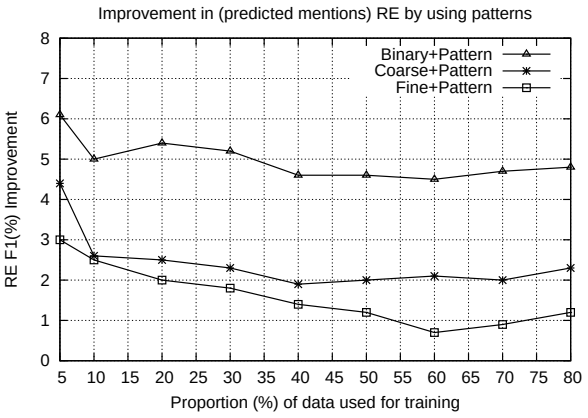


Figure 4: Improvement in (predicted mention) RE.

grained *entity* types. Using the ACE data annotated with mentions and predefined entity types, we build a fine-grained mention entity typing (MET) classifier to disambiguate between 44 labels (43 fine-grained and a null label to indicate not a mention). To obtain the coarse-grained entity type predictions from the classifier, we simply check which coarse-grained type the fine-grained prediction belongs to. We use the LIBLINEAR package with the same settings as earlier specified for the RE system. In each fold, we build a MET classifier using all the (276) training documents in that fold.

We apply RE_{base} on all mention pairs (m_i,m_j) where both m_i and m_j have non null entity type predictions. We show these baseline results in the Rows "Binary", "Coarse", and "Fine" of Table 5.

In Section 2, we described our algorithmic approach (Figure 2) that takes advantage of the structures with predicted mentions. We show the results of this approach in the Rows "Binary+Patterns", "Coarse+Patterns", and "Fine+Patterns" of Table 5. The results show that by leveraging syntactico-semantic structures, we obtain significant F-measure improvements of **8.2**, **4.6**, and **3.6** for binary, coarse-grained, and fine-grained relation predictions respectively.

7 Analysis

We first show statistics regarding the syntactico-semantic structures. In Section 3, we mentioned that ACE-2004 identified five types of structures: premodifier, possessive, preposition, formulaic, and verbal. On the 4,011 examples that we experimented on, premodifiers are the most frequent, accounting for 30.5% of the examples (or about 1,224 examples). The occurrence distributions of the other structures are 18.9% (possessive), 23.9% (preposition), 7.2% (formulaic), and 19.5% (verbal). Hence, the four syntactico-semantic structures that we focused on in this paper account for a large majority (80%) of the relations.

Pattern type	Rec%	Pre%
PreMod	86.8	79.7
Poss	94.3	88.3
Prep	94.6	20.0
Formula	85.5	62.2

Table 6: Recall and precision of the patterns.

In Section 6, we note that out of 55,520 mention pairs, only 4,011 exhibit valid relations. Thus, the proportion of positive relation examples is very sparse at 7.2%. If we can effectively identify and discard most of the negative relation examples, it should improve RE performance, including yielding training data with a more balanced label distribution.

We now analyze the utility of the patterns. As shown in Table 6, the patterns are effective in inferring the structure of mention pairs. For instance, applying the premodifier patterns on the 55,520 mention pairs, we correctly identified 86.8% of the 1,224 premodifier occurrences as premodifiers, while incurring a false-positive rate of only about 20%[6]. We

[6]Random selection will give a precision of about 2.2% (1,224 out of 55,520) and thus a false-positive rate of 97.8%

note that preposition structures are relatively harder to identify. Some of the reasons are due to possibly multiple prepositions in between a mention pair, preposition sense ambiguity, pp-attachment ambiguity, etc. However, in general, we observe that inferring the structures allows us to discard a large portion of the mention pairs which have no valid relation between them. The intuition behind this is the following: if we infer that there is a syntactico-semantic structure between a mention pair, then it is likely that the mention pair exhibits a valid relation. Conversely, if there is a valid relation between a mention pair, then it is likely that there exists a syntactico-semantic structure between the mentions.

Next, we repeat the experiments in Section 6.2 and Section 6.3, while gradually increasing the amount of training data used for training the RE classifiers. The detailed results of using 5% and 80% of all available data are shown in Table 4 and Table 5. Note that these settings are with respect to all 345 documents and thus the 80% setting represents using all 276 training documents in each fold. We plot the intermediate results in Figure 3 and Figure 4. We note that leveraging the structures provides improvements on all experimental settings. Also, intuitively, the *binary* predictions benefit the most from leveraging the structures. How to further exploit this is a possible future work.

8 Conclusion

In this paper, we propose a novel algorithmic approach to RE by exploiting syntactico-semantic structures. We show that this approach provides several advantages and improves RE performance. There are several interesting directions for future work. There are probably many near misses when we apply our structure patterns on predicted mentions. For instance, for both premodifier and possessive structures, we require that one mention completely includes the other. Relaxing this might potentially recover additional valid mention pairs and improve performance. We could also try to learn classifiers to automatically identify and disambiguate between the different syntactico-semantic structures. It will also be interesting to feedback the predictions of the structure patterns to the mention entity typing classifier and possibly retrain to obtain a better classifier.

Acknowledgements This research is supported by the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

We thank Ming-Wei Chang and Quang Do for building the mention extraction system.

References

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Yee Seng Chan and Dan Roth. 2010. Exploiting background knowledge for relation extraction. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 152–160.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2007. Relex – Relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Mark A. Greenwood and Mark Stevenson. 2006. Improving semi-supervised acquisition of relation extraction patterns. In *Proceedings of the COLING-ACL Workshop on Information Extraction Beyond The Document*, pages 29–35.

Jing Jiang and ChengXiang Zhai. 2007. A systematic exploration of the feature space for relation extraction. In *Proceedings of Human Language Technologies - North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 113–120.

Jing Jiang. 2009. Multi-task transfer learning for weakly-supervised relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 1012–1020.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 178–181.

Dan Klein and Christoper D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 3–10.

Longhua Qian, Guodong Zhou, Qiaomin Zhu, and Peide Qian. 2008. Relation extraction using convolution tree kernel expanded with entity features. In *Pacific Asia Conference on Language, Information and Computation*, pages 415–421.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 147–155.

Dan Roth and Wen Tau Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press.

Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 427–434.

Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, and Ji-Rong Wen. 2009. Statsnowball: a statistical approach to extracting entity relationships. In *The International World Wide Web Conference*, pages 101–110.

Together We Can: Bilingual Bootstrapping for WSD

Mitesh M. Khapra Salil Joshi Arindam Chatterjee Pushpak Bhattacharyya
Department Of Computer Science and Engineering,
IIT Bombay,
Powai,
Mumbai, 400076.
{miteshk,salilj,arindam,pb}@cse.iitb.ac.in

Abstract

Recent work on bilingual Word Sense Disambiguation (WSD) has shown that a resource deprived language (L_1) can benefit from the annotation work done in a resource rich language (L_2) via parameter projection. However, this method assumes the presence of sufficient annotated data in one resource rich language which may not always be possible. Instead, we focus on the situation where there are two resource deprived languages, both having a very small amount of seed annotated data and a large amount of untagged data. We then use ***bilingual bootstrapping***, wherein, a model trained using the seed annotated data of L_1 is used to annotate the untagged data of L_2 and vice versa using parameter projection. The untagged instances of L_1 and L_2 which get annotated with high confidence are then added to the seed data of the respective languages and the above process is repeated. Our experiments show that such a bilingual bootstrapping algorithm when evaluated on two different domains with ***small seed sizes*** using Hindi (L_1) and Marathi (L_2) as the language pair performs better than monolingual bootstrapping and significantly reduces annotation cost.

1 Introduction

The high cost of collecting sense annotated data for supervised approaches (Ng and Lee, 1996; Lee et al., 2004) has always remained a matter of concern for some of the resource deprived languages of the world. The problem is even more hard-hitting for multilingual regions (*e.g.*, India which has more than 20 constitutionally recognized languages). To circumvent this problem, unsupervised and knowledge based approaches (Lesk, 1986; Walker and Amsler, 1986; Agirre and Rigau, 1996; McCarthy et al., 2004; Mihalcea, 2005) have been proposed as an alternative but they have failed to deliver good accuracies. Semi-supervised approaches (Yarowsky, 1995) which use a small amount of annotated data and a large amount of untagged data have shown promise albeit for a limited set of target words. The above situation highlights the need for high accuracy resource conscious approaches to all-words multilingual WSD.

Recent work by Khapra et al. (2010) in this direction has shown that it is possible to perform cost effective WSD in a target language (L_2) without compromising much on accuracy by leveraging on the annotation work done in another language (L_1). This is achieved with the help of a novel synset-aligned multilingual dictionary which facilitates the projection of parameters learned from the Wordnet and annotated corpus of L_1 to L_2. This approach thus obviates the need for collecting large amounts of annotated corpora in multiple languages by relying on sufficient annotated corpus in one resource rich language. However, in many situations such a pivot resource rich language itself may not be available. Instead, we might have two or more languages having a small amount of annotated corpus and a large amount of untagged corpus. Addressing such situations is the main focus of this work. Specifically, we address the following question:

> *In the absence of a pivot resource rich language is it possible for two resource deprived languages to mutually benefit from each other's annotated data?*

While addressing the above question we assume that

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 561–569,
Portland, Oregon, June 19-24, 2011.

even though it is hard to obtain large amounts of annotated data in multiple languages, it should be fairly easy to obtain a large amount of untagged data in these languages. We leverage on such untagged data by employing a bootstrapping strategy. The idea is to train an initial model using a small amount of annotated data in both the languages and iteratively expand this seed data by including untagged instances which get tagged with a high confidence in successive iterations. Instead of using monolingual bootstrapping, we use bilingual bootstrapping via parameter projection. In other words, the parameters learned from the annotated data of L_1 (and L_2 respectively) are projected to L_2 (and L_1 respectively) and the projected model is used to tag the untagged instances of L_2 (and L_1 respectively).

Such a bilingual bootstrapping strategy when tested on two domains, *viz.*, Tourism and Health using Hindi (L_1) and Marathi (L_2) as the language pair, consistently does better than a baseline strategy which uses only seed data for training without performing any bootstrapping. Further, it consistently performs better than monolingual bootstrapping. A simple and intuitive explanation for this is as follows. In monolingual bootstrapping a language can benefit only from its own seed data and hence can tag only those instances with high confidence which it has already seen. On the other hand, in bilingual bootstrapping a language can benefit from the seed data available in the other language which was not previously seen in its self corpus. This is very similar to the process of co-training (Blum and Mitchell, 1998) wherein the annotated data in the two languages can be seen as two different views of the same data. Hence, the classifier trained on one view can be improved by adding those untagged instances which are tagged with a high confidence by the classifier trained on the other view.

The remainder of this paper is organized as follows. In section 2 we present related work. Section 3 describes the Synset aligned multilingual dictionary which facilitates parameter projection. Section 4 discusses the work of Khapra et al. (2009) on parameter projection. In section 5 we discuss bilingual bootstrapping which is the main focus of our work followed by a brief discussion on monolingual bootstrapping. Section 6 describes the experimental setup. In section 7 we present the results followed by discussion in section 8. Section 9 concludes the paper.

2 Related Work

Bootstrapping for Word Sense Disambiguation was first discussed in (Yarowsky, 1995). Starting with a very small number of seed collocations an initial decision list is created. This decisions list is then applied to untagged data and the instances which get tagged with a high confidence are added to the seed data. This algorithm thus proceeds iteratively increasing the seed size in successive iterations. This monolingual bootstrapping method showed promise when tested on a limited set of target words but was not tried for all-words WSD.

The failure of monolingual approaches (Ng and Lee, 1996; Lee et al., 2004; Lesk, 1986; Walker and Amsler, 1986; Agirre and Rigau, 1996; McCarthy et al., 2004; Mihalcea, 2005) to deliver high accuracies for all-words WSD at low costs created interest in bilingual approaches which aim at reducing the annotation effort. Recent work in this direction by Khapra et al. (2009) aims at reducing the annotation effort in multiple languages by leveraging on existing resources in a pivot language. They showed that it is possible to project the parameters learned from the annotation work of one language to another language provided aligned Wordnets for the two languages are available. However, they do not address situations where two resource deprived languages have aligned Wordnets but neither has sufficient annotated data. In such cases bilingual bootstrapping can be used so that the two languages can mutually benefit from each other's small annotated data.

Li and Li (2004) proposed a bilingual bootstrapping approach for the more specific task of Word Translation Disambiguation (WTD) as opposed to the more general task of WSD. This approach does not need parallel corpora (just like our approach) and relies only on in-domain corpora from two languages. However, their work was evaluated only on a handful of target words (9 nouns) for WTD as opposed to the broader task of WSD. Our work instead focuses on improving the performance of all words WSD for two resource deprived languages using bilingual bootstrapping. At the heart of our work lies ***parameter projection*** facilitated by a synset aligned

multilingual dictionary described in the next section.

3 Synset Aligned Multilingual Dictionary

A novel and effective method of storage and use of dictionary in a multilingual setting was proposed by Mohanty et al. (2008). For the purpose of current discussion, we will refer to this multilingual dictionary framework as *MultiDict*. One important departure in this framework from the traditional dictionary is that **synsets are linked, and after that the words inside the synsets are linked**. The basic mapping is thus between synsets and thereafter between the words.

Concepts	L1 (English)	L2 (Hindi)	L3 (Marathi)
04321: a youthful male person	{male child, boy}	{लड़का *(ladkaa)*, बालक *(baalak)*, बच्चा *(bachchaa)*}	{मुलगा *(mulgaa)*, पोरगा *(porgaa)*, पोर *(por)*}

Table 1: Multilingual Dictionary Framework

Table 1 shows the structure of MultiDict, with one example row standing for the concept of *boy*. The first column is the pivot describing a concept with a unique ID. The subsequent columns show the words expressing the concept in respective languages (in the example table, *English, Hindi and Marathi*). After the synsets are linked, cross linkages are set up manually from the words of a synset to the words of a linked synset of the pivot language. For example, for the Marathi word मुलगा *(mulgaa)*, "a youthful male person", the correct lexical substitute from the corresponding Hindi synset is लड़का *(ladkaa)*. The average number of such links per synset per language pair is approximately 3. However, since our work takes place in a semi-supervised setting, we do not assume the presence of these manual cross linkages between synset members. Instead, in the above example, we assume that all the words in the Hindi synset are equally probable translations of every word in the corresponding Marathi synset. Such cross-linkages between synset members facilitate parameter projection as explained in the next section.

4 Parameter Projection

Khapra et al. (2009) proposed that the various parameters essential for domain-specific Word Sense Disambiguation can be broadly classified into two categories:

Wordnet-dependent parameters:

- belongingness-to-dominant-concept
- conceptual distance
- semantic distance

Corpus-dependent parameters:

- sense distributions
- corpus co-occurrence

They proposed a scoring function (Equation (1)) which combines these parameters to identify the correct sense of a word in a context:

$$S^* = \arg\max_i(\theta_i V_i + \sum_{j \in J} W_{ij} * V_i * V_j) \qquad (1)$$

where,

$$\begin{aligned} i &\in \textit{Candidate Synsets} \\ J &= \textit{Set of disambiguated words} \\ \theta_i &= BelongingnessToDominantConcept(S_i) \\ V_i &= P(S_i|word) \\ W_{ij} &= CorpusCooccurrence(S_i, S_j) \\ &\quad * 1/WNConceptualDistance(S_i, S_j) \\ &\quad * 1/WNSemanticGraphDistance(S_i, S_j) \end{aligned}$$

The first component $\theta_i V_i$ of Equation (1) captures influence of the corpus specific sense of a word in a domain. The other component $W_{ij} * V_i * V_j$ captures the influence of interaction of the candidate sense with the senses of context words weighted by factors of co-occurrence, conceptual distance and semantic distance.

Wordnet-dependent parameters depend on the structure of the Wordnet whereas the ***Corpus-dependent parameters*** depend on various statistics learned from a sense marked corpora. Both the tasks of (a) constructing a Wordnet from scratch and (b) collecting sense marked corpora for multiple languages are tedious and expensive. Khapra et

al. (2009) observed that by ***projecting relations*** from the Wordnet of a language and by ***projecting corpus statistics*** from the sense marked corpora of the language to those of the target language, *the effort required in constructing semantic graphs for multiple Wordnets and collecting sense marked corpora for multiple languages can be avoided or reduced*. At the heart of their work lies the *MultiDict* described in previous section which facilitates parameter projection in the following manner:

1. By linking with the synsets of a pivot resource rich language (Hindi, in our case), the cost of building Wordnets of other languages is partly reduced (semantic relations are inherited). The Wordnet parameters of Hindi Wordnet now become projectable to other languages.
2. For calculating corpus specific sense distributions, $P(Sense\ S_i|Word\ W)$, we need the counts, $\#(S_i, W)$. By using cross linked words in the synsets, these counts become projectable to the target language (Marathi, in our case) as they can be approximated by the counts of the cross linked Hindi words calculated from the Hindi sense marked corpus as follows:

$$P(S_i|W) = \frac{\#(S_i, marathi\_word)}{\sum_j \#(S_j, marathi\_word)}$$

$$P(S_i|W) \approx \frac{\#(S_i, cross\_linked\_hindi\_word)}{\sum_j \#(S_j, cross\_linked\_hindi\_word)}$$

The rationale behind the above approximation is the observation that within a domain the counts of cross-linked words will remain the same across languages.

This parameter projection strategy as explained above lies at the heart of our work and allows us to perform bilingual bootstrapping by projecting the models learned from one language to another.

5 Bilingual Bootstrapping

We now come to the main contribution of our work, *i.e.*, bilingual bootstrapping. As shown in Algorithm 1, we start with a small amount of seed data (LD_1 and LD_2) in the two languages. Using this data we learn the parameters described in the previous section. We collectively refer to the parameters learned from the seed data as models θ_1 and θ_2 for L_1 and L_2 respectively. The parameter projection strategy described in the previous section is then applied to θ_1 and θ_2 to obtain the projected models $\hat{\theta}_2$ and $\hat{\theta}_1$ respectively. These projected models are then applied to the untagged data of L_1 and L_2 and the instances which get labeled with a high confidence are added to the labeled data of the respective languages. This process is repeated till we reach convergence, *i.e.*, till it is no longer possible to move any data from UD_1 (and UD_2) to LD_1 (and LD_2 respectively).

We compare our algorithm with monolingual bootstrapping where the self models θ_1 and θ_2 are directly used to annotate the unlabeled instances in L_1 and L_2 respectively instead of using the projected models $\hat{\theta}_1$ and $\hat{\theta}_2$. The process of monolingual boot-

Algorithm 1 *Bilingual Bootstrapping*

LD_1 := Seed Labeled Data from L_1
LD_2 := Seed Labeled Data from L_2
UD_1 := Unlabeled Data from L_1
UD_2 := Unlabeled Data from L_2

repeat
  θ_1 := model trained using LD_1
  θ_2 := model trained using LD_2

  {Project models from L_1/L_2 to L_2/L_1}
  $\hat{\theta}_2$:= *project*(θ_1, L_2)
  $\hat{\theta}_1$:= *project*(θ_2, L_1)

  for all $u_1 \in UD_1$ **do**
    s := sense assigned by $\hat{\theta}_1$ to u_1
    if *confidence(s)* $> \epsilon$ **then**
      LD_1 := LD_1 + u_1
      UD_1 := UD_1 - u_1
    end if
  end for

  for all $u_2 \in UD_2$ **do**
    s := sense assigned by $\hat{\theta}_2$ to u_2
    if *confidence(s)* $> \epsilon$ **then**
      LD_2 := LD_2 + u_2
      UD_2 := UD_2 - u_2
    end if
  end for
until *convergence*

Algorithm 2 *Monolingual Bootstrapping*

LD_1 := Seed Labeled Data from L_1
LD_2 := Seed Labeled Data from L_2
UD_1 := Unlabeled Data from L_1
UD_2 := Unlabeled Data from L_2

repeat
  θ_1 := model trained using LD_1
  θ_2 := model trained using LD_2
  for all $u_1 \in UD_1$ **do**
    s := sense assigned by θ_1 to u_1
    if *confidence(s)* $> \epsilon$ **then**
      LD_1 := LD_1 + u_1
      UD_1 := UD_1 - u_1
    end if
  end for

  for all $u_2 \in UD_2$ **do**
    s := sense assigned by θ_2 to u_2
    if *confidence(s)* $> \epsilon$ **then**
      LD_2 := LD_2 + u_2
      UD_2 := UD_2 - u_2
    end if
  end for
until *convergence*

strapping is shown in Algorithm 2.

6 Experimental Setup

We used the publicly available dataset[1] described in Khapra et al. (2010) for all our experiments. The data was collected from two domains, *viz.*, Tourism and Health. The data for Tourism domain was collected by manually translating English documents downloaded from Indian Tourism websites into Hindi and Marathi. Similarly, English documents for Health domain were obtained from two doctors and were manually translated into Hindi and Marathi. The entire data was then manually annotated by three lexicographers adept in Hindi and Marathi. The various statistics pertaining to the total number of words, number of words per POS category and average degree of polysemy are described in Tables 2 to 5.

Although Tables 2 and 3 also report the num-

[1] http://www.cfilt.iitb.ac.in/wsd/annotated_corpus

	Polysemous words		Monosemous words	
Category	**Tourism**	**Health**	**Tourism**	**Health**
Noun	62336	24089	35811	18923
Verb	6386	1401	3667	5109
Adjective	18949	8773	28998	12138
Adverb	4860	2527	13699	7152
All	92531	36790	82175	43322

Table 2: Polysemous and Monosemous words per category in each domain for Hindi

	Polysemous words		Monosemous words	
Category	**Tourism**	**Health**	**Tourism**	**Health**
Noun	45589	17482	27386	11383
Verb	7879	3120	2672	1500
Adjective	13107	4788	16725	6032
Adverb	4036	1727	5023	1874
All	70611	27117	51806	20789

Table 3: Polysemous and Monosemous words per category in each domain for Marathi

	Avg. degree of Wordnet polysemy for polysemous words	
Category	**Tourism**	**Health**
Noun	3.02	3.17
Verb	5.05	6.58
Adjective	2.66	2.75
Adverb	2.52	2.57
All	3.09	3.23

Table 4: Average degree of Wordnet polysemy per category in the 2 domains for Hindi

	Avg. degree of Wordnet polysemy for polysemous words	
Category	**Tourism**	**Health**
Noun	3.06	3.18
Verb	4.96	5.18
Adjective	2.60	2.72
Adverb	2.44	2.45
All	3.14	3.29

Table 5: Average degree of Wordnet polysemy per category in the 2 domains for Marathi

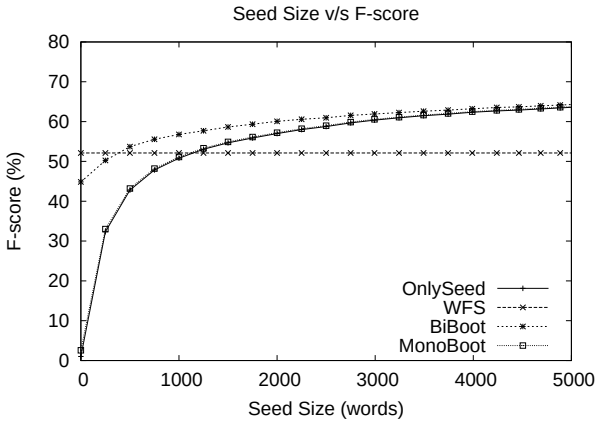


Figure 1: Comparison of *BiBoot*, *MonoBoot*, *OnlySeed* and *WFS* on Hindi Health data

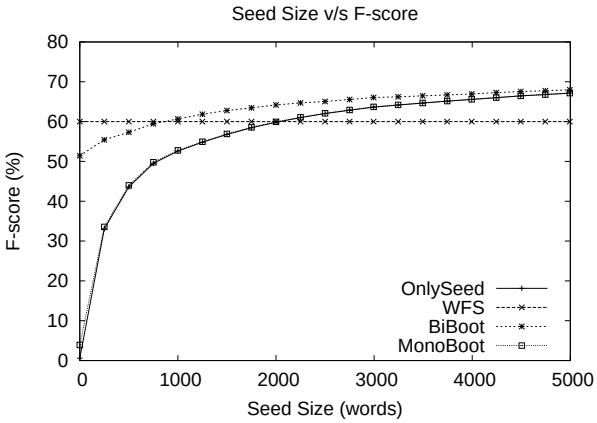


Figure 2: Comparison of *BiBoot*, *MonoBoot*, *OnlySeed* and *WFS* on Hindi Tourism data

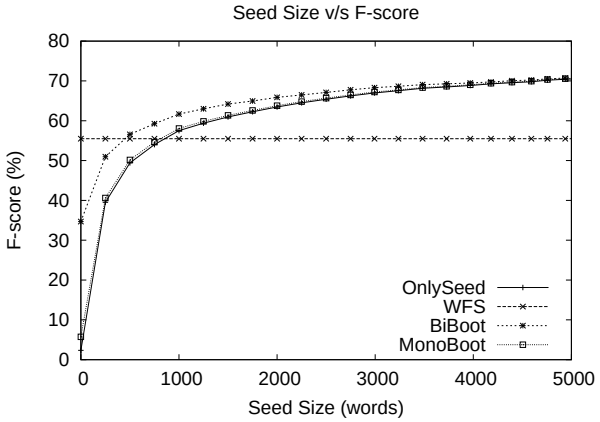


Figure 3: Comparison of *BiBoot*, *MonoBoot*, *OnlySeed* and *WFS* on Marathi Health data

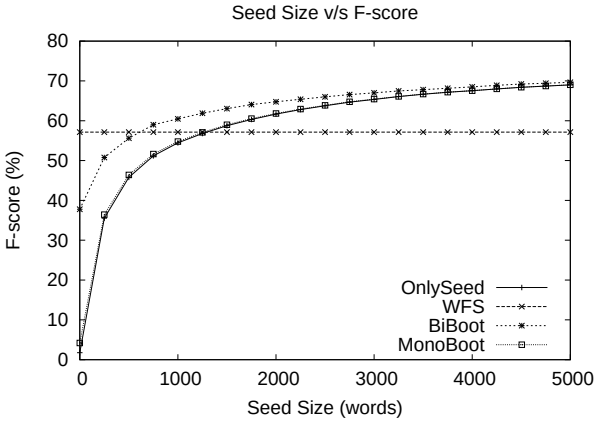


Figure 4: Comparison of *BiBoot*, *MonoBoot*, *OnlySeed* and *WFS* on Marathi Tourism data

ber of monosemous words, we would like to clearly state that we do not consider monosemous words while evaluating the performance of our algorithms (as monosemous words do not need any disambiguation).

We did a 4-fold cross validation of our algorithm using the above described corpora. Note that even though the corpora were parallel we did not use this property in any way in our experiments or algorithm. In fact, the documents in the two languages were randomly split into 4 folds without ensuring that the parallel documents remain in the same folds for the two languages. We experimented with different seed sizes varying from 0 to 5000 in steps of 250. The seed annotated data and untagged instances for bootstrapping are extracted from 3 folds of the data and the final evaluation is done on the held-out data in the 4th fold.

We ran both the bootstrapping algorithms (*i.e.*, *monolingual bootstrapping* and *bilingual bootstrapping*) for 10 iterations but, we observed that after 1-2 iterations the algorithms converge. In each iteration only those words for which $P(assigned\_sense|word) > 0.6$ get moved to the labeled data. Ideally, this threshold (0.6) should have been selected using a development set. However, since our work focuses on resource scarce languages we did not want to incur the additional cost of using a development set. Hence, we used a fixed threshold of 0.6 so that in each iteration only those words get moved to the labeled data for which the assigned sense is clearly a majority sense ($P > 0.6$).

Language-Domain	Algorithm	F-score(%)	No. of tagged words needed to achieve this F-score	% Reduction in annotation cost
Hindi-Health	*Biboot*	57.70	1250	
	OnlySeed	57.99	2250	$\frac{(2250+2250)-(1250+1750)}{(2250+2250)} * 100 = 33.33\%$
Marathi-Health	*Biboot*	64.97	1750	
	OnlySeed	64.51	2250	
Hindi-Tourism	*Biboot*	60.67	1000	
	OnlySeed	59.83	2000	$\frac{(2000+2000)-(1000+1250)}{(2000+2000)} * 100 = 43.75\%$
Marathi-Tourism	*Biboot*	61.90	1250	
	OnlySeed	61.68	2000	

Table 6: Reduction in annotation cost achieved using Bilingual Bootstrapping

7 Results

The results of our experiments are summarized in Figures 1 to 4. The x-axis represents the amount of seed data used and the y-axis represents the F-scores obtained. The different curves in each graph are as follows:

a. ***BiBoot***: This curve represents the F-score obtained after 10 iterations by using bilingual bootstrapping with different amounts of seed data.

b. ***MonoBoot***: This curve represents the F-score obtained after 10 iterations by using monolingual bootstrapping with different amounts of seed data.

c. ***OnlySeed***: This curve represents the F-score obtained by training on the seed data alone without using any bootstrapping.

d. ***WFS***: This curve represents the F-score obtained by simply selecting the first sense from Wordnet, a typically reported baseline.

8 Discussions

In this section we discuss the important observations made from Figures 1 to 4.

8.1 Performance of Bilingual bootstrapping

For small seed sizes, the F-score of bilingual bootstrapping is consistently better than the F-score obtained by training only on the seed data without using any bootstrapping. This is true for both the languages in both the domains. Further, bilingual bootstrapping also does better than monolingual bootstrapping for small seed sizes. As explained earlier, this better performance can be attributed to the fact that in monolingual bootstrapping the algorithm can tag only those instances with high confidence which it has already seen in the training data. Hence, in successive iterations, very little new information becomes available to the algorithm. This is clearly evident from the fact that the curve of monolingual bootstrapping (*MonoBoot*) is always close to the curve of *OnlySeed*.

8.2 Effect of seed size

The benefit of bilingual bootstrapping is clearly felt for small seed sizes. However, as the seed size increases the performance of the 3 algorithms, *viz.*, *MonoBoot*, *BiBoot* and *OnlySeed* is more or less the same. This is intuitive, because, as the seed size increases the algorithm is able to see more and more tagged instances in its self corpora and hence does not need any assistance from the other language. In other words, the annotated data in L_1 is not able to add any new information to the training process of L_2 and vice versa.

8.3 Bilingual bootstrapping reduces annotation cost

The performance boost obtained at small seed sizes suggests that bilingual bootstrapping helps to reduce the overall annotation costs for both the languages. To further illustrate this, we take some sample points from the graph and compare the number of tagged words needed by *BiBoot* and *OnlySeed* to reach the same (or nearly the same) F-score. We present this comparison in Table 6.

The rows for *Hindi-Health* and *Marathi-Health* in Table 6 show that when *BiBoot* is employed we need 1250 tagged words in Hindi and 1750 tagged words in Marathi to attain F-scores of 57.70% and 64.97% respectively. On the other hand, in the absence of bilingual bootstrapping, (*i.e.*, using *OnlySeed*) we need 2250 tagged words each in Hindi and Marathi to achieve similar F-scores. *BiBoot* thus gives a reduction of 33.33% in the overall annotation cost ($\{1250 + 1750\}$ v/s $\{2250 + 2250\}$) while achieving similar F-scores. Similarly, the results for *Hindi-Tourism* and *Marathi-Tourism* show that *BiBoot* gives a reduction of 43.75% in the overall annotation cost while achieving similar F-scores. Further, since the results of *MonoBoot* are almost the same as *OnlySeed*, the above numbers indicate that *BiBoot* provides a reduction in cost when compared to *MonoBoot* also.

8.4 Contribution of monosemous words in the performance of *BiBoot*

As mentioned earlier, monosemous words in the test set are not considered while evaluating the performance of our algorithm but, we add monosemous words to the seed data. However, we do not count monosemous words while calculating the seed size as there is no manual annotation cost associated with monosemous words (they can be tagged automatically by fetching their singleton sense id from the wordnet). We observed that the monosemous words of L_1 help in boosting the performance of L_2 and vice versa. This is because for a given monosemous word in L_2 (or L_1 respectively) the corresponding cross-linked word in L_1 (or L_2 respectively) need not necessarily be monosemous. In such cases, the cross-linked polysemous word in L_2 (or L_1 respectively) benefits from the projected statistics of a monosemous word in L_1 (or L_2 respectively). This explains why *BiBoot* gives an F-score of 35-52% even at zero seed size even though the F-score of *OnlySeed* is only 2-5% (see Figures 1 to 4).

9 Conclusion

We presented a bilingual bootstrapping algorithm for Word Sense Disambiguation which allows two resource deprived languages to mutually benefit from each other's data via parameter projection. The algorithm consistently performs better than monolingual bootstrapping. It also performs better than using only monolingual seed data without using any bootstrapping. The benefit of bilingual bootstrapping is felt prominently when the seed size in the two languages is very small thus highlighting the usefulness of this algorithm in highly resource constrained scenarios.

Acknowledgments

We acknowledge the support of Microsoft Research India in the form of an International Travel Grant, which enabled one of the authors (Mitesh M. Khapra) to attend this conference.

References

Eneko Agirre and German Rigau. 1996. Word sense disambiguation using conceptual density. In *In Proceedings of the 16th International Conference on Computational Linguistics (COLING)*.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. pages 92–100. Morgan Kaufmann Publishers.

Mitesh M. Khapra, Sapan Shah, Piyush Kedia, and Pushpak Bhattacharyya. 2009. Projecting parameters for multilingual word sense disambiguation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 459–467, Singapore, August. Association for Computational Linguistics.

Mitesh Khapra, Saurabh Sohoney, Anup Kulkarni, and Pushpak Bhattacharyya. 2010. Value for money: Balancing annotation effort, lexicon building and accuracy for multilingual wsd. In *Proceedings of the 23rd International Conference on Computational Linguistics*.

Yoong Keok Lee, Hwee Tou Ng, and Tee Kiah Chia. 2004. Supervised word sense disambiguation with support vector machines and multiple knowledge sources. In *Proceedings of Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 137–140.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *In Proceedings of the 5th annual international conference on Systems documentation*.

Hang Li and Cong Li. 2004. Word translation disambiguation using bilingual bootstrapping. *Comput. Linguist.*, 30:1–22, March.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant word senses in untagged text. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 279, Morristown, NJ, USA. Association for Computational Linguistics.

Rada Mihalcea. 2005. Large vocabulary unsupervised word sense disambiguation with graph-based algorithms for sequence data labeling. In *In Proceedings of the Joint Human Language Technology and Empirical Methods in Natural Language Processing Conference (HLT/EMNLP)*, pages 411–418.

Rajat Mohanty, Pushpak Bhattacharyya, Prabhakar Pande, Shraddha Kalele, Mitesh Khapra, and Aditya Sharma. 2008. Synset based multilingual dictionary: Insights, applications and challenges. In *Global Wordnet Conference*.

Hwee Tou Ng and Hian Beng Lee. 1996. Integrating multiple knowledge sources to disambiguate word senses: An exemplar-based approach. In *In Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 40–47.

D. Walker and R. Amsler. 1986. The use of machine readable dictionaries in sublanguage analysis. In *In Analyzing Language in Restricted Domains, Grishman and Kittredge (eds), LEA Press*, pages 69–83.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196, Morristown, NJ, USA. Association for Computational Linguistics.

Which Noun Phrases Denote Which Concepts?

Jayant Krishnamurthy
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
jayantk@cs.cmu.edu

Tom M. Mitchell
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
tom.mitchell@cmu.edu

Abstract

Resolving polysemy and synonymy is required for high-quality information extraction. We present ConceptResolver, a component for the Never-Ending Language Learner (NELL) (Carlson et al., 2010) that handles both phenomena by identifying the latent concepts that noun phrases refer to. ConceptResolver performs both word sense induction and synonym resolution on relations extracted from text using an ontology and a small amount of labeled data. Domain knowledge (the ontology) guides concept creation by defining a set of possible semantic types for concepts. Word sense induction is performed by inferring a set of semantic types for each noun phrase. Synonym detection exploits redundant information to train several domain-specific synonym classifiers in a semi-supervised fashion. When ConceptResolver is run on NELL's knowledge base, 87% of the word senses it creates correspond to real-world concepts, and 85% of noun phrases that it suggests refer to the same concept are indeed synonyms.

1 Introduction

Many information extraction systems construct knowledge bases by extracting structured assertions from free text (e.g., NELL (Carlson et al., 2010), TextRunner (Banko et al., 2007)). A major limitation of many of these systems is that they fail to distinguish between noun phrases and the underlying concepts they refer to. As a result, a polysemous phrase like "apple" will refer sometimes to the concept *Apple Computer (the company)*, and other times to the concept *apple (the fruit)*. Furthermore, two synonymous noun phrases like "apple" and "Apple Computer" can refer to the same underlying concept. The result of ignoring this many-to-many mapping between noun phrases and underlying concepts (see Figure 1) is confusion about the meaning of extracted information. To minimize such confusion, a system must separately represent noun phrases, the underlying concepts to which they can refer, and the many-to-many "can refer to" relation between them.

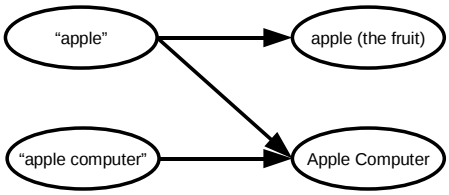


Figure 1: An example mapping from noun phrases (left) to a set of underlying concepts (right). Arrows indicate which noun phrases can refer to which concepts.

```
[eli lilly, lilly]
[kaspersky labs, kaspersky lab, kaspersky]
[careerbuilder, careerbuilder.com]
[l 3 communications, level 3 communications]
[cellular, u.s. cellular]
[jc penney, jc penny]
[nielsen media research, nielsen company]
[universal studios, universal music group, universal]
[amr corporation, amr]
[intel corp, intel corp., intel corporation, intel]

[emmitt smith, chris canty]
[albert pujols, pujols]
[carlos boozer, dennis martinez]
[jason hirsh, taylor buchholz]
[chris snyder, ryan roberts]
[j.p. losman, losman, jp losman]
[san francisco giants, francisco rodriguez]
[andruw jones, andruw]
[aaron heilman, bret boone]
[roberto clemente, clemente]
```

Figure 2: A random sample of concepts created by ConceptResolver. The first 10 concepts are from **company**, while the second 10 are from **athlete**.

The relations extracted by systems like NELL actually apply to concepts, not to noun phrases. Say

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 570–580,
Portland, Oregon, June 19-24, 2011.

the system extracts the relation **ceoOf**(x_1, x_2) between the noun phrases x_1 and x_2. The correct interpretation of this extracted relation is that there exist concepts c_1 and c_2 such that x_1 can refer to c_1, x_2 can refer to c_2, and **ceoOf**(c_1, c_2). If the original relation were **ceoOf**("steve", "apple"), then c_1 would be *Steve Jobs*, and c_2 would be *Apple Computer*. A similar interpretation holds for one-place category predicates like **person**(x_1). We define *concept discovery* as the problem of (1) identifying concepts like c_1 and c_2 from extracted predicates like **ceoOf**(x_1, x_2) and (2) mapping noun phrases like x_1, x_2 to the concepts they can refer to.

The main input to ConceptResolver is a set of extracted category and relation instances over noun phrases, like **person**(x_1) and **ceoOf**(x_1, x_2), produced by running NELL. Here, any individual noun phrase x_i can be labeled with multiple categories and relations. The output of ConceptResolver is a set of concepts, $\{c_1, c_2, ..., c_n\}$, and a mapping from each noun phrase in the input to the set of concepts it can refer to. Like many other systems (Miller, 1995; Yates and Etzioni, 2007; Lin and Pantel, 2002), ConceptResolver represents each output concept c_i as a set of synonymous noun phrases, i.e., $c_i = \{x_{i1}, x_{i2}, ..., x_{im}\}$. For example, Figure 2 shows several concepts output by ConceptResolver; each concept clearly reveals which noun phrases can refer to it. Each concept also has a semantic type that corresponds to a category in ConceptResolver's ontology; for instance, the first 10 concepts in Figure 2 belong to the category **company**.

Previous approaches to concept discovery use little prior knowledge, clustering noun phrases based on co-occurrence statistics (Pantel and Lin, 2002). In comparison, ConceptResolver uses a knowledge-rich approach. In addition to the extracted relations, ConceptResolver takes as input two other sources of information: an ontology, and a small number of labeled synonyms. The ontology contains a schema for the relation and category predicates found in the input instances, including properties of predicates like type restrictions on its domain and range. The category predicates are used to assign semantic types to each concept, and the properties of relation predicates are used to create evidence for synonym resolution. The labeled synonyms are used as training data during synonym resolution, where they are used to train a semi-supervised classifier.

1. **Induce Word Senses**
   i. Use extracted category instances to create one or more senses per noun phrase.
   ii. Use argument type constraints to produce relation evidence for synonym resolution.
2. **Cluster Synonymous Senses**
   For each category C defined in the ontology:
   i. Train a semi-supervised classifier to predict synonymy.
   ii. Cluster word senses with semantic type C using classifier's predictions.
   iii. Output sense clusters as concepts with semantic type C.

Figure 3: High-level outline of ConceptResolver's algorithm.

ConceptResolver discovers concepts using the process outlined in Figure 3. It first performs word sense induction, using the extracted category instances to create one or more unambiguous word senses for each noun phrase in the knowledge base. Each word sense is a copy of the original noun phrase paired with a semantic type (a category) that restricts the concepts it can refer to. ConceptResolver then performs synonym resolution on these word senses. This step treats the senses of each semantic type independently, first training a synonym classifier then clustering the senses based on the classifier's decisions. The result of this process is clusters of synonymous word senses, which are output as concepts. Concepts inherit the semantic type of the word senses they contain.

We evaluate ConceptResolver using a subset of NELL's knowledge base, presenting separate results for the concepts of each semantic type. The evaluation shows that, on average, 87% of the word senses created by ConceptResolver correspond to real-world concepts. We additionally find that, on average, 85% of the noun phrases in each concept refer to the same real-world entity.

2 Prior Work

Previous work on concept discovery has focused on the subproblems of word sense induction and synonym resolution. Word sense induction is typically performed using unsupervised clustering. In the SemEval word sense induction and disambigua-

tion task (Agirre and Soroa, 2007; Manandhar et al., 2010), all of the submissions in 2007 created senses by clustering the contexts each word occurs in, and the 2010 event explicitly disallowed the use of external resources like ontologies. Other systems cluster words to find both word senses and concepts (Pantel and Lin, 2002; Lin and Pantel, 2002). ConceptResolver's category-based approach is quite different from these clustering approaches. Snow et al. (2006) describe a system which adds new word senses to WordNet. However, Snow et al. assume the existence of an oracle which provides the senses of each word. In contrast, ConceptResolver automatically determines the number of senses for each word.

Synonym resolution on relations extracted from web text has been previously studied by Resolver (Yates and Etzioni, 2007), which finds synonyms in relation triples extracted by TextRunner (Banko et al., 2007). In contrast to our system, Resolver is unsupervised and does not have a schema for the relations. Due to different inputs, ConceptResolver and Resolver are not precisely comparable. However, our evaluation shows that ConceptResolver has higher synonym resolution precision than Resolver, which we attribute to our semi-supervised approach and the known relation schema.

Synonym resolution also arises in record linkage (Winkler, 1999; Ravikumar and Cohen, 2004) and citation matching (Bhattacharya and Getoor, 2007; Bhattacharya and Getoor, 2006; Poon and Domingos, 2007). As with word sense induction, many approaches to these problems are unsupervised. A problem with these algorithms is that they require the authors to define domain-specific similarity heuristics to achieve good performance. Other synonym resolution work is fully supervised (Singla and Domingos, 2006; McCallum and Wellner, 2004; Snow et al., 2007), training models using manually constructed sets of synonyms. These approaches use large amounts of labeled data, which can be difficult to create. ConceptResolver's approach lies between these two extremes: we label a small number of synonyms (10 pairs), then use semi-supervised training to learn a similarity function. We think our technique is a good compromise, as it avoids much of the manual effort of the other approaches: tuning the similarity function in one case, and labeling a large amount of data in the other

ConceptResolver uses a novel algorithm for semi-supervised clustering which is conceptually similar to other work in the area. Like other approaches (Basu et al., 2004; Xing et al., 2003; Klein et al., 2002), we learn a similarity measure for clustering based on a set of must-link and cannot-link constraints. Unlike prior work, our algorithm exploits multiple views of the data to improve the similarity measure. As far as we know, ConceptResolver is the first application of semi-supervised clustering to relational data – where the items being clustered are connected by relations (Getoor and Diehl, 2005). Interestingly, the relational setting also provides us with the independent views that are beneficial to semi-supervised training.

Concept discovery is also related to coreference resolution (Ng, 2008; Poon and Domingos, 2008). The difference between the two problems is that coreference resolution finds noun phrases that refer to the same concept within a specific document. We think the concepts produced by a system like ConceptResolver could be used to improve coreference resolution by providing prior knowledge about noun phrases that can refer to the same concept. This knowledge could be especially helpful for cross-document coreference resolution systems (Haghighi and Klein, 2010), which actually represent concepts and track mentions of them across documents.

3 Background: Never-Ending Language Learner

ConceptResolver is designed as a component for the Never-Ending Language Learner (NELL) (Carlson et al., 2010). In this section, we provide some pertinent background information about NELL that influenced the design of ConceptResolver [1].

NELL is an information extraction system that has been running 24x7 for over a year, using coupled semi-supervised learning to populate an ontology from unstructured text found on the web. The ontology defines two types of predicates: *categories* (e.g., **company** and **CEO**) and *relations* (e.g., **ceoOfCompany**). Categories are single-argument predicates, and relations are two-argument predicates.

[1] More information about NELL, including browsable and downloadable versions of its knowledge base, is available from `http://rtw.ml.cmu.edu`.

NELL's *knowledge base* contains both definitions for predicates and extracted instances of each predicate. At present, NELL's knowledge base defines approximately 500 predicates and contains over half a million extracted instances of these predicates with an accuracy of approximately 0.85.

Relations between predicates are an important component of NELL's ontology. For ConceptResolver, the most important relations are **domain** and **range**, which define argument types for each relation predicate. For example, the first argument of **ceoOfCompany** must be a **CEO** and the second argument must be a **company**. Argument type restrictions inform ConceptResolver's word sense induction process (Section 4.1).

Multiple sources of information are used to populate each predicate with high precision. The system runs four independent extractors for each predicate: the first uses web co-occurrence statistics, the second uses HTML structures on webpages, the third uses the morphological structure of the noun phrase itself, and the fourth exploits empirical regularities within the knowledge base. These subcomponents are described in more detail by Carlson et al. (2010) and Wang and Cohen (2007). NELL learns using a bootstrapping process, iteratively re-training these extractors using instances in the knowledge base, then adding some predictions of the learners to the knowledge base. This iterative learning process can be viewed as a discrete approximation to EM which does not explicitly instantiate every latent variable.

As in other information extraction systems, the category and relation instances extracted by NELL contain polysemous and synonymous noun phrases. ConceptResolver was developed to reduce the impact of these phenomena.

4 ConceptResolver

This section describes ConceptResolver, our new component which creates concepts from NELL's extractions. It uses a two-step procedure, first creating one or more senses for each noun phrase, then clustering synonymous senses to create concepts.

4.1 Word Sense Induction

ConceptResolver induces word senses using a simple assumption about noun phrases and concepts. If a noun phrase has multiple senses, the senses should be distinguishable from context. People can determine the sense of an ambiguous word given just a few surrounding words (Kaplan, 1955). We hypothesize that local context enables sense disambiguation by defining the semantic type of the ambiguous word. ConceptResolver makes the simplifying assumption that *all* word senses can be distinguished on the basis of semantic type. As the category predicates in NELL's ontology define a set of possible semantic types, this assumption is equivalent to the *one-sense-per-category assumption*: a noun phrase refers to at most one concept in each category of NELL's ontology. For example, this means that a noun phrase can refer to a **company** and a **fruit**, but not multiple companies.

ConceptResolver uses the extracted category assertions to define word senses. Each word sense is represented as a tuple containing a noun phrase and a category. In synonym resolution, the category acts like a type constraint, and only senses with the same category type can be synonymous. To create senses, the system interprets each extracted category predicate $c(x)$ as evidence that category c contains a concept denoted by noun phrase x. Because it assumes that there is at most one such concept, ConceptResolver creates one sense of x for each extracted category predicate. As a concrete example, say the input assertions contain **company**("apple") and **fruit**("apple"). Sense induction creates two senses for "apple": ("apple", company) and ("apple", fruit).

The second step of sense induction produces evidence for synonym resolution by creating relations between word senses. These relations are created from input relations and the ontology's argument type constraints. Each extracted relation is mapped to all possible sense relations that satisfy the argument type constraints. For example, the noun phrase relation **ceoOfCompany**("steve jobs", "apple") would map to **ceoOfCompany**(("steve jobs", ceo), ("apple", company)). It would not map to a similar relation with ("apple", fruit), however, as ("apple", fruit) is not in the range of **ceoOfCompany**. This process is effective because the relations in the ontology have restrictive domains and ranges, so only a small fraction of sense pairs satisfy the argument type restrictions. It is also not vital that this mapping be perfect, as the sense relations are only

used as evidence for synonym resolution. The final output of sense induction is a sense-disambiguated knowledge base, where each noun phrase has been converted into one or more word senses, and relations hold between pairs of senses.

4.2 Synonym Resolution

After mapping each noun phrase to one or more senses (each with a distinct category type), ConceptResolver performs semi-supervised clustering to find synonymous senses. As only senses with the same category type can be synonymous, our synonym resolution algorithm treats senses of each type independently. For each category, ConceptResolver trains a semi-supervised synonym classifier then uses its predictions to cluster word senses.

Our key insight is that semantic relations and string attributes provide independent views of the data: we can predict that two noun phrases are synonymous either based on the similarity of their text strings, or based on similarity in the relations NELL has extracted about them. As a concrete example, we can decide that ("apple computer", company) and ("apple", company) are synonymous because the text string "apple" is similar to "apple computer," or because we have learned that ("steve jobs", ceo) is the CEO of both companies. ConceptResolver exploits these two independent views using co-training (Blum and Mitchell, 1998) to produce an accurate synonym classifier using only a handful of labels.

4.2.1 Co-Training the Synonym Classifier

For each category, ConceptResolver co-trains a pair of synonym classifiers using a handful of labeled synonymous senses and a large number of automatically created unlabeled sense pairs. Co-training is a semi-supervised learning algorithm for data sets where each instance can be classified from two (or more) independent sets of features. That is, the features of each instance x_i can be partitioned into two views, $x_i = (x_i^1, x_i^2)$, and there exist functions in each view, f^1, f^2, such that $f^1(x_i^1) = f^2(x_i^2) = y_i$. The co-training algorithm uses a bootstrapping procedure to train f^1, f^2 using a small set of labeled examples L and a large pool of unlabeled examples U. The training process repeatedly trains each classifier on the labeled examples, then allows each classifier to label some examples in the unlabeled data pool. Co-training also has PAC-style theoretical guarantees which show that it can learn classifiers with arbitrarily high accuracy under appropriate conditions (Blum and Mitchell, 1998).

Figure 4 provides high-level pseudocode for co-training in the context of ConceptResolver. In ConceptResolver, an instance x_i is a pair of senses (e.g., <("apple", company), ("microsoft", company)>), the two views x_i^1 and x_i^2 are derived from string attributes and semantic relations, and the output y_i is whether the senses are synonyms. (The features of each view are described later in this section.) L is initialized with a small number of labeled sense pairs. Ideally, U would contain all pairs of senses in the category, but this set grows quadratically in category size. Therefore, ConceptResolver uses the canopies algorithm (McCallum et al., 2000) to initialize U with a subset of the sense pairs that are more likely to be synonymous.

Both the string similarity classifier and the relation classifier are trained using L_2-regularized logistic regression. The regularization parameter λ is automatically selected on each iteration by searching for a value which maximizes the loglikelihood of a validation set, which is constructed by randomly sampling 25% of L on each iteration. λ is re-selected on each iteration because the initial labeled data set is extremely small, so the initial validation set is not necessarily representative of the actual data. In our experiments, the initial validation set contains only 15 instances.

The string similarity classifier bases its decision on the original noun phrase which mapped to each sense. We use several string similarity measures as features, including SoftTFIDF (Cohen et al., 2003), Level 2 JaroWinkler (Cohen et al., 2003), Fellegi-Sunter (Fellegi and Sunter, 1969), and Monge-Elkan (Monge and Elkan, 1996). The first three algorithms produce similarity scores by matching words in the two phrases and the fourth is an edit distance. We also use a heuristic abbreviation detection algorithm (Schwartz and Hearst, 2003) and convert its output into a score by dividing the length of the detected abbreviation by the total length of the string.

The relation classifier's features capture several intuitive ways to determine that two items are synonyms from the items they are related to. The relation view contains three features for each relation

For each category C:

1. Initialize labeled data L with 10 positive and 50 negative examples (pairs of senses)
2. Initialize unlabeled data U by running canopies (McCallum et al., 2000) on all senses in C.
3. Repeat 50 times:
   i. Train the string similarity classifier on L
   ii. Train the relation classifier on L
   iii. Label U with each classifier
   iv. Add the most confident 5 positive and 25 negative predictions of both classifiers to L

Figure 4: The co-training algorithm for learning synonym classifiers.

r whose domain is compatible with the current category. Consider the sense pair (s, t), and let $r(s)$ denote s's values for relation r (i.e., $r(s) = \{v : r(s, v)\}$). For each relation r, we instantiate the following features:

- **(Senses which share values are synonyms)** The percent of values of r shared by both s and t, that is $\frac{|r(s) \cap r(t)|}{|r(s) \cup r(t)|}$.
- **(Senses with different values are not synonyms)** The percent of values of r not shared by s and t, or $1 - \frac{|r(s) \cap r(t)|}{|r(s) \cup r(t)|}$. The feature is set to 0 if either $r(s)$ or $r(t)$ is empty. This feature is only instantiated if the ontology specifies that r has at most one value per sense.
- **(Some relations indicate synonymy)** A boolean feature which is true if $t \in r(s)$ or $s \in r(t)$.

The output of co-training is a pair of classifiers for each category. We combine their predictions using the assumption that the two views X^1, X^2 are conditionally independent given Y. As we trained both classifiers using logistic regression, we have models for the probabilities $P(Y|X^1)$ and $P(Y|X^2)$. The conditional independence assumption implies that we can combine their predictions using the formula:

$$P(Y = 1|X^1, X^2) = \frac{P(Y = 1|X^1)P(Y = 1|X^2)P(Y = 0)}{\sum_{y=0,1} P(Y = y|X^1)P(Y = y|X^2)(1 - P(Y = y))}$$

The above formula involves a prior term, $P(Y)$, because the underlying classifiers are discriminative. We set $P(Y = 1) = .5$ in our experiments as this setting reduces our dependence on the (typically poorly calibrated) probability estimates of logistic regression. We also limited the probability predictions of each classifier to lie in $[.01, .99]$ to avoid divide-by-zero errors. The probability $P(Y|X^1, X^2)$ is the final synonym classifier which is used for agglomerative clustering.

4.2.2 Agglomerative Clustering

The second step of our algorithm runs agglomerative clustering to enforce transitivity constraints on the predictions of the co-trained synonym classifier. As noted in previous works (Snow et al., 2006), synonymy is a transitive relation. If a and b are synonyms, and b and c are synonyms, then a and c must also be synonyms. Unfortunately, co-training is not guaranteed to learn a function that satisfies these transitivity constraints. We enforce the constraints by running agglomerative clustering, as clusterings of instances trivially satisfy the transitivity property.

ConceptResolver uses the clustering algorithm described by Snow et al. (2006), which defines a probabilistic model for clustering and a procedure to (locally) maximize the likelihood of the final clustering. The algorithm is essentially bottom-up agglomerative clustering of word senses using a similarity score derived from $P(Y|X^1, X^2)$. The similarity score for two senses is defined as:

$$\log \frac{P(Y = 0)P(Y = 1|X^1, X^2)}{P(Y = 1)P(Y = 0|X^1, X^2)}$$

The similarity score for two clusters is the sum of the similarity scores for all pairs of senses. The agglomerative clustering algorithm iteratively merges the two most similar clusters, stopping when the score of the best possible pair is below 0. The clusters of word senses produced by this process are the concepts for each category.

5 Evaluation

We perform several experiments to measure ConceptResolver's performance at each of its respective tasks. The first experiment evaluates word sense induction using Freebase as a canonical set of concepts. The second experiment evaluates synonym resolution by comparing ConceptResolver's sense clusters to a gold standard clustering.

For both experiments, we used a knowledge base created by running 140 iterations of NELL. We preprocessed this knowledge base by removing all noun

phrases with zero extracted relations. As ConceptResolver treats the instances of each category predicate independently, we chose 7 categories from NELL's ontology to use in the evaluation. The categories were selected on the basis of the number of extracted relations that ConceptResolver could use to detect synonyms. The number of noun phrases in each category is shown in Table 2. We manually labeled 10 pairs of synonymous senses for each of these categories. The system automatically synthesized 50 negative examples from the positive examples by assuming each pair represents a distinct concept, so senses in different pairs are not synonyms.

5.1 Word Sense Induction Evaluation

Our first experiment evaluates the performance of ConceptResolver's category-based word sense induction. We estimate two quantities: (1) *sense precision*, the fraction of senses created by our system that correspond to real-world entities, and (2) *sense recall*, the fraction of real-world entities that ConceptResolver creates senses for. Sense recall is only measured over entities which are represented by a noun phrase in ConceptResolver's input assertions – it is a measure of ConceptResolver's ability to create senses for the noun phrases it is given. Sense precision is directly determined by how frequently NELL's extractors propose correct senses for noun phrases, while sense recall is related to the correctness of the one-sense-per-category assumption.

Precision and recall were evaluated by comparing the senses created by ConceptResolver to concepts in Freebase (Bollacker et al., 2008). We sampled 100 noun phrases from each category and matched each noun phrase to a set of Freebase concepts. We interpret each matching Freebase concept as a sense of the noun phrase. We chose Freebase because it had good coverage for our evaluation categories.

To align ConceptResolver's senses with Freebase, we first matched each of our categories with a set of similar Freebase categories[2]. We then used a combination of Freebase's search API and Mechanical Turk to align noun phrases with Freebase concepts: we searched for the noun phrase in Freebase, then had Mechanical Turk workers label which of the top 10 resulting Freebase concepts the noun phrase could refer to. After obtaining the list of matching Freebase concepts for each noun phrase, we computed sense precision as the number of noun phrases matching ≥ 1 Freebase concept divided by 100, the total number of noun phrases. Sense recall is the reciprocal of the average number of Freebase concepts per noun phrase. Noun phrases matching 0 Freebase concepts were not included in this computation.

Category	Precision	Recall	Freebase concepts per Phrase
athlete	0.95	0.56	1.76
city	0.97	0.25	3.86
coach	0.86	0.94	1.06
company	0.85	0.41	2.41
country	0.74	0.56	1.77
sportsteam	0.89	0.30	3.28
stadiumoreventvenue	0.83	0.61	1.63

Table 1: ConceptResolver's word sense induction performance

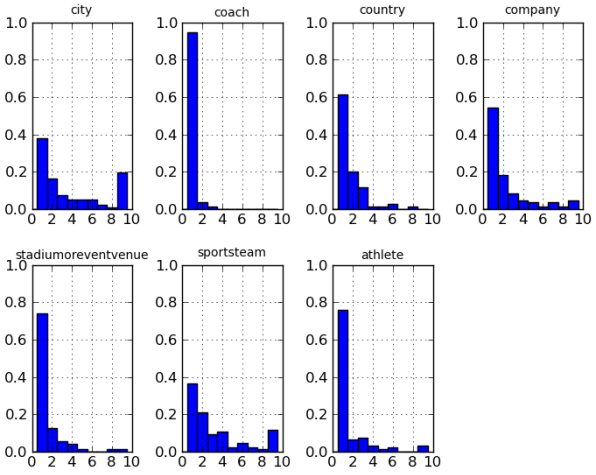


Figure 5: Empirical distribution of the number of Freebase concepts per noun phrase in each category

The results of the evaluation in Table 1 show that ConceptResolver's word sense induction works quite well for many categories. Most categories have high precision, while recall varies by category. Categories like **coach** are relatively unambiguous, with almost exactly 1 sense per noun phrase. Other categories have almost 4 senses per noun phrase. However, this average is somewhat misleading. Figure 5 shows the distribution of the number of concepts per noun phrase in each category. The distribution shows that most noun phrases are unambiguous, but a small number of noun phrases have a large number of senses. In many cases, these noun phrases

[2]In Freebase, concepts are called Topics and categories are called Types. For clarity, we use our terminology throughout.

are generic terms for many items in the category; for example, "palace" in **stadiumoreventvenue** refers to 10 Freebase concepts. Freebase's category definitions are also overly technical in some cases – for example, Freebase's version of **company** has a concept for each registered corporation. This definition means that some companies like *Volkswagen* have more than one concept (in this case, 9 concepts). These results suggest that the one-sense-per-category assumption holds for most noun phrases.

An important footnote to this evaluation is that the categories in NELL's ontology are somewhat arbitrary, and that creating subcategories would improve sense recall. For example, we could define subcategories of **sportsteam** for various sports (e.g., **football team**); these new categories would allow ConceptResolver to distinguish between teams with the same name that play different sports. Creating subcategories could improve performance in categories with a high level of polysemy.

5.2 Synonym Resolution Evaluation

Our second experiment evaluates synonym resolution by comparing the concepts created by ConceptResolver to a gold standard set of concepts. Although this experiment is mainly designed to evaluate ConceptResolver's ability to detect synonyms, it is somewhat affected by the word sense induction process. Specifically, the gold standard clustering contains noun phrases that refer to multiple concepts within the same category. (It is unclear how to create a gold standard clustering without allowing such mappings.) The word sense induction process produces only one of these mappings, which limits maximum possible recall in this experiment.

For this experiment, we report two different measures of clustering performance. The first measure is the precision and recall of pairwise synonym decisions, typically known as cluster precision and recall. We dub this the clustering metric. We also adopt the precision/recall measure from Resolver (Yates and Etzioni, 2007), which we dub the Resolver metric. The Resolver metric aligns each proposed cluster containing $\geq$ 2 senses with a gold standard cluster (i.e., a real-world concept) by selecting the cluster that a plurality of the senses in the proposed cluster refer to. Precision is then the fraction of senses in the proposed cluster which are also in the gold standard cluster; recall is computed analogously by swapping the roles of the proposed and gold standard clusters. Resolver precision can be interpreted as the probability that a randomly sampled sense (in a cluster with at least 2 senses) is in a cluster representing its true meaning. Incorrect senses were removed from the data set before evaluating precision; however, these senses may still affect performance by influencing the clustering process.

Precision was evaluated by sampling 100 random concepts proposed by ConceptResolver, then manually scoring each concept using both of the metrics above. This process mimics aligning each sampled concept with its best possible match in a gold standard clustering, then measuring precision with respect to the gold standard.

Recall was evaluated by comparing the system's output to a manually constructed set of concepts for each category. To create this set, we randomly sampled noun phrases from each category and manually matched each noun phrase to one or more real-world entities. We then found other noun phrases which referred to each entity and created a concept for each entity with at least one unambiguous reference. This process can create multiple senses for a noun phrase, depending on the real-world entities represented in the input assertions. We only included concepts containing at least 2 senses in the test set, as singleton concepts do not contribute to either recall metric. The size of each recall test set is listed in Table 2; we created smaller test sets for categories where synonyms were harder to find. Incorrectly categorized noun phrases were not included in the gold standard as they do not correspond to any real-world entities.

Table 2 shows the performance of ConceptResolver on each evaluation category. For each category, we also report the baseline recall achieved by placing each sense in its own cluster. ConceptResolver has high precision for several of the categories. Other categories like **athlete** and **city** have somewhat lower precision. To make this difference concrete, Figure 2 (first page) shows a random sample of 10 concepts from both **company** and **athlete**. Recall varies even more widely across categories, partly because the categories have varying levels of polysemy, and partly due to differences in average concept size. The differences in average concept size are reflected in the baseline recall numbers.

Category	# of Phrases	Recall Set Size	Resolver Metric				Clustering Metric			
			Precision	Recall	F1	Baseline Recall	Precision	Recall	F1	Baseline Recall
athlete	3886	80	0.69	0.69	0.69	0.46	0.41	0.45	0.43	0.00
city	5710	50	0.66	0.52	0.58	0.42	0.30	0.10	0.15	0.00
coach	889	60	0.90	0.93	0.91	0.43	0.83	0.88	0.85	0.00
company	3553	60	0.93	0.71	0.81	0.39	0.79	0.44	0.57	0.00
country	693	60	0.98	0.50	0.66	0.30	0.94	0.15	0.26	0.00
sportsteam	2085	100	0.95	0.48	0.64	0.29	0.87	0.15	0.26	0.00
stadiumoreventvenue	1662	100	0.84	0.73	0.78	0.39	0.65	0.49	0.56	0.00

Table 2: Synonym resolution performance of ConceptResolver

We attribute the differences in precision across categories to the different relations available for each category. For example, none of the relations for **athlete** uniquely identify a single athlete, and therefore synonymy cannot be accurately represented in the relation view. Adding more relations to NELL's ontology may improve performance in these cases.

We note that the synonym resolution portion of ConceptResolver is tuned for precision, and that perfect recall is not necessarily attainable. Many word senses participate in only one relation, which may not provide enough evidence to detect synonymy. As NELL continually extracts more knowledge, it is reasonable for ConceptResolver to abstain from these decisions until more evidence is available.

6 Discussion

In order for information extraction systems to accurately represent knowledge, they must represent noun phrases, concepts, and the many-to-many mapping from noun phrases to concepts they denote. We present ConceptResolver, a system which takes extracted relations between noun phrases and identifies latent concepts that the noun phrases refer to. Two lessons from ConceptResolver are that (1) ontologies aid word sense induction, as the senses of polysemous words tend to have distinct semantic types, and (2) redundant information, in the form of string similarity and extracted relations, helps train accurate synonym classifiers.

An interesting aspect of ConceptResolver is that its performance should improve as NELL's ontology and knowledge base grow in size. Defining finer-grained categories will improve performance at word sense induction, as more precise categories will contain fewer ambiguous noun phrases. Both extracting more relation instances and adding new relations to the ontology will improve synonym resolution. These scaling properties allow manual effort to be spent on high-level ontology operations, not on labeling individual instances. We are interested in observing ConceptResolver's performance as NELL's ontology and knowledge base grow.

For simplicity of exposition, we have implicitly assumed thus far that the categories in NELL's ontology are mutually exclusive. However, the ontology contains compatible categories like **male** and **politician**, where a single concept can belong to both categories. In these situations, the one-sense-per-category assumption may create too many word senses. We currently address this problem with a heuristic post-processing step: we merge all pairs of concepts that belong to compatible categories and share at least one referring noun phrase. This heuristic typically works well, however there are problems. An example of a problematic case is "obama," which NELL believes is a **male**, **female**, and **politician**. In this case, the heuristic cannot decide which "obama" (the male or female) is the politician. As such cases are fairly rare, we have not developed a more sophisticated solution to this problem.

ConceptResolver has been integrated into NELL's continual learning process. NELL's current set of concepts can be viewed through the knowledge base browser on NELL's website, `http://rtw.ml.cmu.edu`.

Acknowledgments

This work is supported in part by DARPA (under contract numbers FA8750-08-1-0009 and AF8750-09-C-0179) and by Google. We also gratefully acknowledge the contributions of our colleagues on the NELL project, Jamie Callan for the ClueWeb09 web crawl and Yahoo! for use of their M45 computing cluster. Finally, we thank the anonymous reviewers for their helpful comments.

References

Eneko Agirre and Aitor Soroa. 2007. Semeval-2007 task 02: Evaluating word sense induction and discrimination systems. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 7–12.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 2670–2676.

Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. 2004. A probabilistic framework for semi-supervised clustering. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 59–68.

Indrajit Bhattacharya and Lise Getoor. 2006. A latent dirichlet model for unsupervised entity resolution. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pages 47–58.

Indrajit Bhattacharya and Lise Getoor. 2007. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, 1(1).

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 92–100.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*.

William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. 2003. A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proceedings of the IJCAI-03 Workshop on Information Integration*, pages 73–78, August.

Ivan P. Fellegi and Alan B. Sunter. 1969. A theory for record linkage. *Journal of the American Statistical Association*, 64:1183–1210.

Lise Getoor and Christopher P. Diehl. 2005. Link mining: a survey. *SIGKDD Explorations Newsletter*, 7:3–12.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 385–393, June.

Abraham Kaplan. 1955. An experimental study of ambiguity and context. *Mechanical Translation*, 2:39–46.

Dan Klein, Sepandar D. Kamvar, and Christopher D. Manning. 2002. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 307–314.

Dekang Lin and Patrick Pantel. 2002. Concept discovery from text. In *Proceedings of the 19th International Conference on Computational linguistics - Volume 1*, pages 1–7.

Suresh Manandhar, Ioannis P. Klapaftis, Dmitriy Dligach, and Sameer S. Pradhan. 2010. Semeval-2010 task 14: Word sense induction & disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 63–68.

Andrew McCallum and Ben Wellner. 2004. Conditional models of identity uncertainty with application to noun coreference. In *Advances in Neural Information Processing Systems 18*.

Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 169–178.

George A. Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38:39–41.

Alvaro Monge and Charles Elkan. 1996. The field matching problem: Algorithms and applications. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 267–270.

Vincent Ng. 2008. Unsupervised models for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 640–649.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 613–619.

Hoifung Poon and Pedro Domingos. 2007. Joint inference in information extraction. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence - Volume 1*, pages 913–918.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with markov logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 650–659.

Pradeep Ravikumar and William W. Cohen. 2004. A hierarchical graphical model for record linkage. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 454–461.

Ariel S. Schwartz and Marti A. Hearst. 2003. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Proceedings of the Pacific Symposium on BIOCOMPUTING 2003*, pages 451–462.

Parag Singla and Pedro Domingos. 2006. Entity resolution with markov logic. In *Proceedings of the Sixth International Conference on Data Mining*, pages 572–582.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 801–808, Morristown, NJ, USA.

Rion Snow, Sushant Prakash, Daniel Jurafsky, and Andrew Y. Ng. 2007. Learning to merge word senses. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1005–1014, June.

Richard C. Wang and William W. Cohen. 2007. Language-independent set expansion of named entities using the web. In *Proceedings of the Seventh IEEE International Conference on Data Mining*, pages 342–350.

William E. Winkler. 1999. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau.

Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. 2003. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 17*, pages 505–512.

Alexander Yates and Oren Etzioni. 2007. Unsupervised resolution of objects and relations on the web. In *Proceedings of the 2007 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Semantic Representation of Negation Using Focus Detection

Eduardo Blanco and **Dan Moldovan**
Human Language Technology Research Institute
The University of Texas at Dallas
Richardson, TX 75080 USA
{eduardo,moldovan}@hlt.utdallas.edu

Abstract

Negation is present in all human languages and it is used to reverse the polarity of part of statements that are otherwise affirmative by default. A negated statement often carries positive implicit meaning, but to pinpoint the positive part from the negative part is rather difficult. This paper aims at thoroughly representing the semantics of negation by revealing implicit positive meaning. The proposed representation relies on focus of negation detection. For this, new annotation over PropBank and a learning algorithm are proposed.

1 Introduction

Understanding the meaning of text is a long term goal in the natural language processing community. Whereas philosophers and linguists have proposed several theories, along with models to represent the meaning of text, the field of computational linguistics is still far from doing this automatically. The ambiguity of language, the need to detect implicit knowledge, and the demand for commonsense knowledge and reasoning are a few of the difficulties to overcome. Substantial progress has been made, though, especially on detection of semantic relations, ontologies and reasoning methods.

Negation is present in all languages and it is always the case that statements are affirmative by default. Negation is marked and it typically signals something unusual or an exception. It may be present in all units of language, e.g., words (*in*credible), clauses (*He does*n't *have friends*). Negation and its correlates (truth values, lying, irony, false or contradictory statements) are exclusive characteristics of humans (Horn, 1989; Horn and Kato, 2000).

Negation is fairly well-understood in grammars; the valid ways to express a negation are documented. However, there has not been extensive research on detecting it, and more importantly, on representing the semantics of negation. Negation has been largely ignored within the area of semantic relations.

At first glance, one would think that interpreting negation could be reduced to finding negative keywords, detect their scope using syntactic analysis and reverse its polarity. Actually, it is more complex. Negation plays a remarkable role in text understanding and it poses considerable challenges.

Detecting the scope of negation in itself is challenging: *All vegetarians do not eat meat* means that vegetarians do not eat meat and yet *All that glitters is not gold* means that it is not the case that all that glitters is gold (so out of all things that glitter, some are gold and some are not). In the former example, the universal quantifier *all* has scope over the negation; in the latter, the negation has scope over *all*.

In logic, two negatives always cancel each other out. On the other hand, in language this is only theoretically the case: *she is not unhappy* does not mean that *she is happy*; it means that *she is not fully unhappy, but she is not happy either*.

Some negated statements carry a positive implicit meaning. For example, *cows do not eat meat* implies that *cows eat something other than meat*. Otherwise, the speaker would have stated *cows do not eat*. A clearer example is the correct and yet puzzling statement *tables do not eat meat*. This sentence sounds

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 581–589,
Portland, Oregon, June 19-24, 2011.

unnatural because of the underlying positive statement (i.e., *tables eat something other than meat*).

Negation can express *less than* or *in between* when used in a scalar context. For example, *John does not have three children* probably means that he has either one or two children. Contrasts may use negation to disagree about a statement and not to negate it, e.g., *That place is not big, it is massive* defines the place as *massive*, and therefore, *big*.

2 Related Work

Negation has been widely studied outside of computational linguistics. In logic, negation is usually the simplest unary operator and it reverses the truth value. The seminal work by Horn (1989) presents the main thoughts in philosophy and psychology. Linguists have found negation a complex phenomenon; Huddleston and Pullum (2002) dedicate over 60 pages to it. Negation interacts with quantifiers and anaphora (Hintikka, 2002), and influences reasoning (Dowty, 1994; Sánchez Valencia, 1991). Zeijlstra (2007) analyzes the position and form of negative elements and negative concords.

Rooth (1985) presented a theory of focus in his dissertation and posterior publications (e.g., Rooth (1992)). In this paper, we follow the insights on scope and focus of negation by Huddleston and Pullum (2002) rather than Rooth's (1985).

Within natural language processing, negation has drawn attention mainly in sentiment analysis (Wilson et al., 2009; Wiegand et al., 2010) and the biomedical domain. Recently, the Negation and Speculation in NLP Workshop (Morante and Sporleder, 2010) and the CoNLL-2010 Shared Task (Farkas et al., 2010) targeted negation mostly on those subfields. Morante and Daelemans (2009) and Özgür and Radev (2009) propose scope detectors using the BioScope corpus. Councill et al. (2010) present a supervised scope detector using their own annotation. Some NLP applications deal indirectly with negation, e.g., machine translation (van Munster, 1988), text classification (Rose et al., 2003) and recognizing entailments (Bos and Markert, 2005).

Regarding corpora, the BioScope corpus annotates negation marks and linguistic scopes exclusively on biomedical texts. It does not annotate focus and it purposely ignores negations such as *(talking about the reaction of certain elements) in NK3.3 cells is not always identical* (Vincze et al., 2008), which carry the kind of positive meaning this work aims at extracting (in NK3.3 cells *is often* identical). PropBank (Palmer et al., 2005) only indicates the verb to which a negation mark attaches; it does not provide any information about the scope or focus. FrameNet (Baker et al., 1998) does not consider negation and FactBank (Saurí and Pustejovsky, 2009) only annotates degrees of factuality for events.

None of the above references aim at detecting or annotating the focus of negation in natural language. Neither do they aim at carefully representing the meaning of negated statements nor extracting implicit positive meaning from them.

3 Negation in Natural Language

Simply put, negation is a process that turns a statement into its opposite. Unlike affirmative statements, negation is marked by words (e.g., *not, no, never*) or affixes (e.g., *-n't, un-*). Negation can interact with other words in special ways. For example, negated clauses use different connective adjuncts that positive clauses do: *neither, nor* instead of *either, or*. The so-called *negatively-oriented polarity-sensitive items* (Huddleston and Pullum, 2002) include, among many others, words starting with *any-* (*anybody*, *anyone*, *anywhere*, etc.), the modal auxiliaries *dare* and *need* and the grammatical units *at all*, *much* and *till*. Negation in verbs usually requires an auxiliary; if none is present, the auxiliary *do* is inserted (*I read the paper* vs. *I didn't read the paper*).

3.1 Meaning of Negated Statements

State-of-the-art semantic role labelers (e.g., the ones trained over PropBank) do not completely represent the meaning of negated statements. Given *John didn't build a house to impress Mary*, they encode AGENT(*John, build*), THEME(*a house, build*), PURPOSE(*to impress Mary, build*), NEGATION(*n't, build*). This representation corresponds to the interpretation *it is not the case that John built a house to impress Mary*, ignoring that it is implicitly stated that *John did build a house*.

Several examples are shown Table 1. For all statements *s*, current role labelers would only encode *it is not the case that* s. However, examples (1–7)

	Statement	Interpretation
1	John didn't build a house to impress Mary.	John built a house for other purpose.
2	I don't have a watch with me.	I have a watch, but it is not with me.
3	We don't have an evacuation plan for flooding.	We have an evacuation plan for something else (e.g., fire).
4	They didn't release the UFO files until 2008.	They released the UFO files in 2008.
5	John doesn't know exactly how they met.	John knows how they met, but not exactly.
6	His new job doesn't require driving.	His new job has requirements, but it does not require driving.
7	His new job doesn't require driving yet.	His new job requires driving in the future.
8	His new job doesn't require anything.	His new job has no requirements.
9	A panic on Wall Street doesn't exactly inspire confidence.	A panic on Wall Streen discourages confidence.

Table 1: Examples of negated statements and their interpretations considering underlying positive meaning. A wavy underline indicates the focus of negation (Section 3.3); examples (8, 9) do not carry any positive meaning.

carry positive meaning underneath the direct meaning. Regarding (4), encoding that the UFO files *were released in 2008* is crucial to fully interpret the statement. (6–8) show that different verb arguments modify the interpretation and even signal the existence of positive meaning. Examples (5, 9) further illustrate the difficulty of the task; they are very similar (both have AGENT, THEME and MANNER) and their interpretation is altogether different. Note that (8, 9) do not carry any positive meaning; even though their interpretations do not contain a verbal negation, the meaning remains negative. Some examples could be interpreted differently depending on the context (Section 4.2.1).

This paper aims at thoroughly representing the semantics of negation by revealing implicit positive meaning. The main contributions are: (1) interpretation of negation using focus detection; (2) focus of negation annotation over all PropBank negated sentences[1]; (3) feature set to detect the focus of negation; and (4) model to semantically represent negation and reveal its underlying positive meaning.

3.2 Negation Types

Huddleston and Pullum (2002) distinguish four contrasts for negation:

- Verbal if the marker of negation is grammatically associated with the verb (*I did not see anything at all*); non-verbal if it is associated with a dependent of the verb (*I saw nothing at all*).
- Analytic if the sole function of the negated mark is to mark negation (*Bill did not go*); synthetic if it has some other function as well (*[Nobody]*$_{\text{AGENT}}$ *went to the meeting*).
- Clausal if the negation yields a negative clause (*She didn't have a large income*); subclausal otherwise (*She had a not inconsiderable income*).
- Ordinary if it indicates that something is not the case, e.g., (1) *She didn't have lunch with my old man: he couldn't make it*; metalinguistic if it does not dispute the truth but rather reformulates a statement, e.g., (2) *She didn't have lunch with your 'old man': she had lunch with your father*. Note that in (1) the lunch never took place, whereas in (2) a lunch did take place.

In this paper, we focus on verbal, analytic, clausal, and both metalinguistic and ordinary negation.

3.3 Scope and Focus

Negation has both scope and focus and they are extremely important to capture its semantics. Scope is the part of the meaning that is negated. Focus is that part of the scope that is most prominently or explicitly negated (Huddleston and Pullum, 2002).

Both concepts are tightly connected. Scope corresponds to all elements any of whose individual falsity would make the negated statement true. Focus is the element of the scope that is *intended* to be interpreted as false to make the overall negative true.

Consider (1) *Cows don't eat meat* and its positive counterpart (2) *Cows eat meat*. The truth conditions of (2) are: (a) somebody eats something; (b) cows are the ones who eat; and (c) meat is what is eaten.

In order for (2) to be true, (a–c) have to be true. And the falsity of any of them is sufficient to make (1) true. In other words, (1) would be true if *nobody eats*, *cows don't eat* or *meat is not eaten*. Therefore, all three statements (a–c) are inside the scope of (1).

The focus is more difficult to identify, especially

[1] Annotation will be available on the author's website

1	AGENT(*the cow, didn't eat*)	THEME(*grass, didn't eat*)	INSTRUMENT(*with a fork, didn't eat*)
2	NOT[AGENT(*the cow, ate*)	THEME(*grass, ate*)	INSTRUMENT(*with a fork, ate*)]
3	NOT[AGENT(*the cow, ate*)]	THEME(*grass, ate*)	INSTRUMENT(*with a fork, ate*)
4	AGENT(*the cow, ate*)	NOT[THEME(*grass, ate*)]	INSTRUMENT(*with a fork, ate*)
5	AGENT(*the cow, ate*)	THEME(*grass, ate*)	NOT[INSTRUMENT(*with a fork, ate*)]

Table 2: Possible semantic representations for *The cow didn't eat grass with a fork*.

without knowing stress or intonation. Text understanding is needed and context plays an important role. The most probable focus for (1) is *meat*, which corresponds to the interpretation *cows eat something else than meat*. Another possible focus is *cows*, which yields *someone eats meat, but not cows*.

Both scope and focus are primarily semantic, highly ambiguous and context-dependent. More examples can be found in Tables 1 and 3 and (Huddleston and Pullum, 2002, Chap. 9).

4 Approach to Semantic Representation of Negation

Negation does not stand on its own. To be useful, it should be added as part of another existing knowledge representation. In this Section, we outline how to incorporate negation into semantic relations.

4.1 Semantic Relations

Semantic relations capture connections between concepts and label them according to their nature. It is out of the scope of this paper to define them in depth, establish a set to consider or discuss their detection. Instead, we use generic semantic roles.

Given *s: The cow didn't eat grass with a fork*, typical semantic roles encode AGENT(*the cow, eat*), THEME(*grass, eat*), INSTRUMENT(*with a fork, eat*) and NEGATION(*n't, eat*). This representation only differs on the last relation from the positive counterpart. Its interpretation is *it is not the case that* s.

Several options arise to thoroughly represent *s*. First, we find it useful to consider the semantic representation of the affirmative counterpart: AGENT(*the cow, ate*), THEME(*grass, ate*), and INSTRUMENT(*with a fork, ate*). Second, we believe detecting the focus of negation is useful. Even though it is open to discussion, the focus corresponds to INSTRUMENT(*with a fork, ate*) Thus, the negated statement should be interpreted as *the cow ate grass, but it did not do so using a fork*.

Table 2 depicts five different possible semantic representations. Option (1) does not incorporate any explicit representation of negation. It attaches the negated mark and auxiliary to *eat*; the negation is part of the relation arguments. This option fails to detect any underlying positive meaning and corresponds to the interpretation *the cow did not eat, grass was not eaten* and *a fork was not used to eat*.

Options (2–5) embody negation into the representation with the *pseudo-relation* NOT. NOT takes as its argument an instantiated relation or set of relations and indicates that they do not hold.

Option (2) includes all the scope as the argument of NOT and corresponds to the interpretation *it is not the case that the cow ate grass with a fork*. Like typical semantic roles, option (2) does not reveal the implicit positive meaning carried by statement *s*. Options (3–5) encode different interpretations:

- (3) negates the AGENT; it corresponds to *the cow didn't eat, but grass was eaten with a fork.*
- (4) applies NOT to the THEME; it corresponds to *the cow ate something with a fork, but not grass*.
- (5) denies the INSTRUMENT, encoding the meaning *the cow ate grass, but it did not use a fork*.

Option (5) is preferred since it captures the best implicit positive meaning. It corresponds to the semantic representation of the affirmative counterpart after applying the pseudo-relation NOT over the focus of the negation. This fact justifies and motivates the detection of the focus of negation.

4.2 Annotating the Focus of Negation

Due to the lack of corpora containing annotation for focus of negation, new annotation is needed. An obvious option is to add it to any text collection. However, building on top of publicly available resources is a better approach: they are known by the community, they contain useful information for detecting the focus of negation and tools have already been developed to predict their annotation.

	Statement	V	A0	A1	A2	A4	TMP	MNR	ADV	LOC	PNC	EXT	DIS	MOD
1	Even if [that deal]$_{A1}$ isn't [revived]$_{V}$, NBC hopes to find another.													
	– Even if that deal is suppressed, NBC hopes to find another one.	⋆	-	+	-	-	-	-	-	-	-	-	-	-
2	[He]$_{A0}$ [simply]$_{MDIS}$ [ca]$_{MMOD}$n't [stomach]$_{V}$ [the taste of Heinz]$_{A1}$, she says.													
	– He simply can stomach any ketchup but Heinz's.	+	+	⋆	-	-	-	-	-	-	-	-	+	+
3	[A decision]$_{A1}$ isn't [expected]$_{V}$ [until some time next year]$_{MTMP}$.													
	– A decision is expected at some time next year.	+	-	+	-	-	⋆	-	-	-	-	-	-	-
4	[...] it told the SEC [it]$_{A0}$ [could]$_{MMOD}$n't [provide]$_{V}$ [financial statements]$_{A1}$ [by the end of its first extension]$_{MTMP}$ "[without unreasonable burden or expense]$_{MMNR}$".													
	– It could provide them by that time with a huge overhead.	+	+	+	-	-	+	⋆	-	-	-	-	-	+
5	[For example]$_{MDIS}$, [P&G]$_{A0}$ [up until now]$_{MTMP}$ hasn't [sold]$_{V}$ [coffee]$_{A1}$ [to airlines]$_{A2}$ and does only limited business with hotels and large restaurant chains.													
	– Up until now, P&G has sold coffee, but not to airlines.	+	+	+	⋆	-	+	-	-	-	-	-	+	-
6	[Decent life ...]$_{A1}$ [wo]$_{MMOD}$n't be [restored]$_{V}$ [unless the government reclaims the streets from the gangs]$_{MADV}$.													
	– It will be restored if the government reclaims the streets from the gangs.	+	-	+	-	-	-	-	⋆	-	-	-	-	+
7	But [quite a few money managers]$_{A0}$ aren't [buying]$_{V}$ [it]$_{A1}$.													
	– Very little managers are buying it.	+	⋆	+	-	-	-	-	-	-	-	-	-	-
8	[When]$_{MTMP}$ [she]$_{A0}$ isn't [performing]$_{V}$ [for an audience]$_{MPNC}$, she prepares for a song by removing the wad of gum from her mouth, and indicates that she's finished by sticking the gum back in.													
	– She prepares in that way when she is performing, but not for an audience.	+	+	-	-	-	+	-	-	-	⋆	-	-	-
9	[The company's net worth]$_{A1}$ [can]$_{MMOD}$not [fall]$_{V}$ [below $185 million]$_{A4}$ [after the dividends are issued]$_{MTMP}$.													
	– It can fall after the dividends are issued, but not below $185 million.	+	-	+	-	⋆	+	-	-	-	-	-	-	+
10	Mario Gabelli, an expert at spotting takeover candidates, says that [takeovers]$_{A1}$ aren't [totally]$_{MEXT}$ [gone]$_{V}$.													
	– Mario Gabelli says that takeovers are partially gone.	+	-	+	-	-	-	-	-	-	⋆	-	-	-

Table 3: Negated statements from PropBank and their interpretation considering underlying positive meaning. Focus is underlined; '+' indicates that the role is present, '-' that it is not and '⋆' that it corresponds to the focus of negation.

We decided to work over PropBank. Unlike other resources (e.g., FrameNet), gold syntactic trees are available. Compared to the BioScope corpus, PropBank provides semantic annotation and is not limited to the biomedical domain. On top of that, there has been active research on predicting PropBank roles for years. The additional annotation can be readily used by any system trained with PropBank, quickly incorporating interpretation of negation.

4.2.1 Annotation Guidelines

The focus of a negation involving verb v is resolved as:

- If it cannot be inferred that an action v occurred, focus is role MNEG.
- Otherwise, focus is the role that is most prominently negated.

All decisions are made considering as context the previous and next sentence. The mark -NOT is used to indicate the focus. Consider the following statement (file wsj_2282, sentence 16).

> [While profitable]$_{MADV_{1,2}}$, [it]$_{A1_1,A0_2}$ "was[n't]$_{MNEG_1}$ [growing]$_{v_1}$ and was[n't]$_{MNEG_2}$ [providing]$_{v_2}$ [a satisfactory return on invested capital]$_{A1_2}$," he says.

The previous sentence is *Applied, then a closely held company, was stagnating under the management of its controlling family*. Regarding the first verb (*growing*), one cannot infer that anything was growing, so focus is MNEG. For the second verb (*providing*), it is implicitly stated that the company was providing *a not satisfactory return on investment*, therefore, focus is A1.

The guidelines assume that the focus corresponds to a single role or the verb. In cases where more than one role could be selected, the most likely focus is chosen; context and text understanding are key. We define the most likely focus as the one that yields the most meaningful implicit information.

For example, in (Table 3, example 2) *[He]*$_{A0}$ could be chosen as focus, yielding *someone can stomach the taste of Heinz, but not him*. However, given the previous sentence (*[...] her husband is*

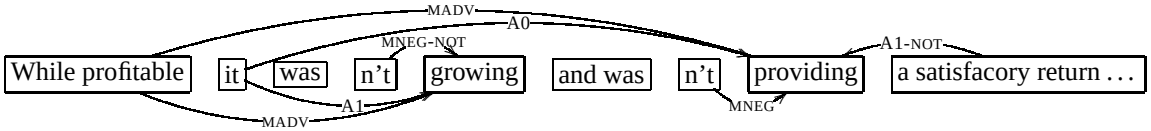


Figure 1: Example of focus annotation (marked with -NOT). Its interpretation is explained in Section 4.2.2.

adamant about eating only Hunt's ketchup), it is clear that the best option is A1. Example (5) has a similar ambiguity between A0 and A2, example (9) between MTMP and A4, etc. The role that yields the most useful positive implicit information given the context is always chosen as focus.

Table 3 provides several examples having as their focus different roles. Example (1) does not carry any positive meaning, the focus is V. In (2–10) the verb must be interpreted as affirmative, as well as all roles except the one marked with '⋆' (i.e., the focus). For each example, we provide PropBank annotation (top), the new annotation (i.e., the focus, bottom right) and its interpretation (bottom left).

4.2.2 Interpretation of -NOT

The mark -NOT is interpreted as follows:

- If MNEG-NOT(x, y), then verb y must be negated; the statement does not carry positive meaning.
- If any other role is marked with -NOT, ROLE-NOT(x, y) must be interpreted as *it is not the case that x is* ROLE *of y*.

Unmarked roles are interpreted positive; they correspond to implicit positive meaning. Role labels (A0, MTMP, etc.) maintain the same meaning from PropBank (Palmer et al., 2005). MNEG can be ignored since it is overwritten by -NOT.

The new annotation for the example (Figure 1) must be interpreted as: *While profitable, it (the company) was not growing and was providing a not satisfactory return on investment*. Paraphrasing, *While profitable, it was shrinking or idle and was providing an unsatisfactory return on investment*. We discover an entailment and an implicature respectively.

4.3 Annotation Process

We annotated the 3,993 verbal negations signaled with MNEG in PropBank. Before annotation began, all semantic information was removed by mapping all role labels to ARG. This step is necessary to ensure that focus selection is not biased by the semantic labels provided by PropBank.

Role	#Inst.	Focus
		# – %
A1	2,930	1,194 – 40.75
MNEG	3,196	1,109 – 34.70
MTMP	609	246 – 40.39
MMNR	250	190 – 76.00
A2	501	179 – 35.73
MADV	466	94 – 20.17
A0	2,163	73 – 3.37
MLOC	114	22 – 19.30
MEXT	25	22 – 88.00
A4	26	22 – 84.62
A3	48	18 – 37.50
MDIR	35	13 – 37.14
MPNC	87	9 – 10.34
MDIS	287	6 – 2.09

Table 4: Roles, total instantiations and counts corresponding to focus over training and held-out instances.

As annotation tool, we use Jubilee (Choi et al., 2010). For each instance, annotators decide the focus given the full syntactic tree, as well as the previous and next sentence. A post-processing step incorporates focus annotation to the original PropBank by adding -NOT to the corresponding role.

In a first round, 50% of instances were annotated twice. Inter-annotator agreement was 0.72. After careful examination of the disagreements, they were resolved and annotators were given clearer instructions. The main point of conflict was selecting a focus that yields valid implicit meaning, but not the most valuable (Section 4.2.1). Due to space constraints, we cannot elaborate more on this issue. The remaining instances were annotated once. Table 4 depicts counts for each role.

5 Learning Algorithm

We propose a supervised learning approach. Each sentence from PropBank containing a verbal negation becomes an instance. The decision to be made is to choose the role that corresponds to the focus.

No.	Feature	Values	Explanation
1	role-present	{y, n}	is role present?
2	role-f-pos	{DT, NNP, ...}	First POS tag of role
3	role-f-word	{This, to, overseas, ...}	First word of role
4	role-length	$\mathbb{N}$	number fo words in role
5	role-posit	$\mathbb{N}$	position within the set of roles
6	A1-top	{NP, SBAR, PP, ...}	syntactic node of A1
7	A1-postag	{y, n}	does A1 contain the tag *postag*?
8	A1-keyword	{y, n}	does A1 cotain the word *keyword*?
9	first-role	{A1, MLOC, ...}	label of the first role
10	last-role	{A1, MLOC, ...}	label of the last role
11	verb-word	{appear, describe, ...}	main verb
12	verb-postag	{VBN, VBZ, ...}	POS tag main verb
13	VP-words	{were-n't, be-quickly, ...}	sequence of words of VP until verb
14	VP-postags	{VBP-RB-RB-VBG, VBN-VBG, ...}	sequence of POS tags of VP until verb
15	VP-has-CC	{y, n}	does the VP contain a CC?
16	VP-has-RB	{y, n}	does the VP contain a RB?
17	predicate	{rule-out, come-up, ...}	predicate
18	them-role-A0	{preparer, assigner, ...}	thematic role for A0
19	them-role-A1	{effort, container, ...}	thematic role for A1
20	them-role-A2	{audience, loaner, ...}	thematic role for A2
21	them-role-A3	{intensifier, collateral, ...}	thematic role for A3
22	them-role-A4	{beneficiary, end point, ...}	thematic role for A4

Table 5: Full set of features. Features (1–5) are extracted for all roles, (7, 8) for all POS tags and keywords detected.

The 3,993 annotated instances are divided into training (70%), held-out (10%) and test (20%). The held-out portion is used to tune the feature set and results are reported for the test split only, i.e., using unseen instances. Because PropBank adds semantic role annotation on top of the Penn TreeBank, we have available syntactic annotation and semantic role labels for all instances.

5.1 Baselines

We implemented four baselines to measure the difficulty of the task:

- A1: select A1, if not present then MNEG.
- FIRST: select first role.
- LAST: select last role.
- BASIC: same than FOC-DET but only using features *last_role* and flags indicating the presence of roles.

5.2 Selecting Features

The BASIC baseline obtains a respectable accuracy of 61.38 (Table 6). Most errors correspond to instances having as focus the two most likely foci: A1 and MNEG (Table 4). We improve BASIC with an extended feature set which targets especially A1 and the verb (Table 5).

Features (1–5) are extracted for each role and capture their presence, first POS tag and word, length and position within the roles present for that instance. Features (6–8) further characterize A1. A1-postag is extracted for the following POS tags: DT, JJ, PRP, CD, RB, VB and WP; A1-keyword for the following words: *any*, *anybody*, *anymore*, *anyone*, *anything*, *anytime*, *anywhere*, *certain*, *enough*, *full*, *many*, *much*, *other*, *some*, *specifics*, *too* and *until*. These lists of POS tags and keywords were extracted after manual examination of training examples and aim at signaling whether this role correspond to the focus. Examples of A1 corresponding to the focus and including one of the POS tags or keywords are:

- *[Apparently]*$_{\text{MADV}}$, *[the respondents]*$_{A0}$ *do n't think [that an economic slowdown would harm the major investment markets very*^{RB} *much]*$_{\text{A1}}$. (i.e., the responders think it would harm the investements little).

- *[The oil company]*$_{\text{A0}}$ *does n't anticipate [any*^{keyword} *additional charges]*$_{\text{A1}}$ (i.e., *the company anticipates no additional charges*).
- *[Money managers and other bond buyers]*$_{\text{A0}}$ *haven't [shown]*$_{\text{V}}$ *[much*^{keyword} *interest in the Refcorp bonds]*$_{\text{A1}}$ (i.e., *they have shown little interest in the bonds*).
- *He concedes H&R Block is well-entrenched and a great company, but says "[it]*$_{\text{A1}}$ *doesn't [grow]*$_{\text{V}}$ *[fast enough*^{keyword} *for us]*$_{\text{A1}}$*"* (i.e., it is growing too slow for us).
- [We]$_{A0}$ don't [see]$_{\text{V}}$ [a domestic source for some^{keyword} of our HDTV requirements]$_{\text{A1}}$, and that's a source of concern [. . .] (i.e., we see a domestic source for some other of our HDTV requirements)

Features (11–16) correspond to the main verb. `VP-words` (`VP-postag`) captures the full sequence of words (POS tags) from the beginning of the VP until the main verb. Features (15–16) check for POS tags as the presence of certain tags usually signal that the verb is not the focus of negation (e.g., *[Thus]*$_{\text{MDIS}}$*, he asserts, [Lloyd's]*$_{\text{A0}}$ *[[ca]*$_{\text{MMOD}}$*n't [react]*$_{v}$ *[quickly*^{RB}*]*$_{MMNR}$ *[to competition]*$_{\text{A1}}$*]*$_{VP}$).

Features (17–22) tackle the predicate, which includes the main verb and may include other words (typically prepositions). We consider the words in the predicate, as well as the specific thematic roles for each numbered argument. This is useful since PropBank uses different numbered arguments for the same thematic role depending on the frame (e.g., A3 is used as PURPOSE in *authorize.01* and as INSTRUMENT in *avert.01*).

6 Experiments and Results

As a learning algorithm, we use bagging with C4.5 decision trees. This combination is fast to train and test, and typically provides good performance. More features than the ones depicted were tried, but we only report the final set. For example, the parent node for all roles was considered and discarded. We name the model considering all features and trained using bagging with C4.5 trees `FOC-DET`.

Results over the test split are depicted in Table 6. Simply choosing A1 as the focus yields an accuracy of 42.11. A better baseline is to always pick the last role (58.39 accuracy). Feeding the learning algorithm exclusively the label corresponding to the last role and flags indicating the presence of roles yields 61.38 accuracy (`BASIC` baseline).

System	Accuracy
`A1`	42.11
`FIRST`	7.00
`LAST`	58.39
`BASIC`	61.38
`FOC-DET`	65.50

Table 6: Accuracies over test split.

Having an agreement of 0.72, there is still room for improvement. The full set of features yields 65.50 accuracy. The difference in accuracy between `BASIC` and `FOC-DET` (4.12) is statistically significant (Z-value $= 1.71$). We test the significance of the difference in performance between two systems i and j on a set of ins instances with the Z-score test, where $z = \frac{abs(err_i, err_j)}{\sigma_d}$, err_k is the error made using set k and $\sigma_d = \sqrt{\frac{err_i(1-err_i)}{ins} + \frac{err_j(1-err_j)}{ins}}$.

7 Conclusions

In this paper, we present a novel way to semantically represent negation using focus detection. Implicit positive meaning is identified, giving a thorough interpretation of negated statements.

Due to the lack of corpora annotating the focus of negation, we have added this information to all the negations marked with MNEG in PropBank. A set of features is depicted and a supervised model proposed. The task is highly ambiguous and semantic features have proven helpful.

A verbal negation is interpreted by considering all roles positive except the one corresponding to the focus. This has proven useful as shown in several examples. In some cases, though, it is not easy to obtain the meaning of a negated role.

Consider (Table 3, example 5) *P&G hasn't sold coffee to airlines*. The proposed representation encodes *P&G has sold coffee, but not to airlines*. However, it is not said that the buyers are likely to have been other kinds of companies. Even without fully identifying the buyer, we believe it is of utmost importance to detect that *P&G has sold coffee*. Empirical data (Table 4) shows that over 65% of negations in PropBank carry implicit positive meaning.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 17th international conference on Computational Linguistics*, Montreal, Canada.

Johan Bos and Katja Markert. 2005. Recognising Textual Entailment with Logical Inference. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 628–635, Vancouver, British Columbia, Canada.

Jinho D. Choi, Claire Bonial, and Martha Palmer. 2010. Propbank Instance Annotation Guidelines Using a Dedicated Editor, Jubilee. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta.

Isaac Councill, Ryan McDonald, and Leonid Velikovich. 2010. What's great and what's not: learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, pages 51–59, Uppsala, Sweden.

David Dowty. 1994. The Role of Negative Polarity and Concord Marking in Natural Language Reasoning. In *Proceedings of Semantics and Linguistics Theory (SALT) 4*, pages 114–144.

Richárd Farkas, Veronika Vincze, György Móra, János Csirik, and György Szarvas. 2010. The CoNLL-2010 Shared Task: Learning to Detect Hedges and their Scope in Natural Language Text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 1–12, Uppsala, Sweden.

Jaakko Hintikka. 2002. Negation in Logic and in Natural Language. *Linguistics and Philosophy*, 25(5/6).

Laurence R. Horn and Yasuhiko Kato, editors. 2000. *Negation and Polarity - Syntactic and Semantic Perspectives (Oxford Linguistics)*. Oxford University Press, USA.

Laurence R. Horn. 1989. *A Natural History of Negation*. University Of Chicago Press.

Rodney D. Huddleston and Geoffrey K. Pullum. 2002. *The Cambridge Grammar of the English Language*. Cambridge University Press.

Roser Morante and Walter Daelemans. 2009. Learning the Scope of Hedge Cues in Biomedical Texts. In *Proceedings of the BioNLP 2009 Workshop*, pages 28–36, Boulder, Colorado.

Roser Morante and Caroline Sporleder, editors. 2010. *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*. University of Antwerp, Uppsala, Sweden.

Arzucan Özgür and Dragomir R. Radev. 2009. Detecting Speculations and their Scopes in Scientific Text. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1398–1407, Singapore.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

Mats Rooth. 1985. *Association with Focus*. Ph.D. thesis, Univeristy of Massachusetts, Amherst.

Mats Rooth. 1992. A Theory of Focus Interpretation. *Natural Language Semantics*, 1:75–116.

Carolyn P. Rose, Antonio Roque, Dumisizwe Bhembe, and Kurt Vanlehn. 2003. A Hybrid Text Classification Approach for Analysis of Student Essays. In *In Building Educational Applications Using Natural Language Processing*, pages 68–75.

Victor Sánchez Valencia. 1991. *Studies on Natural Logic and Categorial Grammar*. Ph.D. thesis, University of Amsterdam.

Roser Saurí and James Pustejovsky. 2009. FactBank: a corpus annotated with event factuality. *Language Resources and Evaluation*, 43(3):227–268.

Elly van Munster. 1988. The treatment of Scope and Negation in Rosetta. In *Proceedings of the 12th International Conference on Computational Linguistics*, Budapest, Hungary.

Veronika Vincze, Gyorgy Szarvas, Richard Farkas, Gyorgy Mora, and Janos Csirik. 2008. The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics*, 9(Suppl 11):S9+.

Michael Wiegand, Alexandra Balahur, Benjamin Roth, Dietrich Klakow, and Andrés Montoyo. 2010. A survey on the role of negation in sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, pages 60–68, Uppsala, Sweden, July.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2009. Recognizing Contextual Polarity: An Exploration of Features for Phrase-Level Sentiment Analysis. *Computational Linguistics*, 35(3):399–433.

H. Zeijlstra. 2007. Negation in Natural Language: On the Form and Meaning of Negative Elements. *Language and Linguistics Compass*, 1(5):498–518.

Learning Dependency-Based Compositional Semantics

Percy Liang
UC Berkeley
pliang@cs.berkeley.edu

Michael I. Jordan
UC Berkeley
jordan@cs.berkeley.edu

Dan Klein
UC Berkeley
klein@cs.berkeley.edu

Abstract

Compositional question answering begins by mapping questions to logical forms, but training a semantic parser to perform this mapping typically requires the costly annotation of the target logical forms. In this paper, we learn to map questions to answers via latent logical forms, which are induced automatically from question-answer pairs. In tackling this challenging learning problem, we introduce a new semantic representation which highlights a parallel between dependency syntax and efficient evaluation of logical forms. On two standard semantic parsing benchmarks (GEO and JOBS), our system obtains the highest published accuracies, despite requiring no annotated logical forms.

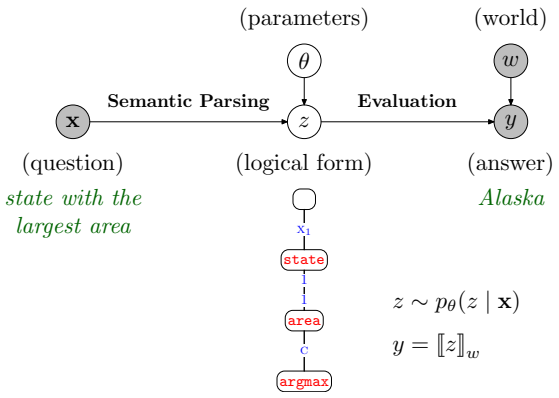


Figure 1: Our probabilistic model: a question $\mathbf{x}$ is mapped to a latent logical form z, which is then evaluated with respect to a world w (database of facts), producing an answer y. We represent logical forms z as labeled trees, induced automatically from $(\mathbf{x}, y)$ pairs.

1 Introduction

What is the total population of the ten largest capitals in the US? Answering these types of complex questions compositionally involves first mapping the questions into logical forms (semantic parsing). Supervised semantic parsers (Zelle and Mooney, 1996; Tang and Mooney, 2001; Ge and Mooney, 2005; Zettlemoyer and Collins, 2005; Kate and Mooney, 2007; Zettlemoyer and Collins, 2007; Wong and Mooney, 2007; Kwiatkowski et al., 2010) rely on manual annotation of logical forms, which is expensive. On the other hand, existing unsupervised semantic parsers (Poon and Domingos, 2009) do not handle deeper linguistic phenomena such as quantification, negation, and superlatives.

As in Clarke et al. (2010), we obviate the need for annotated logical forms by considering the end-to-end problem of mapping questions to answers. However, we still model the logical form (now as a latent variable) to capture the complexities of language. Figure 1 shows our probabilistic model: We want to induce latent logical forms z (and parameters θ) given only question-answer pairs $(\mathbf{x}, y)$, which is much cheaper to obtain than $(\mathbf{x}, z)$ pairs.

The core problem that arises in this setting is program induction: finding a logical form z (over an exponentially large space of possibilities) that produces the target answer y. Unlike standard semantic parsing, our end goal is only to generate the correct y, so we are free to choose the representation for z. Which one should we use?

The dominant paradigm in compositional semantics is Montague semantics, which constructs lambda calculus forms in a bottom-up manner. CCG is one instantiation (Steedman, 2000), which is used by many semantic parsers, e.g., Zettlemoyer and Collins (2005). However, the logical forms there can become quite complex, and in the context of program induction, this would lead to an unwieldy search space. At the same time, representations such as FunQL (Kate et al., 2005), which was used in

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 590–599,
Portland, Oregon, June 19-24, 2011.

Clarke et al. (2010), are simpler but lack the full expressive power of lambda calculus.

The main technical contribution of this work is a new semantic representation, *dependency-based compositional semantics* (DCS), which is both simple and expressive (Section 2). The logical forms in this framework are trees, which is desirable for two reasons: (i) they parallel syntactic dependency trees, which facilitates parsing and learning; and (ii) evaluating them to obtain the answer is computationally efficient.

We trained our model using an EM-like algorithm (Section 3) on two benchmarks, GEO and JOBS (Section 4). Our system outperforms all existing systems despite using no annotated logical forms.

2 Semantic Representation

We first present a basic version (Section 2.1) of dependency-based compositional semantics (DCS), which captures the core idea of using trees to represent formal semantics. We then introduce the full version (Section 2.2), which handles linguistic phenomena such as quantification, where syntactic and semantic scope diverge.

We start with some definitions, using US geography as an example domain. Let $\mathcal{V}$ be the set of all *values*, which includes primitives (e.g., $3, \texttt{CA} \in \mathcal{V}$) as well as sets and tuples formed from other values (e.g., $3, \{3,4,7\}, (\texttt{CA}, \{5\}) \in \mathcal{V}$). Let $\mathcal{P}$ be a set of *predicates* (e.g., $\texttt{state}, \texttt{count} \in \mathcal{P}$), which are just symbols.

A *world* w is mapping from each predicate $p \in \mathcal{P}$ to a set of tuples; for example, $w(\texttt{state}) = \{(\texttt{CA}), (\texttt{OR}), \dots\}$. Conceptually, a world is a relational database where each predicate is a relation (possibly infinite). Define a special predicate $\emptyset$ with $w(\emptyset) = \mathcal{V}$. We represent functions by a set of input-output pairs, e.g., $w(\texttt{count}) = \{(S, n) : n = |S|\}$. As another example, $w(\texttt{average}) = \{(S, \bar{x}) : \bar{x} = |S_1|^{-1} \sum_{x \in S_1} S(x)\}$, where a set of pairs S is treated as a set-valued function $S(x) = \{y : (x, y) \in S\}$ with domain $S_1 = \{x : (x, y) \in S\}$.

The logical forms in DCS are called *DCS trees*, where nodes are labeled with predicates, and edges are labeled with relations. Formally:

Definition 1 (DCS trees) *Let $\mathcal{Z}$ be the set of DCS trees, where each $z \in \mathcal{Z}$ consists of (i) a predicate $z.p \in \mathcal{P}$ and (ii) a sequence of edges $z.e_1, \dots, z.e_m$, each edge e consisting of a relation $e.r \in \mathcal{R}$ (see Table 1) and a child tree $e.c \in \mathcal{Z}$.*

Relations $\mathcal{R}$

$\frac{j}{j'}$	(join)	E	(extract)
Σ	(aggregate)	Q	(quantify)
$\mathrm{X}_{\mathbf{i}}$	(execute)	C	(compare)

Table 1: Possible relations appearing on the edges of a DCS tree. Here, $j, j' \in \{1, 2, \dots\}$ and $\mathbf{i} \in \{1, 2, \dots\}^*$.

We write a DCS tree z as $\langle p; r_1 : c_1; \dots; r_m : c_m \rangle$. Figure 2(a) shows an example of a DCS tree. Although a DCS tree is a logical form, note that it looks like a syntactic dependency tree with predicates in place of words. It is this transparency between syntax and semantics provided by DCS which leads to a simple and streamlined compositional semantics suitable for program induction.

2.1 Basic Version

The basic version of DCS restricts $\mathcal{R}$ to join and aggregate relations (see Table 1). Let us start by considering a DCS tree z with only join relations. Such a z defines a constraint satisfaction problem (CSP) with nodes as variables. The CSP has two types of constraints: (i) $x \in w(p)$ for each node x labeled with predicate $p \in \mathcal{P}$; and (ii) $x_j = y_{j'}$ (the j-th component of x must equal the j'-th component of y) for each edge (x, y) labeled with $\frac{j}{j'} \in \mathcal{R}$.

A solution to the CSP is an assignment of nodes to values that satisfies all the constraints. We say a value v is *consistent* for a node x if there exists a solution that assigns v to x. The *denotation* $[\![z]\!]_w$ (z evaluated on w) is the set of consistent values of the root node (see Figure 2 for an example).

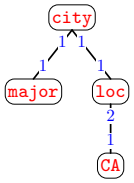
Computation We can compute the denotation $[\![z]\!]_w$ of a DCS tree z by exploiting dynamic programming on trees (Dechter, 2003). The recurrence is as follows:

$$[\![\left\langle p; \tfrac{j_1}{j'_1} : c_1; \cdots; \tfrac{j_m}{j'_m} : c_m \right\rangle]\!]_w = w(p) \cap \bigcap_{i=1}^{m} \{v : v_{j_i} = t_{j'_i}, t \in [\![c_i]\!]_w\}. \quad (1)$$

At each node, we compute the set of tuples v consistent with the predicate at that node ($v \in w(p)$), and

Example: *major city in California*

$z = \langle \texttt{city}; \frac{1}{1}:\langle \texttt{major}\rangle ; \frac{1}{1}:\langle \texttt{loc}; \frac{2}{1}:\langle \texttt{CA}\rangle\rangle\rangle$

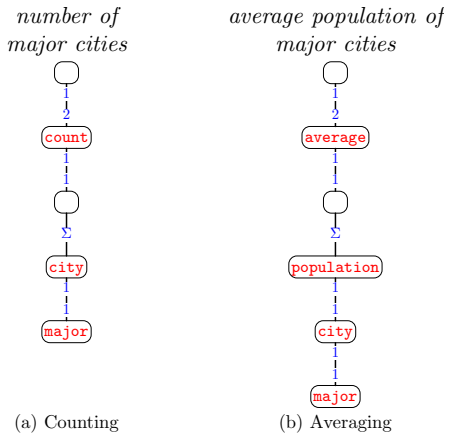


$$\lambda c\, \exists m\, \exists \ell\, \exists s\,.\ \texttt{city}(c) \wedge \texttt{major}(m) \wedge \texttt{loc}(\ell) \wedge \texttt{CA}(s) \wedge c_1 = m_1 \wedge c_1 = \ell_1 \wedge \ell_2 = s_1$$

(a) DCS tree (b) Lambda calculus formula

(c) Denotation: $[\![z]\!]_w = \{\texttt{SF}, \texttt{LA}, \ldots\}$

Figure 2: (a) An example of a DCS tree (written in both the mathematical and graphical notation). Each node is labeled with a predicate, and each edge is labeled with a relation. (b) A DCS tree z with only join relations encodes a constraint satisfaction problem. (c) The denotation of z is the set of consistent values for the root node.



(a) Counting (b) Averaging

Figure 3: Examples of DCS trees that use the aggregate relation (Σ) to (a) compute the cardinality of a set and (b) take the average over a set.

for each child i, the j_i-th component of v must equal the j'_i-th component of some t in the child's denotation ($t \in [\![c_i]\!]_w$). This algorithm is linear in the number of nodes times the size of the denotations.[1]

Now the dual importance of trees in DCS is clear: We have seen that trees parallel syntactic dependency structure, which will facilitate parsing. In addition, trees enable efficient computation, thereby establishing a new connection between dependency syntax and efficient semantic evaluation.

Aggregate relation DCS trees that only use join relations can represent arbitrarily complex compositional structures, but they cannot capture higher-order phenomena in language. For example, consider the phrase *number of major cities*, and suppose that *number* corresponds to the `count` predicate. It is impossible to represent the semantics of this phrase with just a CSP, so we introduce a new *aggregate relation*, notated Σ. Consider a tree $\langle \Sigma : c \rangle$, whose root is connected to a child c via Σ. If the denotation of c is a set of values s, the parent's denotation is then a singleton set containing s. Formally:

$$[\![\langle \Sigma : c \rangle]\!]_w = \{[\![c]\!]_w\}. \quad (2)$$

Figure 3(a) shows the DCS tree for our running example. The denotation of the middle node is $\{s\}$, where s is all major cities. Having instantiated s as a value, everything above this node is an ordinary CSP: s constrains the `count` node, which in turns constrains the root node to $|s|$.

A DCS tree that contains only join and aggregate relations can be viewed as a collection of tree-structured CSPs connected via aggregate relations. The tree structure still enables us to compute denotations efficiently based on (1) and (2).

2.2 Full Version

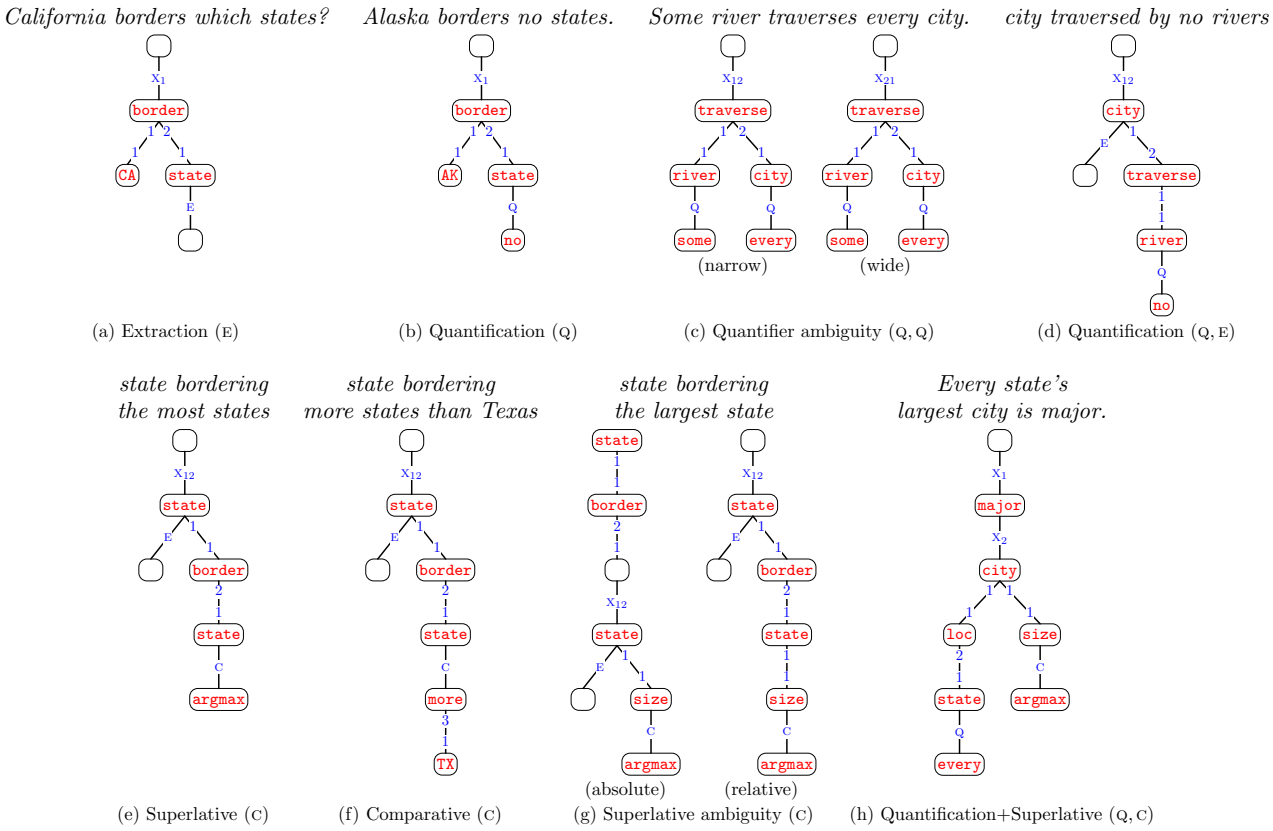
The basic version of DCS described thus far handles a core subset of language. But consider Figure 4: (a) is headed by *borders*, but *states* needs to be extracted; in (b), the quantifier *no* is syntactically dominated by the head verb *borders* but needs to take wider scope. We now present the full version of DCS which handles this type of divergence between syntactic and semantic scope.

The key idea that allows us to give semantically-scoped denotations to syntactically-scoped trees is as follows: We mark a node low in the tree with a *mark relation* (one of E, Q, or C). Then higher up in the tree, we invoke it with an *execute relation* $\text{X}_\mathbf{i}$ to create the desired semantic scope.[2]

This mark-execute construct acts non-locally, so to maintain compositionality, we must augment the

[1] Infinite denotations (such as $[\![<]\!]_w$) are represented as implicit sets on which we can perform membership queries. The intersection of two sets can be performed as long as at least one of the sets is finite.

[2] Our mark-execute construct is analogous to Montague's quantifying in, Cooper storage, and Carpenter's scoping constructor (Carpenter, 1998).



(a) Extraction (E) (b) Quantification (Q) (c) Quantifier ambiguity (Q, Q) (d) Quantification (Q, E)

(e) Superlative (C) (f) Comparative (C) (g) Superlative ambiguity (C) (h) Quantification+Superlative (Q, C)

Figure 4: Example DCS trees for utterances in which syntactic and semantic scope diverge. These trees reflect the syntactic structure, which facilitates parsing, but importantly, these trees also precisely encode the correct semantic scope. The main mechanism is using a mark relation (E, Q, or C) low in the tree paired with an execute relation ($\mathrm{X}_\mathbf{i}$) higher up at the desired semantic point.

denotation $d = [\![z]\!]_w$ to include any information about the marked nodes in z that can be accessed by an execute relation later on. In the basic version, d was simply the consistent assignments to the root. Now d contains the consistent joint assignments to the *active nodes* (which include the root and all marked nodes), as well as information stored about each marked node. Think of d as consisting of n *columns*, one for each active node according to a pre-order traversal of z. Column 1 always corresponds to the root node. Formally, a denotation is defined as follows (see Figure 5 for an example):

Definition 2 (Denotations) *Let $\mathcal{D}$ be the set of denotations, where each $d \in \mathcal{D}$ consists of*

- *a set of arrays $d.A$, where each array $\mathbf{a} = [a_1, \ldots, a_n] \in d.A$ is a sequence of n tuples ($a_i \in \mathcal{V}^*$); and*
- *a list of n* stores *$d.\boldsymbol{\alpha} = (d.\alpha_1, \ldots, d.\alpha_n)$, where each store α contains a mark relation $\alpha.r \in \{\mathrm{E}, \mathrm{Q}, \mathrm{C}, \emptyset\}$, a base denotation $\alpha.b \in \mathcal{D} \cup \{\emptyset\}$, and a child denotation $\alpha.c \in \mathcal{D} \cup \{\emptyset\}$.*

We write d as $\langle\!\langle A; (r_1, b_1, c_1); \ldots; (r_n, b_n, c_n) \rangle\!\rangle$. We use $d\{r_i = x\}$ to mean d with $d.r_i = d.\alpha_i.r = x$ (similar definitions apply for $d\{\alpha_i = x\}$, $d\{b_i = x\}$, and $d\{c_i = x\}$).

The denotation of a DCS tree can now be defined recursively:

$$[\![\langle p \rangle]\!]_w = \langle\!\langle \{[v] : v \in w(p)\}; \emptyset \rangle\!\rangle, \quad (3)$$

$$[\![\left\langle p; \mathbf{e}; \begin{smallmatrix} j \\ j' \end{smallmatrix} : c \right\rangle]\!]_w = [\![p; \mathbf{e}]\!]_w \bowtie_{j,j'} [\![c]\!]_w, \quad (4)$$

$$[\![\langle p; \mathbf{e}; \Sigma : c \rangle]\!]_w = [\![p; \mathbf{e}]\!]_w \bowtie_{*,*} \Sigma\left([\![c]\!]_w\right), \quad (5)$$

$$[\![\langle p; \mathbf{e}; \mathrm{X}_\mathbf{i} : c \rangle]\!]_w = [\![p; \mathbf{e}]\!]_w \bowtie_{*,*} \mathbf{X_i}([\![c]\!]_w), \quad (6)$$

$$[\![\langle p; \mathbf{e}; \mathrm{E} : c \rangle]\!]_w = \mathbf{M}([\![p; \mathbf{e}]\!]_w, \mathrm{E}, c), \quad (7)$$

$$[\![\langle p; \mathbf{e}; \mathrm{C} : c \rangle]\!]_w = \mathbf{M}([\![p; \mathbf{e}]\!]_w, \mathrm{C}, c), \quad (8)$$

$$[\![\langle p; \mathrm{Q} : c; \mathbf{e} \rangle]\!]_w = \mathbf{M}([\![p; \mathbf{e}]\!]_w, \mathrm{Q}, c). \quad (9)$$

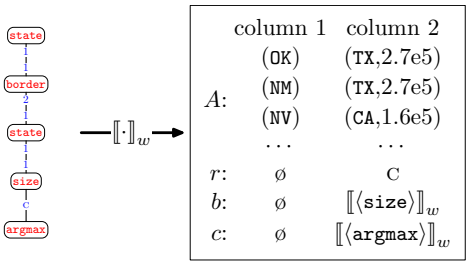


Figure 5: Example of the denotation for a DCS tree with a compare relation C. This denotation has two columns, one for each active node—the root node `state` and the marked node `size`.

The base case is defined in (3): if z is a single node with predicate p, then the denotation of z has one column with the tuples $w(p)$ and an empty store. The other six cases handle different edge relations. These definitions depend on several operations ($\bowtie_{\mathbf{j},\mathbf{j}'}, \Sigma, \mathbf{X_i}, \mathbf{M}$) which we will define shortly, but let us first get some intuition.

Let z be a DCS tree. If the last child c of z's root is a join ($\binom{j}{j'}$), aggregate (Σ), or execute ($\mathrm{X}_\mathbf{i}$) relation ((4)–(6)), then we simply recurse on z with c removed and join it with some transformation (identity, Σ, or $\mathrm{X}_\mathbf{i}$) of c's denotation. If the last (or first) child is connected via a mark relation E, C (or Q), then we strip off that child and put the appropriate information in the store by invoking $\mathbf{M}$.

We now define the operations $\bowtie_{\mathbf{j},\mathbf{j}'}, \Sigma, \mathbf{X_i}, \mathbf{M}$. Some helpful notation: For a sequence $\mathbf{v} = (v_1, \ldots, v_n)$ and indices $\mathbf{i} = (i_1, \ldots, i_k)$, let $\mathbf{v_i} = (v_{i_1}, \ldots, v_{i_k})$ be the projection of $\mathbf{v}$ onto $\mathbf{i}$; we write $\mathbf{v}_{-\mathbf{i}}$ to mean $\mathbf{v}_{[1,\ldots,n]\setminus\mathbf{i}}$. Extending this notation to denotations, let $\langle\!\langle A; \boldsymbol{\alpha} \rangle\!\rangle[\mathbf{i}] = \langle\!\langle \{\mathbf{a_i} : \mathbf{a} \in A\}; \boldsymbol{\alpha_i} \rangle\!\rangle$. Let $d[-\emptyset] = d[-\mathbf{i}]$, where $\mathbf{i}$ are the columns with empty stores. For example, for d in Figure 5, $d[1]$ keeps column 1, $d[-\emptyset]$ keeps column 2, and $d[2,-2]$ swaps the two columns.

Join The *join* of two denotations d and d' with respect to components $\mathbf{j}$ and $\mathbf{j}'$ ($*$ means all components) is formed by concatenating all arrays $\mathbf{a}$ of d with all compatible arrays $\mathbf{a}'$ of d', where compatibility means $a_{1\mathbf{j}} = a'_{1\mathbf{j}'}$. The stores are also concatenated ($\boldsymbol{\alpha} + \boldsymbol{\alpha}'$). Non-initial columns with empty stores are projected away by applying $\cdot[1,-\emptyset]$. The full definition of join is as follows:

$$\langle\!\langle A; \boldsymbol{\alpha} \rangle\!\rangle \bowtie_{\mathbf{j},\mathbf{j}'} \langle\!\langle A'; \boldsymbol{\alpha}' \rangle\!\rangle = \langle\!\langle A''; \boldsymbol{\alpha} + \boldsymbol{\alpha}' \rangle\!\rangle[1,-\emptyset],$$
$$A'' = \{\mathbf{a} + \mathbf{a}' : \mathbf{a} \in A, \mathbf{a}' \in A', a_{1\mathbf{j}} = a'_{1\mathbf{j}'}\}. \quad (10)$$

Aggregate The aggregate operation takes a denotation and forms a set out of the tuples in the first column for each setting of the rest of the columns:

$$\Sigma(\langle\!\langle A; \boldsymbol{\alpha} \rangle\!\rangle) = \langle\!\langle A' \cup A''; \boldsymbol{\alpha} \rangle\!\rangle \quad (11)$$
$$A' = \{[S(\mathbf{a}), a_2, \ldots, a_n] : \mathbf{a} \in A\}$$
$$S(\mathbf{a}) = \{a'_1 : [a'_1, a_2, \ldots, a_n] \in A\}$$
$$A'' = \{[\emptyset, a_2, \ldots, a_n] : \neg\exists a_1, \mathbf{a} \in A,$$
$$\forall 2 \le i \le n, [a_i] \in d.b_i[1].A\}.$$

2.2.1 Mark and Execute

Now we turn to the mark ($\mathbf{M}$) and execute ($\mathbf{X_i}$) operations, which handles the divergence between syntactic and semantic scope. In some sense, this is the technical core of DCS. Marking is simple: When a node (e.g., `size` in Figure 5) is marked (e.g., with relation C), we simply put the relation r, current denotation d and child c's denotation into the store of column 1:

$$\mathbf{M}(d, r, c) = d\{r_1 = r, b_1 = d, c_1 = [\![c]\!]_w\}. \quad (12)$$

The execute operation $\mathbf{X_i}(d)$ processes columns $\mathbf{i}$ in reverse order. It suffices to define $\mathbf{X}_i(d)$ for a single column i. There are three cases:

Extraction ($d.r_i = \mathrm{E}$) In the basic version, the denotation of a tree was always the set of consistent values of the root node. Extraction allows us to return the set of consistent values of a marked non-root node. Formally, extraction simply moves the i-th column to the front: $\mathbf{X}_i(d) = d[i, -(i, \emptyset)]\{\alpha_1 = \emptyset\}$. For example, in Figure 4(a), before execution, the denotation of the DCS tree is $\langle\!\langle \{[(\texttt{CA}, \texttt{OR}), (\texttt{OR})], \ldots\}; \emptyset; (\mathrm{E}, [\![\langle \texttt{state} \rangle]\!]_w, \emptyset) \rangle\!\rangle$; after applying X_1, we have $\langle\!\langle \{[(\texttt{OR})], \ldots\}; \emptyset \rangle\!\rangle$.

Generalized Quantification ($d.r_i = \mathrm{Q}$) Generalized quantifiers are predicates on two sets, a restrictor A and a nuclear scope B. For example, $w(\texttt{no}) = \{(A, B) : A \cap B = \emptyset\}$ and $w(\texttt{most}) = \{(A, B) : |A \cap B| > \frac{1}{2}|A|\}$.

In a DCS tree, the quantifier appears as the child of a Q relation, and the restrictor is the parent (see Figure 4(b) for an example). This information is retrieved from the store when the

quantifier in column i is executed. In particular, the restrictor is $A = \Sigma(d.b_i)$ and the nuclear scope is $B = \Sigma(d[i, -(i, \emptyset)])$. We then apply $d.c_i$ to these two sets (technically, denotations) and project away the first column: $\mathbf{X}_i(d) = ((d.c_i \bowtie_{1,1} A) \bowtie_{2,1} B)[-1]$.

For the example in Figure 4(b), the denotation of the DCS tree before execution is $\langle\langle \emptyset; \emptyset; (\texttt{Q}, [\![\langle \texttt{state} \rangle]\!]_w, [\![\langle \texttt{no} \rangle]\!]_w) \rangle\rangle$. The restrictor set ($A$) is the set of all states, and the nuclear scope (B) is the empty set. Since (A, B) exists in no, the final denotation, which projects away the actual pair, is $\langle\langle \{[]\} \rangle\rangle$ (our representation of true).

Figure 4(c) shows an example with two interacting quantifiers. The quantifier scope ambiguity is resolved by the choice of execute relation; $\texttt{X}_{12}$ gives the narrow reading and $\texttt{X}_{21}$ gives the wide reading. Figure 4(d) shows how extraction and quantification work together.

Comparatives and Superlatives ($d.r_i = \texttt{C}$) To compare entities, we use a set S of (x, y) pairs, where x is an entity and y is a number. For superlatives, the argmax predicate denotes pairs of sets and the set's largest element(s): $w(\texttt{argmax}) = \{(S, x^*) : x^* \in \text{argmax}_{x \in S_1} \max S(x)\}$. For comparatives, $w(\texttt{more})$ contains triples (S, x, y), where x is "more than" y as measured by S; formally: $w(\texttt{more}) = \{(S, x, y) : \max S(x) > \max S(y)\}$.

In a superlative/comparative construction, the root x of the DCS tree is the entity to be compared, the child c of a C relation is the comparative or superlative, and its parent p contains the information used for comparison (see Figure 4(e) for an example). If d is the denotation of the root, its i-th column contains this information. There are two cases: (i) if the i-th column of d contains pairs (e.g., size in Figure 5), then let $d' = [\![\langle \emptyset \rangle]\!]_w \bowtie_{1,2} d[i, -i]$, which reads out the second components of these pairs; (ii) otherwise (e.g., state in Figure 4(e)), let $d' = [\![\langle \emptyset \rangle]\!]_w \bowtie_{1,2} [\![\langle \texttt{count} \rangle]\!]_w \bowtie_{1,1} \Sigma(d[i, -i])$, which counts the number of things (e.g., states) that occur with each value of the root x. Given d', we construct a denotation S by concatenating ($+_\mathbf{i}$) the second and first columns of d' ($S = \Sigma(+_{2,1}(d'\{\alpha_2 = \emptyset\}))$) and apply the superlative/comparative: $\mathbf{X}_i(d) = ([\![\langle \emptyset \rangle]\!]_w \bowtie_{1,2} (d.c_i \bowtie_{1,1} S))\{\alpha_1 = d.\alpha_1\}$.

Figure 4(f) shows that comparatives are handled using the exact same machinery as superlatives. Figure 4(g) shows that we can naturally account for superlative ambiguity based on where the scope-determining execute relation is placed.

3 Semantic Parsing

We now turn to the task of mapping natural language utterances to DCS trees. Our first question is: given an utterance $\mathbf{x}$, what trees $z \in \mathcal{Z}$ are permissible? To define the search space, we first assume a fixed set of *lexical triggers* L. Each trigger is a pair $(\mathbf{x}, p)$, where $\mathbf{x}$ is a sequence of words (usually one) and p is a predicate (e.g., $\mathbf{x}$ = *California* and p = CA). We use $L(\mathbf{x})$ to denote the set of predicates p triggered by $\mathbf{x}$ ($(\mathbf{x}, p) \in L$). Let $L(\epsilon)$ be the set of *trace predicates*, which can be introduced without an overt lexical trigger.

Given an utterance $\mathbf{x} = (x_1, \ldots, x_n)$, we define $\mathcal{Z}_L(\mathbf{x}) \subset \mathcal{Z}$, the set of permissible DCS trees for $\mathbf{x}$. The basic approach is reminiscent of projective labeled dependency parsing: For each span $i..j$, we build a set of trees $C_{i,j}$ and set $\mathcal{Z}_L(\mathbf{x}) = C_{0,n}$. Each set $C_{i,j}$ is constructed recursively by combining the trees of its subspans $C_{i,k}$ and $C_{k',j}$ for each pair of split points k, k' (words between k and k' are ignored). These combinations are then augmented via a function A and filtered via a function F, to be specified later. Formally, $C_{i,j}$ is defined recursively as follows:

$$C_{i,j} = F\Big(A\Big(L(\mathbf{x}_{i+1..j}) \cup \bigcup_{\substack{i \le k \le k' < j \\ a \in C_{i,k} \\ b \in C_{k',j}}} T_1(a, b)\Big)\Big). \tag{13}$$

In (13), $L(\mathbf{x}_{i+1..j})$ is the set of predicates triggered by the phrase under span $i..j$ (the base case), and $T_d(a, b) = \vec{T}_d(a, b) \cup \overleftarrow{T}_d(b, a)$, which returns all ways of combining trees a and b where b is a descendant of a ($\vec{T}_d$) or vice-versa ($\overleftarrow{T}_d$). The former is defined recursively as follows: $\vec{T}_0(a, b) = \emptyset$, and

$$\vec{T}_d(a, b) = \bigcup_{\substack{r \in \mathcal{R} \\ p \in L(\epsilon)}} \{\langle a; r : b \rangle\} \cup \vec{T}_{d-1}(a, \langle p; r : b \rangle).$$

The latter ($\overleftarrow{T}_k$) is defined similarly. Essentially, $\vec{T}_d(a, b)$ allows us to insert up to d trace predicates between the roots of a and b. This is useful for modeling relations in noun compounds (e.g.,

California cities), and it also allows us to underspecify L. In particular, our L will not include verbs or prepositions; rather, we rely on the predicates corresponding to those words to be triggered by traces.

The augmentation function A takes a set of trees and optionally attaches $\texttt{E}$ and $\texttt{X}_\texttt{i}$ relations to the root (e.g., $A(\langle \texttt{city} \rangle) = \{\langle \texttt{city} \rangle, \langle \texttt{city}; \texttt{E}\!:\!\emptyset \rangle\}$). The filtering function F rules out improperly-typed trees such as $\langle \texttt{city}; {}^{0}_{0}\!:\langle \texttt{state} \rangle \rangle$. To further reduce the search space, F imposes a few additional constraints, e.g., limiting the number of marked nodes to 2 and only allowing trace predicates between arity 1 predicates.

Model We now present our discriminative semantic parsing model, which places a log-linear distribution over $z \in \mathcal{Z}_L(\mathbf{x})$ given an utterance $\mathbf{x}$. Formally, $p_\theta(z \mid \mathbf{x}) \propto e^{\phi(\mathbf{x},z)^\top \theta}$, where θ and $\phi(\mathbf{x}, z)$ are parameter and feature vectors, respectively. As a running example, consider $\mathbf{x}$ = *city that is in California* and $z = \langle \texttt{city}; {}^{1}_{1}\!:\langle \texttt{loc}; {}^{2}_{1}\!:\langle \texttt{CA} \rangle \rangle \rangle$, where *city* triggers `city` and *California* triggers `CA`.

To define the features, we technically need to augment each tree $z \in \mathcal{Z}_L(\mathbf{x})$ with alignment information—namely, for each predicate in z, the span in $\mathbf{x}$ (if any) that triggered it. This extra information is already generated from the recursive definition in (13).

The feature vector $\phi(\mathbf{x}, z)$ is defined by sums of five simple indicator feature templates: ($\mathbf{F}_1$) a word triggers a predicate (e.g., $[\textit{city}, \texttt{city}]$); ($\mathbf{F}_2$) a word is under a relation (e.g., $[\textit{that}, {}^{1}_{1}]$); ($\mathbf{F}_3$) a word is under a trace predicate (e.g., $[\textit{in}, \texttt{loc}]$); ($\mathbf{F}_4$) two predicates are linked via a relation in the left or right direction (e.g., $[\texttt{city}, {}^{1}_{1}, \texttt{loc}, \text{RIGHT}]$); and ($\mathbf{F}_5$) a predicate has a child relation (e.g., $[\texttt{city}, {}^{1}_{1}]$).

Learning Given a training dataset $\mathcal{D}$ containing $(\mathbf{x}, y)$ pairs, we define the regularized marginal log-likelihood objective $\mathcal{O}(\theta) = \sum_{(\mathbf{x},y)\in\mathcal{D}} \log p_\theta([\![z]\!]_w = y \mid \mathbf{x}, z \in \mathcal{Z}_L(\mathbf{x})) - \lambda\|\theta\|_2^2$, which sums over all DCS trees z that evaluate to the target answer y.

Our model is arc-factored, so we can sum over all DCS trees in $\mathcal{Z}_L(\mathbf{x})$ using dynamic programming. However, in order to learn, we need to sum over $\{z \in \mathcal{Z}_L(\mathbf{x}) : [\![z]\!]_w = y\}$, and unfortunately, the additional constraint $[\![z]\!]_w = y$ does not factorize. We therefore resort to beam search. Specifically, we truncate each $C_{i,j}$ to a maximum of K candidates sorted by decreasing score based on parameters θ. Let $\tilde{\mathcal{Z}}_{L,\theta}(\mathbf{x})$ be this approximation of $\mathcal{Z}_L(\mathbf{x})$.

Our learning algorithm alternates between (i) using the current parameters θ to generate the K-best set $\tilde{\mathcal{Z}}_{L,\theta}(\mathbf{x})$ for each training example $\mathbf{x}$, and (ii) optimizing the parameters to put probability mass on the correct trees in these sets; sets containing no correct answers are skipped. Formally, let $\tilde{\mathcal{O}}(\theta, \theta')$ be the objective function $\mathcal{O}(\theta)$ with $\mathcal{Z}_L(\mathbf{x})$ replaced with $\tilde{\mathcal{Z}}_{L,\theta'}(\mathbf{x})$. We optimize $\tilde{\mathcal{O}}(\theta, \theta')$ by setting $\theta^{(0)} = \vec{0}$ and iteratively solving $\theta^{(t+1)} = \text{argmax}_\theta \tilde{\mathcal{O}}(\theta, \theta^{(t)})$ using L-BFGS until $t = T$. In all experiments, we set $\lambda = 0.01$, $T = 5$, and $K = 100$. After training, given a new utterance $\mathbf{x}$, our system outputs the most likely y, summing out the latent logical form z: $\text{argmax}_y\, p_{\theta^{(T)}}(y \mid \mathbf{x}, z \in \tilde{\mathcal{Z}}_{L,\theta^{(T)}})$.

4 Experiments

We tested our system on two standard datasets, GEO and JOBS. In each dataset, each sentence $\mathbf{x}$ is annotated with a Prolog logical form, which we use only to evaluate and get an answer y. This evaluation is done with respect to a world w. Recall that a world w maps each predicate $p \in \mathcal{P}$ to a set of tuples $w(p)$. There are three types of predicates in $\mathcal{P}$: *generic* (e.g., `argmax`), *data* (e.g., `city`), and *value* (e.g., `CA`). GEO has 48 non-value predicates and JOBS has 26. For GEO, w is the standard US geography database that comes with the dataset. For JOBS, if we use the standard Jobs database, close to half the y's are empty, which makes it uninteresting. We therefore generated a random Jobs database instead as follows: we created 100 job IDs. For each data predicate p (e.g., `language`), we add each possible tuple (e.g., $(\texttt{job37}, \texttt{Java})$) to $w(p)$ independently with probability 0.8.

We used the same training-test splits as Zettlemoyer and Collins (2005) (600+280 for GEO and 500+140 for JOBS). During development, we further held out a random 30% of the training sets for validation.

Our lexical triggers L include the following: (i) predicates for a small set of $\approx$ 20 function words (e.g., $(\textit{most}, \texttt{argmax})$), (ii) (x, x) for each value

System	Accuracy
Clarke et al. (2010) w/answers	73.2
Clarke et al. (2010) w/logical forms	80.4
Our system (DCS with L)	78.9
Our system (DCS with L^+)	**87.2**

Table 2: Results on GEO with 250 training and 250 test examples. Our results are averaged over 10 random 250+250 splits taken from our 600 training examples. Of the three systems that do not use logical forms, our two systems yield significant improvements. Our better system even outperforms the system that uses logical forms.

predicate x in w (e.g., $(\textit{Boston}, \texttt{Boston})$), and (iii) predicates for each POS tag in $\{\text{JJ}, \text{NN}, \text{NNS}\}$ (e.g., $(\text{JJ}, \texttt{size})$, $(\text{JJ}, \texttt{area})$, etc.).[3] Predicates corresponding to verbs and prepositions (e.g., `traverse`) are not included as overt lexical triggers, but rather in the trace predicates $L(\epsilon)$.

We also define an augmented lexicon L^+ which includes a prototype word x for each predicate appearing in (iii) above (e.g., $(\textit{large}, \texttt{size})$), which cancels the predicates triggered by x's POS tag. For GEO, there are 22 prototype words; for JOBS, there are 5. Specifying these triggers requires minimal domain-specific supervision.

Results We first compare our system with Clarke et al. (2010) (henceforth, SEMRESP), which also learns a semantic parser from question-answer pairs. Table 2 shows that our system using lexical triggers L (henceforth, DCS) outperforms SEMRESP (78.9% over 73.2%). In fact, although neither DCS nor SEMRESP uses logical forms, DCS uses even less supervision than SEMRESP. SEMRESP requires a lexicon of 1.42 words per non-value predicate, WordNet features, and syntactic parse trees; DCS requires only words for the domain-independent predicates (overall, around 0.5 words per non-value predicate), POS tags, and very simple indicator features. In fact, DCS performs comparably to even the version of SEMRESP trained using logical forms. If we add prototype triggers (use L^+), the resulting system (DCS$^+$) outperforms both versions of SEMRESP by a significant margin (87.2% over 73.2% and 80.4%).

[3]We used the Berkeley Parser (Petrov et al., 2006) to perform POS tagging. The triggers $L(x)$ for a word x thus include $L(t)$ where t is the POS tag of x.

System	GEO	JOBS
Tang and Mooney (2001)	79.4	79.8
Wong and Mooney (2007)	86.6	–
Zettlemoyer and Collins (2005)	79.3	79.3
Zettlemoyer and Collins (2007)	81.6	–
Kwiatkowski et al. (2010)	88.2	–
Kwiatkowski et al. (2010)	88.9	–
Our system (DCS with L)	88.6	91.4
Our system (DCS with L^+)	**91.1**	**95.0**

Table 3: Accuracy (recall) of systems on the two benchmarks. The systems are divided into three groups. Group 1 uses 10-fold cross-validation; groups 2 and 3 use the independent test set. Groups 1 and 2 measure accuracy of logical form; group 3 measures accuracy of the answer; but there is very small difference between the two as seen from the Kwiatkowski et al. (2010) numbers. Our best system improves substantially over past work, despite using no logical forms as training data.

Next, we compared our systems (DCS and DCS$^+$) with the state-of-the-art semantic parsers on the full dataset for both GEO and JOBS (see Table 3). All other systems require logical forms as training data, whereas ours does not. Table 3 shows that even DCS, which does not use prototypes, is comparable to the best previous system (Kwiatkowski et al., 2010), and by adding a few prototypes, DCS$^+$ offers a decisive edge (91.1% over 88.9% on GEO). Rather than using lexical triggers, several of the other systems use IBM word alignment models to produce an initial word-predicate mapping. This option is not available to us since we do not have annotated logical forms, so we must instead rely on lexical triggers to define the search space. Note that having lexical triggers is a much weaker requirement than having a CCG lexicon, and far easier to obtain than logical forms.

Intuitions How is our system learning? Initially, the weights are zero, so the beam search is essentially unguided. We find that only for a small fraction of training examples do the K-best sets contain any trees yielding the correct answer (29% for DCS on GEO). However, training on just these examples is enough to improve the parameters, and this 29% increases to 66% and then to 95% over the next few iterations. This bootstrapping behavior occurs naturally: The "easy" examples are processed first, where easy is defined by the ability of the current

model to generate the correct answer using any tree.

Our system learns lexical associations between words and predicates. For example, *area* (by virtue of being a noun) triggers many predicates: `city`, `state`, `area`, etc. Inspecting the final parameters (DCS on GEO), we find that the feature $[\textit{area}, \texttt{area}]$ has a much higher weight than $[\textit{area}, \texttt{city}]$. Trace predicates can be inserted anywhere, but the features favor some insertions depending on the words present (for example, $[\textit{in}, \texttt{loc}]$ has high weight).

The errors that the system makes stem from multiple sources, including errors in the POS tags (e.g., *states* is sometimes tagged as a verb, which triggers no predicates), confusion of Washington state with Washington D.C., learning the wrong lexical associations due to data sparsity, and having an insufficiently large K.

5 Discussion

A major focus of this work is on our semantic representation, DCS, which offers a new perspective on compositional semantics. To contrast, consider CCG (Steedman, 2000), in which semantic parsing is driven from the lexicon. The lexicon encodes information about how each word can used in context; for example, the lexical entry for *borders* is $\text{S}\backslash\text{NP}/\text{NP} : \lambda y.\lambda x.\texttt{border}(x, y)$, which means *borders* looks right for the first argument and left for the second. These rules are often too stringent, and for complex utterances, especially in free word-order languages, either disharmonic combinators are employed (Zettlemoyer and Collins, 2007) or words are given multiple lexical entries (Kwiatkowski et al., 2010).

In DCS, we start with lexical triggers, which are more basic than CCG lexical entries. A trigger for *borders* specifies only that `border` can be used, but not how. The combination rules are encoded in the features as soft preferences. This yields a more factorized and flexible representation that is easier to search through and parametrize using features. It also allows us to easily add new lexical triggers without becoming mired in the semantic formalism.

Quantifiers and superlatives significantly complicate scoping in lambda calculus, and often type raising needs to be employed. In DCS, the mark-execute construct provides a flexible framework for dealing with scope variation. Think of DCS as a higher-level programming language tailored to natural language, which results in programs (DCS trees) which are much simpler than the logically-equivalent lambda calculus formulae.

The idea of using CSPs to represent semantics is inspired by Discourse Representation Theory (DRT) (Kamp and Reyle, 1993; Kamp et al., 2005), where variables are discourse referents. The restriction to trees is similar to economical DRT (Bos, 2009).

The other major focus of this work is program induction—inferring logical forms from their denotations. There has been a fair amount of past work on this topic: Liang et al. (2010) induces combinatory logic programs in a non-linguistic setting. Eisenstein et al. (2009) induces conjunctive formulae and uses them as features in another learning problem. Piantadosi et al. (2008) induces first-order formulae using CCG in a small domain assuming observed lexical semantics. The closest work to ours is Clarke et al. (2010), which we discussed earlier.

The integration of natural language with denotations computed against a world (grounding) is becoming increasingly popular. Feedback from the world has been used to guide both syntactic parsing (Schuler, 2003) and semantic parsing (Popescu et al., 2003; Clarke et al., 2010). Past work has also focused on aligning text to a world (Liang et al., 2009), using text in reinforcement learning (Branavan et al., 2009; Branavan et al., 2010), and many others. Our work pushes the grounded language agenda towards deeper representations of language—think grounded compositional semantics.

6 Conclusion

We built a system that interprets natural language utterances much more accurately than existing systems, despite using no annotated logical forms. Our system is based on a new semantic representation, DCS, which offers a simple and expressive alternative to lambda calculus. Free from the burden of annotating logical forms, we hope to use our techniques in developing even more accurate and broader-coverage language understanding systems.

Acknowledgments We thank Luke Zettlemoyer and Tom Kwiatkowski for providing us with data and answering questions.

References

J. Bos. 2009. A controlled fragment of DRT. In *Workshop on Controlled Natural Language*, pages 1–5.

S. Branavan, H. Chen, L. S. Zettlemoyer, and R. Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, Singapore. Association for Computational Linguistics.

S. Branavan, L. Zettlemoyer, and R. Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

B. Carpenter. 1998. *Type-Logical Semantics*. MIT Press.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Computational Natural Language Learning (CoNLL)*.

R. Dechter. 2003. *Constraint Processing*. Morgan Kaufmann.

J. Eisenstein, J. Clarke, D. Goldwasser, and D. Roth. 2009. Reading to learn: Constructing features from semantic abstracts. In *Empirical Methods in Natural Language Processing (EMNLP)*, Singapore.

R. Ge and R. J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Computational Natural Language Learning (CoNLL)*, pages 9–16, Ann Arbor, Michigan.

H. Kamp and U. Reyle. 1993. *From Discourse to Logic: An Introduction to the Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer, Dordrecht.

H. Kamp, J. v. Genabith, and U. Reyle. 2005. Discourse representation theory. In *Handbook of Philosophical Logic*.

R. J. Kate and R. J. Mooney. 2007. Learning language semantics from ambiguous supervision. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 895–900, Cambridge, MA. MIT Press.

R. J. Kate, Y. W. Wong, and R. J. Mooney. 2005. Learning to transform natural to formal languages. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1062–1068, Cambridge, MA. MIT Press.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Empirical Methods in Natural Language Processing (EMNLP)*.

P. Liang, M. I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, Singapore. Association for Computational Linguistics.

P. Liang, M. I. Jordan, and D. Klein. 2010. Learning programs: A hierarchical Bayesian approach. In *International Conference on Machine Learning (ICML)*. Omnipress.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *International Conference on Computational Linguistics and Association for Computational Linguistics (COLING/ACL)*, pages 433–440. Association for Computational Linguistics.

S. T. Piantadosi, N. D. Goodman, B. A. Ellis, and J. B. Tenenbaum. 2008. A Bayesian model of the acquisition of compositional semantics. In *Proceedings of the Thirtieth Annual Conference of the Cognitive Science Society*.

H. Poon and P. Domingos. 2009. Unsupervised semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*, Singapore.

A. Popescu, O. Etzioni, and H. Kautz. 2003. Towards a theory of natural language interfaces to databases. In *International Conference on Intelligent User Interfaces (IUI)*.

W. Schuler. 2003. Using model-theoretic semantic interpretation to guide statistical parsing and word recognition in a spoken language interface. In *Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

M. Steedman. 2000. *The Syntactic Process*. MIT Press.

L. R. Tang and R. J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *European Conference on Machine Learning*, pages 466–477.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Association for Computational Linguistics (ACL)*, pages 960–967, Prague, Czech Republic. Association for Computational Linguistics.

M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic proramming. In *Association for the Advancement of Artificial Intelligence (AAAI)*, Cambridge, MA. MIT Press.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*, pages 658–666.

L. S. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 678–687.

Unsupervised Part-of-Speech Tagging with Bilingual Graph-Based Projections

Dipanjan Das*
Carnegie Mellon University
Pittsburgh, PA 15213, USA
dipanjan@cs.cmu.edu

Slav Petrov
Google Research
New York, NY 10011, USA
slav@google.com

Abstract

We describe a novel approach for inducing unsupervised part-of-speech taggers for languages that have no labeled training data, but have translated text in a resource-rich language. Our method does not assume any knowledge about the target language (in particular no tagging dictionary is assumed), making it applicable to a wide array of resource-poor languages. We use graph-based label propagation for cross-lingual knowledge transfer and use the projected labels as features in an unsupervised model (Berg-Kirkpatrick et al., 2010). Across eight European languages, our approach results in an average absolute improvement of 10.4% over a state-of-the-art baseline, and 16.7% over vanilla hidden Markov models induced with the Expectation Maximization algorithm.

1 Introduction

Supervised learning approaches have advanced the state-of-the-art on a variety of tasks in natural language processing, resulting in highly accurate systems. Supervised part-of-speech (POS) taggers, for example, approach the level of inter-annotator agreement (Shen et al., 2007, 97.3% accuracy for English). However, supervised methods rely on labeled training data, which is time-consuming and expensive to generate. Unsupervised learning approaches appear to be a natural solution to this problem, as they require only unannotated text for training models. Unfortunately, the best completely unsupervised English POS tagger (that does not make use of a tagging dictionary) reaches only 76.1% accuracy (Christodoulopoulos et al., 2010), making its practical usability questionable at best.

To bridge this gap, we consider a practically motivated scenario, in which we want to leverage existing resources from a resource-rich language (like English) when building tools for resource-poor foreign languages.[1] We assume that absolutely no labeled training data is available for the foreign language of interest, but that we have access to parallel data with a resource-rich language. This scenario is applicable to a large set of languages and has been considered by a number of authors in the past (Alshawi et al., 2000; Xi and Hwa, 2005; Ganchev et al., 2009). Naseem et al. (2009) and Snyder et al. (2009) study related but different multilingual grammar and tagger induction tasks, where it is assumed that no labeled data at all is available.

Our work is closest to that of Yarowsky and Ngai (2001), but differs in two important ways. First, we use a novel graph-based framework for projecting syntactic information across language boundaries. To this end, we construct a bilingual graph over word types to establish a connection between the two languages (§3), and then use graph label propagation to project syntactic information from English to the foreign language (§4). Second, we treat the projected labels as features in an unsuper-

*This research was carried out during an internship at Google Research.

[1] For simplicity of exposition we refer to the resource-poor language as the "foreign language." Similarly, we use English as the resource-rich language, but any other language with labeled resources could be used instead.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 600–609,
Portland, Oregon, June 19-24, 2011.

vised model (§5), rather than using them directly for supervised training. To make the projection practical, we rely on the twelve *universal part-of-speech tags* of Petrov et al. (2011). Syntactic universals are a well studied concept in linguistics (Carnie, 2002; Newmeyer, 2005), and were recently used in similar form by Naseem et al. (2010) for multilingual grammar induction. Because there might be some controversy about the exact definitions of such universals, this set of coarse-grained POS categories is defined operationally, by collapsing language (or treebank) specific distinctions to a set of categories that exists across all languages. These universal POS categories not only facilitate the transfer of POS information from one language to another, but also relieve us from using controversial evaluation metrics,[2] by establishing a direct correspondence between the induced hidden states in the foreign language and the observed English labels.

We evaluate our approach on eight European languages (§6), and show that both our contributions provide consistent and statistically significant improvements. Our final average POS tagging accuracy of 83.4% compares very favorably to the average accuracy of Berg-Kirkpatrick et al.'s monolingual unsupervised state-of-the-art model (73.0%), and considerably bridges the gap to fully supervised POS tagging performance (96.6%).

2 Approach Overview

The focus of this work is on building POS taggers for foreign languages, assuming that we have an English POS tagger and some parallel text between the two languages. Central to our approach (see Algorithm 1) is a bilingual similarity graph built from a sentence-aligned parallel corpus. As discussed in more detail in §3, we use two types of vertices in our graph: on the foreign language side vertices correspond to trigram types, while the vertices on the English side are individual word types. Graph construction does not require any labeled data, but makes use of two similarity functions. The edge weights between the foreign language trigrams are computed using a co-occurence based similarity function, designed to indicate how syntactically similar the middle words of the connected trigrams are (§3.2). To establish a soft correspondence between the two languages, we use a second similarity function, which leverages standard unsupervised word alignment statistics (§3.3).[3]

Algorithm 1 Bilingual POS Induction

Require: Parallel English and foreign language data $\mathcal{D}^e$ and $\mathcal{D}^f$, unlabeled foreign training data Γ^f; English tagger.

Ensure: Θ^f, a set of parameters learned using a constrained unsupervised model (§5).

1: $\mathcal{D}^{e \leftrightarrow f} \leftarrow \textsf{word-align-bitext}(\mathcal{D}^e, \mathcal{D}^f)$
2: $\widehat{\mathcal{D}^e} \leftarrow \textsf{pos-tag-supervised}(\mathcal{D}^e)$
3: $\mathcal{A} \leftarrow \textsf{extract-alignments}(\mathcal{D}^{e \leftrightarrow f}, \widehat{\mathcal{D}^e})$
4: $\mathsf{G} \leftarrow \textsf{construct-graph}(\Gamma^f, \mathcal{D}^f, \mathcal{A})$
5: $\tilde{\mathsf{G}} \leftarrow \textsf{graph-propagate}(\mathsf{G})$
6: $\Delta \leftarrow \textsf{extract-word-constraints}(\tilde{\mathsf{G}})$
7: $\Theta^f \leftarrow \textsf{pos-induce-constrained}(\Gamma^f, \Delta)$
8: Return Θ^f

Since we have no labeled foreign data, our goal is to project syntactic information from the English side to the foreign side. To initialize the graph we tag the English side of the parallel text using a supervised model. By aggregating the POS labels of the English tokens to types, we can generate label distributions for the English vertices. Label propagation can then be used to transfer the labels to the *peripheral* foreign vertices (i.e. the ones adjacent to the English vertices) first, and then among all of the foreign vertices (§4). The POS distributions over the foreign trigram types are used as features to learn a better unsupervised POS tagger (§5). The following three sections elaborate these different stages is more detail.

3 Graph Construction

In graph-based learning approaches one constructs a graph whose vertices are labeled and unlabeled examples, and whose weighted edges encode the degree to which the examples they link have the same label (Zhu et al., 2003). Graph construction for structured prediction problems such as POS tagging is non-trivial: on the one hand, using individual words as the vertices throws away the context

[2] See Christodoulopoulos et al. (2010) for a discussion of metrics for evaluating unsupervised POS induction systems.

[3] The word alignment methods do not use POS information.

necessary for disambiguation; on the other hand, it is unclear how to define (sequence) similarity if the vertices correspond to entire sentences. Altun et al. (2005) proposed a technique that uses graph based similarity between labeled and unlabeled parts of structured data in a discriminative framework for semi-supervised learning. More recently, Subramanya et al. (2010) defined a graph over the cliques in an underlying structured prediction model. They considered a semi-supervised POS tagging scenario and showed that one can use a graph over trigram types, and edge weights based on distributional similarity, to improve a supervised conditional random field tagger.

3.1 Graph Vertices

We extend Subramanya et al.'s intuitions to our bilingual setup. Because the information flow in our graph is asymmetric (from English to the foreign language), we use different types of vertices for each language. The foreign language vertices (denoted by V_f) correspond to foreign trigram types, exactly as in Subramanya et al. (2010). On the English side, however, the vertices (denoted by V_e) correspond to word types. Because all English vertices are going to be labeled, we do not need to disambiguate them by embedding them in trigrams. Furthermore, we do not connect the English vertices to each other, but only to foreign language vertices.[4]

The graph vertices are extracted from the different sides of a parallel corpus ($\mathcal{D}^e$, $\mathcal{D}^f$) and an additional unlabeled monolingual foreign corpus Γ^f, which will be used later for training. We use two different similarity functions to define the edge weights among the foreign vertices and between vertices from different languages.

3.2 Monolingual Similarity Function

Our monolingual similarity function (for connecting pairs of foreign trigram types) is the same as the one used by Subramanya et al. (2010). We briefly review it here for completeness. We define a symmetric similarity function $\mathrm{K}(u_i, u_j)$ over two foreign language vertices $u_i, u_j \in V_f$ based on the co-occurrence statistics of the nine feature concepts given in Table 1. Each feature concept is akin to a random variable and its occurrence in the text corresponds to a particular instantiation of that random variable. For each trigram type $x_2\ x_3\ x_4$ in a sequence $x_1\ x_2\ x_3\ x_4\ x_5$, we count how many times that trigram type co-occurs with the different instantiations of each concept, and compute the point-wise mutual information (PMI) between the two.[5] The similarity between two trigram types is given by summing over the PMI values over feature instantiations that they have in common. This is similar to stacking the different feature instantiations into long (sparse) vectors and computing the cosine similarity between them. Finally, note that while most feature concepts are lexicalized, others, such as the suffix concept, are not.

Description	Feature
Trigram + Context	$x_1\ x_2\ x_3\ x_4\ x_5$
Trigram	$x_2\ x_3\ x_4$
Left Context	$x_1\ x_2$
Right Context	$x_4\ x_5$
Center Word	x_3
Trigram − Center Word	$x_2\ x_4$
Left Word + Right Context	$x_2\ x_4\ x_5$
Left Context + Right Word	$x_1\ x_2\ x_4$
Suffix	HasSuffix(x_3)

Table 1: Various features used for computing edge weights between foreign trigram types.

Given this similarity function, we define a nearest neighbor graph, where the edge weight for the n most similar vertices is set to the value of the similarity function and to 0 for all other vertices. We use $\mathcal{N}(u)$ to denote the neighborhood of vertex u, and fixed $n = 5$ in our experiments.

3.3 Bilingual Similarity Function

To define a similarity function between the English and the foreign vertices, we rely on high-confidence word alignments. Since our graph is built from a parallel corpus, we can use standard word alignment techniques to align the English sentences $\mathcal{D}^e$

[4] This is because we are primarily interested in learning foreign language taggers, rather than improving supervised English taggers. Note, however, that it would be possible to use our graph-based framework also for completely unsupervised POS induction in both languages, similar to Snyder et al. (2009).

[5] Note that many combinations are impossible giving a PMI value of 0; e.g., when the trigram type and the feature instantiation don't have words in common.

and their foreign language translations $\mathcal{D}^f$.[6] Label propagation in the graph will provide coverage and high recall, and we therefore extract only intersected high-confidence (> 0.9) alignments $\mathcal{D}^{e \leftrightarrow f}$.

Based on these high-confidence alignments we can extract tuples of the form $[u \leftrightarrow v]$, where u is a foreign trigram type, whose middle word aligns to an English word type v. Our bilingual similarity function then sets the edge weights in proportion to these tuple counts.

3.4 Graph Initialization

So far the graph has been completely unlabeled. To initialize the graph for label propagation we use a supervised English tagger to label the English side of the bitext.[7] We then simply count the individual labels of the English tokens and normalize the counts to produce tag distributions over English word types. These tag distributions are used to initialize the label distributions over the English vertices in the graph. Note that since all English vertices were extracted from the parallel text, we will have an initial label distribution for all vertices in V_e.

3.5 Graph Example

A very small excerpt from an Italian-English graph is shown in Figure 1. As one can see, only the trigrams [suo **incarceramento** ,], [suo **iter** ,] and [suo **carattere** ,] are connected to English words. In this particular case, all English vertices are labeled as nouns by the supervised tagger. In general, the neighborhoods can be more diverse and we allow a soft label distribution over the vertices. It is worth noting that the middle words of the Italian trigrams are nouns too, which exhibits the fact that the similarity metric connects types having the same syntactic category. In the label propagation stage, we propagate the automatic English tags to the aligned Italian trigram types, followed by further propagation solely among the Italian vertices.

[6] We ran six iterations of IBM Model 1 (Brown et al., 1993), followed by six iterations of the HMM model (Vogel et al., 1996) in both directions.

[7] We used a tagger based on a trigram Markov model (Brants, 2000) trained on the Wall Street Journal portion of the Penn Treebank (Marcus et al., 1993), for its fast speed and reasonable accuracy (96.7% on sections 22-24 of the treebank, but presumably much lower on the (out-of-domain) parallel corpus).

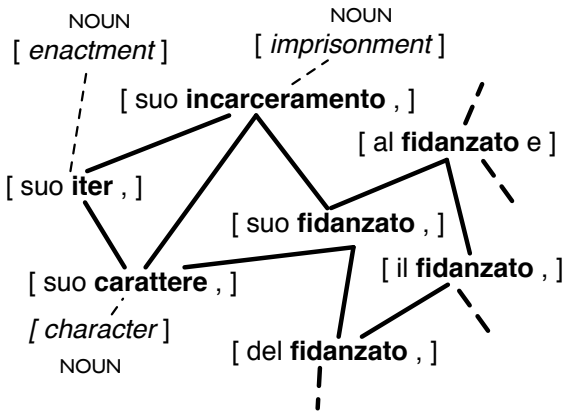


Figure 1: An excerpt from the graph for Italian. Three of the Italian vertices are connected to an automatically labeled English vertex. Label propagation is used to propagate these tags inwards and results in tag distributions for the middle word of each Italian trigram.

4 POS Projection

Given the bilingual graph described in the previous section, we can use label propagation to project the English POS labels to the foreign language. We use label propagation in two stages to generate soft labels on all the vertices in the graph. In the first stage, we run a single step of label propagation, which transfers the label distributions from the English vertices to the connected foreign language vertices (say, V_f^l) at the periphery of the graph. Note that because we extracted only high-confidence alignments, many foreign vertices will not be connected to any English vertices. This stage of label propagation results in a tag distribution r_i over labels y, which encodes the proportion of times the middle word of $u_i \in V_f$ aligns to English words v_y tagged with label y:

$$\mathrm{r}_i(y) = \frac{\sum_{v_y} \#[u_i \leftrightarrow v_y]}{\sum_{y'} \sum_{v_{y'}} \#[u_i \leftrightarrow v_{y'}]} \quad (1)$$

The second stage consists of running traditional label propagation to propagate labels from these peripheral vertices V_f^l to all foreign language vertices

in the graph, optimizing the following objective:

$$\begin{aligned}\mathcal{C}(\mathrm{q}) &= \sum_{u_i \in V_f \setminus V_f^l, u_j \in \mathcal{N}(u_i)} w_{ij} \|\mathrm{q}_i - \mathrm{q}_j\|^2 \\ &+ \nu \sum_{u_i \in V_f \setminus V_f^l} \|\mathrm{q}_i - U\|^2 \\ \text{s.t.} \quad & \sum_y \mathrm{q}_i(y) = 1 \; \forall u_i \\ & \mathrm{q}_i(y) \geq 0 \; \forall u_i, y \\ & \mathrm{q}_i = \mathrm{r}_i \; \forall u_i \in V_f^l \end{aligned} \tag{2}$$

where the q_i ($i = 1, \ldots, |V_f|$) are the label distributions over the foreign language vertices and μ and ν are hyperparameters that we discuss in §6.4. We use a squared loss to penalize neighboring vertices that have different label distributions: $\|\mathrm{q}_i - \mathrm{q}_j\|^2 = \sum_y (\mathrm{q}_i(y) - \mathrm{q}_j(y))^2$, and additionally regularize the label distributions towards the uniform distribution U over all possible labels $\mathcal{Y}$. It can be shown that this objective is convex in q.

The first term in the objective function is the graph smoothness regularizer which encourages the distributions of similar vertices (large w_{ij}) to be similar. The second term is a regularizer and encourages all type marginals to be uniform to the extent that is allowed by the first two terms (cf. maximum entropy principle). If an unlabeled vertex does not have a path to any labeled vertex, this term ensures that the converged marginal for this vertex will be uniform over all tags, allowing the middle word of such an unlabeled vertex to take on any of the possible tags.

While it is possible to derive a closed form solution for this convex objective function, it would require the inversion of a matrix of order $|V_f|$. Instead, we resort to an iterative update based method. We formulate the update as follows:

$$\mathrm{q}_i^{(m)}(y) = \begin{cases} \mathrm{r}_i(y) & \text{if } u_i \in V_f^l \\ \dfrac{\gamma_i(y)}{\kappa_i} & \text{otherwise} \end{cases} \tag{3}$$

where $\forall u_i \in V_f \setminus V_f^l$, $\gamma_i(y)$ and κ_i are defined as:

$$\gamma_i(y) = \sum_{u_j \in \mathcal{N}(u_i)} w_{ij} \mathrm{q}_j^{(m-1)}(y) + \nu \, U(y) \tag{4}$$

$$\kappa_i = \nu + \sum_{u_j \in \mathcal{N}(u_i)} w_{ij} \tag{5}$$

We ran this procedure for 10 iterations.

5 POS Induction

After running label propagation (LP), we compute tag probabilities for foreign word types x by marginalizing the POS tag distributions of foreign trigrams $u_i = x_- \; x \; x_+$ over the left and right context words:

$$p(y|x) = \frac{\sum_{x_-, x_+} \mathrm{q}_i(y)}{\sum_{x_-, x_+, y'} \mathrm{q}_i(y')} \tag{6}$$

We then extract a set of possible tags $\mathtt{t}_x(y)$ by eliminating labels whose probability is below a threshold value τ:

$$\mathtt{t}_x(y) = \begin{cases} 1 & \text{if } p(y|x) \geq \tau \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

We describe how we choose τ in §6.4. This vector $\mathtt{t}_x$ is constructed for every word in the foreign vocabulary and will be used to provide features for the unsupervised foreign language POS tagger.

We develop our POS induction model based on the feature-based HMM of Berg-Kirkpatrick et al. (2010). For a sentence $\mathbf{x}$ and a state sequence $\mathbf{z}$, a first order Markov model defines a distribution:

$$\begin{aligned} P_\Theta(\mathbf{X} = \mathbf{x}, \mathbf{Z} = \mathbf{z}) &= P_\Theta(Z_1 = z_1) \cdot \\ \prod_{i=1}^{|\mathbf{x}|} & \underbrace{P_\Theta(Z_{i+1} = z_{i+1} \mid Z_i = z_i)}_{\text{transition}} \cdot \\ & \underbrace{P_\Theta(X_i = x_i \mid Z_i = z_i)}_{\text{emission}} \end{aligned} \tag{8}$$

In a traditional Markov model, the emission distribution $P_\Theta(X_i = x_i \mid Z_i = z_i)$ is a set of multinomials. The feature-based model replaces the emission distribution with a log-linear model, such that:

$$P_\Theta(X = x \mid Z = z) = \frac{\exp \Theta^\top \mathbf{f}(x, z)}{\sum_{x' \in \text{Val}(X)} \exp \Theta^\top \mathbf{f}(x', z)} \tag{9}$$

where $\text{Val}(X)$ corresponds to the entire vocabulary. This locally normalized log-linear model can look at various aspects of the observation x, incorporating overlapping features of the observation. In our experiments, we used the same set of features as Berg-Kirkpatrick et al. (2010): an indicator feature based

on the word identity x, features checking whether x contains digits or hyphens, whether the first letter of x is upper case, and suffix features up to length 3. All features were conjoined with the state z.

We trained this model by optimizing the following objective function:

$$\mathcal{L}(\Theta) = \sum_{i=1}^{N} \log \sum_{\mathbf{z}} P_\Theta(\mathbf{X} = \mathbf{x}^{(i)}, \mathbf{Z} = \mathbf{z}^{(i)}) - C\|\Theta\|_2^2 \quad (10)$$

Note that this involves marginalizing out all possible state configurations $\mathbf{z}$ for a sentence $\mathbf{x}$, resulting in a non-convex objective. To optimize this function, we used L-BFGS, a quasi-Newton method (Liu and Nocedal, 1989). For English POS tagging, Berg-Kirkpatrick et al. (2010) found that this direct gradient method performed better (>7% absolute accuracy) than using a feature-enhanced modification of the Expectation-Maximization (EM) algorithm (Dempster et al., 1977).[8] Moreover, this route of optimization outperformed a vanilla HMM trained with EM by 12%.

We adopted this state-of-the-art model because it makes it easy to experiment with various ways of incorporating our novel *constraint* feature into the log-linear emission model. This feature $\mathsf{f_t}$ incorporates information from the smoothed graph and prunes hidden states that are inconsistent with the thresholded vector t_x. The function $\lambda : \mathsf{F} \rightarrow \mathsf{C}$ maps from the language specific fine-grained tagset F to the coarser universal tagset C and is described in detail in §6.2:

$$\mathsf{f_t}(x, z) = \log(\mathsf{t}_x(y)), \text{ if } \lambda(z) = y \quad (11)$$

Note that when $\mathsf{t}_x(y) = 1$ the feature value is 0 and has no effect on the model, while its value is $-\infty$ when $\mathsf{t}_x(y) = 0$ and constrains the HMM's state space. This formulation of the constraint feature is equivalent to the use of a tagging dictionary extracted from the graph using a threshold τ on the posterior distribution of tags for a given word type (Eq. 7). It would have therefore also been possible to use the integer programming (IP) based approach of Ravi and Knight (2009) instead of the feature-HMM for POS induction on the foreign side. However, we do not explore this possibility in the current work.

[8] See §3.1 of Berg-Kirkpatrick et al. (2010) for more details about their modification of EM, and how gradients are computed for L-BFGS.

6 Experiments and Results

Before presenting our results, we describe the datasets that we used, as well as two baselines.

6.1 Datasets

We utilized two kinds of datasets in our experiments: (i) monolingual treebanks[9] and (ii) large amounts of parallel text with English on one side. The availability of these resources guided our selection of foreign languages. For monolingual treebank data we relied on the CoNLL-X and CoNLL-2007 shared tasks on dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007). The parallel data came from the Europarl corpus (Koehn, 2005) and the ODS United Nations dataset (UN, 2006). Taking the intersection of languages in these resources, and selecting languages with large amounts of parallel data, yields the following set of eight Indo-European languages: Danish, Dutch, German, Greek, Italian, Portuguese, Spanish and Swedish.

Of course, we are primarily interested in applying our techniques to languages for which no labeled resources are available. However, we needed to restrict ourselves to these languages in order to be able to evaluate the performance of our approach. We paid particular attention to minimize the number of free parameters, and used the same hyperparameters for all language pairs, rather than attempting language-specific tuning. We hope that this will allow practitioners to apply our approach directly to languages for which no resources are available.

6.2 Part-of-Speech Tagset and HMM States

We use the universal POS tagset of Petrov et al. (2011) in our experiments.[10] This set C consists of the following 12 coarse-grained tags: NOUN (nouns), VERB (verbs), ADJ (adjectives), ADV (adverbs), PRON (pronouns), DET (determiners), ADP (prepositions or postpositions), NUM (numerals), CONJ (conjunctions), PRT (particles), PUNC

[9] We extracted only the words and their POS tags from the treebanks.

[10] Available at http://code.google.com/p/universal-pos-tags/.

(punctuation marks) and X (a catch-all for other categories such as abbreviations or foreign words). While there might be some controversy about the exact definition of such a tagset, these 12 categories cover the most frequent part-of-speech and exist in one form or another in all of the languages that we studied.

For each language under consideration, Petrov et al. (2011) provide a mapping λ from the fine-grained language specific POS tags in the foreign treebank to the universal POS tags. The supervised POS tagging accuracies (on this tagset) are shown in the last row of Table 2. The taggers were trained on datasets labeled with the universal tags.

The number of latent HMM states for each language in our experiments was set to the number of fine tags in the language's treebank. In other words, the set of hidden states F was chosen to be the fine set of treebank tags. Therefore, the number of fine tags varied across languages for our experiments; however, one could as well have fixed the set of HMM states to be a constant across languages, and created one mapping to the universal POS tagset.

6.3 Various Models

To provide a thorough analysis, we evaluated three baselines and two oracles in addition to two variants of our graph-based approach. We were intentionally lenient with our baselines:

- **EM-HMM:** A traditional HMM baseline, with multinomial emission and transition distributions estimated by the Expectation Maximization algorithm. We evaluated POS tagging accuracy using the lenient many-to-1 evaluation approach (Johnson, 2007).
- **Feature-HMM:** The vanilla feature-HMM of Berg-Kirkpatrick et al. (2010) (i.e. no additional constraint feature) served as a second baseline. Model parameters were estimated with L-BFGS and evaluation again used a greedy many-to-1 mapping.
- **Projection:** Our third baseline incorporates bilingual information by projecting POS tags directly across alignments in the parallel data. For unaligned words, we set the tag to the most frequent tag in the corresponding treebank. For each language, we took the same number of sentences from the bitext as there are in its treebank, and trained a *supervised* feature-HMM. This can be seen as a rough approximation of Yarowsky and Ngai (2001).

We tried two versions of our graph-based approach:

- **No LP:** Our first version takes advantage of our bilingual graph, but extracts the constraint feature after the first stage of label propagation (Eq. 1). Because many foreign word types are not aligned to an English word (see Table 3), and we do not run label propagation on the foreign side, we expect the projected information to have less coverage. Furthermore we expect the label distributions on the foreign to be fairly noisy, because the graph constraints have not been taken into account yet.
- **With LP:** Our full model uses both stages of label propagation (Eq. 2) before extracting the constraint features. As a result, we are able to extract the constraint feature for all foreign word types and furthermore expect the projected tag distributions to be smoother and more stable.

Our oracles took advantage of the labeled treebanks:

- **TB Dictionary:** We extracted tagging dictionaries from the treebanks and and used them as constraint features in the feature-based HMM. Evaluation was done using the prespecified mappings.
- **Supervised:** We trained the supervised model of Brants (2000) on the original treebanks and mapped the language-specific tags to the universal tags for evaluation.

6.4 Experimental Setup

While we tried to minimize the number of free parameters in our model, there are a few hyperparameters that need to be set. Fortunately, performance was stable across various values, and we were able to use the same hyperparameters for all languages.

We used $C = 1.0$ as the L_2 regularization constant in (Eq. 10) and trained both EM and L-BFGS for 1000 iterations. When extracting the vector

	Model	Danish	Dutch	German	Greek	Italian	Portuguese	Spanish	Swedish	**Avg**
baselines	EM-HMM	68.7	57.0	75.9	65.8	63.7	62.9	71.5	68.4	66.7
	Feature-HMM	69.1	65.1	81.3	71.8	68.1	78.4	80.2	70.1	73.0
	Projection	73.6	77.0	83.2	79.3	79.7	82.6	80.1	74.7	78.8
our approach	No LP	79.0	78.8	82.4	76.3	84.8	87.0	82.8	79.4	81.3
	With LP	**83.2**	**79.5**	82.8	**82.5**	**86.8**	**87.9**	**84.2**	**80.5**	83.4
oracles	TB Dictionary	*93.1*	*94.7*	*93.5*	*96.6*	*96.4*	*94.0*	*95.8*	*85.5*	*93.7*
	Supervised	*96.9*	*94.9*	*98.2*	*97.8*	*95.8*	*97.2*	*96.8*	*94.8*	*96.6*

Table 2: Part-of-speech tagging accuracies for various baselines and oracles, as well as our approach. "**Avg**" denotes macro-average across the eight languages.

t_x used to compute the constraint feature from the graph, we tried three threshold values for τ (see Eq. 7). Because we don't have a separate development set, we used the training set to select among them and found 0.2 to work slightly better than 0.1 and 0.3. For seven out of eight languages a threshold of 0.2 gave the best results for our final model, which indicates that for languages without any validation set, $\tau = 0.2$ can be used. For graph propagation, the hyperparameter ν was set to 2×10^{-6} and was not tuned. The graph was constructed using 2 million trigrams; we chose these by truncating the parallel datasets up to the number of sentence pairs that contained 2 million trigrams.

6.5 Results

Table 2 shows our complete set of results. As expected, the vanilla HMM trained with EM performs the worst. The feature-HMM model works better for all languages, generalizing the results achieved for English by Berg-Kirkpatrick et al. (2010). Our "Projection" baseline is able to benefit from the bilingual information and greatly improves upon the monolingual baselines, but falls short of the "No LP" model by 2.5% on an average. The "No LP" model does not outperform direct projection for German and Greek, but performs better for six out of eight languages. Overall, it gives improvements ranging from 1.1% for German to 14.7% for Italian, for an average improvement of 8.3% over the unsupervised feature-HMM model. For comparison, the completely unsupervised feature-HMM baseline accuracy on the universal POS tags for English is 79.4%, and goes up to 88.7% with a treebank dictionary.

Our full model ("With LP") outperforms the unsupervised baselines and the "No LP" setting for all languages. It falls short of the "Projection" baseline for German, but is statistically indistinguishable in terms of accuracy. As indicated by bolding, for seven out of eight languages the improvements of the "With LP" setting are statistically significant with respect to the other models, including the "No LP" setting.[11] Overall, it performs 10.4% better than the hitherto state-of-the-art feature-HMM baseline, and 4.6% better than direct projection, when we macro-average the accuracy over all languages.

6.6 Discussion

Our full model outperforms the "No LP" setting because it has better vocabulary coverage and allows the extraction of a larger set of constraint features. We tabulate this increase in Table 3. For all languages, the vocabulary sizes increase by several thousand words. Although the tag distributions of the foreign words (Eq. 6) are noisy, the results confirm that label propagation within the foreign language part of the graph adds significant quality for every language.

Figure 2 shows an excerpt of a sentence from the Italian test set and the tags assigned by four different models, as well as the gold tags. While the first three models get three to four tags wrong, our best model gets only one word wrong and is the most accurate among the four models for this example. Examining the word *fidanzato* for the "No LP" and "With LP" models is particularly instructive. As Figure 1 shows, this word has no high-confidence alignment in the Italian-English bitext. As a result, its POS tag needs to be induced in the "No LP" case, while the

[11] A word level paired-t-test is significant at $p < 0.01$ for Danish, Greek, Italian, Portuguese, Spanish and Swedish, and $p < 0.05$ for Dutch.

	si	trovava	in	un	parco	con	il	fidanzato	Paolo	F.	,	27	anni	,	rappresentante
EM-HMM:	*CONJ*	*NOUN*	*DET*	DET	NOUN	ADP	DET	NOUN	.	NOUN	.	NUM	NOUN	.	NOUN
Feature-HMM:	PRON	VERB	ADP	DET	NOUN	*CONJ*	DET	NOUN	NOUN	NOUN	.	*ADP*	NOUN	.	*VERB*
No LP:	*VERB*	VERB	ADP	DET	NOUN	ADP	DET	*ADJ*	NOUN	*ADJ*	.	NUM	NOUN	.	NOUN
With LP:	*VERB*	VERB	ADP	DET	NOUN	ADP	DET	NOUN	NOUN	NOUN	.	NUM	NOUN	.	NOUN
Gold:	PRON	VERB	ADP	DET	NOUN	ADP	DET	NOUN	NOUN	NOUN	.	NUM	NOUN	.	NOUN

Figure 2: Tags produced by the different models along with the reference set of tags for a part of a sentence from the Italian test set. Italicized tags denote incorrect labels.

Language	**# words with constraints**	
	"No LP"	"With LP"
Danish	88,240	128, 391
Dutch	51,169	74,892
German	59,534	107,249
Greek	90,231	114,002
Italian	48,904	62,461
Portuguese	46,787	65,737
Spanish	72,215	82,459
Swedish	70,181	88,454

Table 3: Size of the vocabularies for the "No LP" and "With LP" models for which we can impose constraints.

correct tag is available as a constraint feature in the "With LP" case.

7 Conclusion

We have shown the efficacy of graph-based label propagation for projecting part-of-speech information across languages. Because we are interested in applying our techniques to languages for which no labeled resources are available, we paid particular attention to minimize the number of free parameters and used the same hyperparameters for all language pairs. Our results suggest that it is possible to learn accurate POS taggers for languages which do not have any annotated data, but have translations into a resource-rich language. Our results outperform strong unsupervised baselines as well as approaches that rely on direct projections, and bridge the gap between purely supervised and unsupervised POS tagging models.

Acknowledgements

We would like to thank Ryan McDonald for numerous discussions on this topic. We would also like to thank Amarnag Subramanya for helping us with the implementation of label propagation and Shankar Kumar for access to the parallel data. Finally, we thank Kuzman Ganchev and the three anonymous reviewers for helpful suggestions and comments on earlier drafts of this paper.

References

Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. 2000. Head-transducer models for speech translation and their automatic acquisition from bilingual data. *Machine Translation*, 15.

Yasemin Altun, David McAllester, and Mikhail Belkin. 2005. Maximum margin semi-supervised learning for structured variables. In *Proc. of NIPS*.

Taylor Berg-Kirkpatrick, Alexandre B. Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proc. of NAACL-HLT*.

Thorsten Brants. 2000. TnT - a statistical part-of-speech tagger. In *Proc. of ANLP*.

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

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

Andrew Carnie. 2002. *Syntax: A Generative Introduction (Introducing Linguistics)*. Blackwell Publishing.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two decades of unsupervised POS induction: How far have we come? In *Proc. of EMNLP*.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proc. of ACL-IJCNLP*.

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proc. of EMNLP-CoNLL*.

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

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45.

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

Tahira Naseem, Benjamin Snyder, Jacob Eisenstein, and Regina Barzilay. 2009. Multilingual part-of-speech tagging: Two unsupervised approaches. *JAIR*, 36.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proc. of EMNLP*.

Frederick J. Newmeyer. 2005. *Possible and Probable Languages: A Generative Perspective on Linguistic Typology.* Oxford University Press.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of CoNLL.*

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. *ArXiv:1104.2086*.

Sujith Ravi and Kevin Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proc. of ACL-IJCNLP*.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proc. of ACL.*

Benjamin Snyder, Tahira Naseem, and Regina Barzilay. 2009. Unsupervised multilingual grammar induction. In *Proc. of ACL-IJCNLP*.

Amar Subramanya, Slav Petrov, and Fernando Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *Proc. of EMNLP*.

UN. 2006. ODS UN parallel corpus.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proc. of COLING*.

Chenhai Xi and Rebecca Hwa. 2005. A backoff model for bootstrapping resources for non-English languages. In *Proc. of HLT-EMNLP*.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proc. of NAACL.*

Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. of ICML.*

Global Learning of Typed Entailment Rules

Jonathan Berant
Tel Aviv University
Tel Aviv, Israel
jonatha6@post.tau.ac.il

Ido Dagan
Bar-Ilan University
Ramat-Gan, Israel
dagan@cs.biu.ac.il

Jacob Goldberger
Bar-Ilan University
Ramat-Gan, Israel
goldbej@eng.biu.ac.il

Abstract

Extensive knowledge bases of entailment rules between predicates are crucial for applied semantic inference. In this paper we propose an algorithm that utilizes transitivity constraints to learn a globally-optimal set of entailment rules for typed predicates. We model the task as a graph learning problem and suggest methods that scale the algorithm to larger graphs. We apply the algorithm over a large data set of extracted predicate instances, from which a resource of typed entailment rules has been recently released (Schoenmackers et al., 2010). Our results show that using global transitivity information substantially improves performance over this resource and several baselines, and that our scaling methods allow us to increase the scope of global learning of entailment-rule graphs.

1 Introduction

Generic approaches for applied semantic inference from text gained growing attention in recent years, particularly under the Textual Entailment (TE) framework (Dagan et al., 2009). TE is a generic paradigm for semantic inference, where the objective is to recognize whether a target meaning can be inferred from a given text. A crucial component of inference systems is extensive resources of *entailment rules*, also known as *inference rules*, i.e., rules that specify a directional inference relation between fragments of text. One important type of rule is rules that specify entailment relations between predicates and their arguments. For example, the rule *'X annex Y → X control Y'* helps recognize that the text *'Japan annexed Okinawa'* answers the question *'Which country controls Okinawa?'*. Thus, acquisition of such knowledge received considerable attention in the last decade (Lin and Pantel, 2001; Sekine, 2005; Szpektor and Dagan, 2009; Schoenmackers et al., 2010).

Most past work took a "local learning" approach, learning each entailment rule independently of others. It is clear though, that there are global interactions between predicates. Notably, entailment is a *transitive* relation and so the rules $A \rightarrow B$ and $B \rightarrow C$ imply $A \rightarrow C$.

Recently, Berant et al. (2010) proposed a *global* graph optimization procedure that uses *Integer Linear Programming (ILP)* to find the best set of entailment rules under a transitivity constraint. Imposing this constraint raised two challenges. The first of *ambiguity:* transitivity does not always hold when predicates are ambiguous, e.g., *X buy Y → X acquire Y* and *X acquire Y → X learn Y*, but *X buy Y ↛ X learn Y* since these two rules correspond to two different senses of *acquire*. The second challenge is *scalability:* ILP solvers do not scale well since ILP is an NP-complete problem. Berant et al. circumvented these issues by learning rules where one of the predicate's arguments is instantiated (e.g., *'X reduce nausea → X affect nausea'*), which is useful for learning small graphs on-the-fly, given a target concept such as *nausea*. While rules may be effectively learned when needed, their scope is narrow and they are not useful as a generic knowledge resource.

This paper aims to take global rule learning one step further. To this end, we adopt the representation suggested by Schoenmackers et al. (2010), who learned inference rules between *typed predicates*, i.e., predicates where the argument types (e.g., *city* or *drug*) are specified. Schoenmackers et al. uti-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 610–619,
Portland, Oregon, June 19-24, 2011.

lized typed predicates since they were dealing with noisy and ambiguous web text. Typing predicates helps disambiguation and filtering of noise, while still maintaining rules of wide-applicability. Their method employs a local learning approach, while the number of predicates in their data is too large to be handled directly by an ILP solver.

In this paper we suggest applying global optimization learning to open domain typed entailment rules. To that end, we show how to construct a structure termed *typed entailment graph*, where the nodes are typed predicates and the edges represent entailment rules. We suggest scaling techniques that allow to optimally learn such graphs over a large set of typed predicates by first *decomposing* nodes into components and then applying *incremental ILP* (Riedel and Clarke, 2006). Using these techniques, the obtained algorithm is guaranteed to return an optimal solution. We ran our algorithm over the data set of Schoenmackers et al. and release a resource of 30,000 rules[1] that achieves substantially higher recall without harming precision. To the best of our knowledge, this is the first resource of that scale to use global optimization for learning predicative entailment rules. Our evaluation shows that global transitivity improves the F_1 score of rule learning by 27% over several baselines and that our scaling techniques allow dealing with larger graphs, resulting in improved coverage.

2 Background

Most work on learning entailment rules between predicates considered each rule independently of others, using two sources of information: *lexicographic resources* and *distributional similarity*.

Lexicographic resources are manually-prepared knowledge bases containing semantic information on predicates. A widely-used resource is WordNet (Fellbaum, 1998), where relations such as *synonymy* and *hyponymy* can be used to generate rules. Other resources include NomLex (Macleod et al., 1998; Szpektor and Dagan, 2009) and FrameNet (Baker and Lowe, 1998; Ben Aharon et al., 2010).

Lexicographic resources are accurate but have low coverage. Distributional similarity algorithms use large corpora to learn broader resources by assuming that semantically similar predicates appear with similar arguments. These algorithms usually represent a predicate with one or more vectors and use some function to compute argument similarity. Distributional similarity algorithms differ in their *feature representation*: Some use a *binary* representation: each predicate is represented by one feature vector where each feature is a pair of arguments (Szpektor et al., 2004; Yates and Etzioni, 2009). This representation performs well, but suffers when data is sparse. The *binary-DIRT* representation deals with sparsity by representing a predicate with a pair of vectors, one for each argument (Lin and Pantel, 2001). Last, a richer form of representation, termed *unary*, has been suggested where a different predicate is defined for each argument (Szpektor and Dagan, 2008). Different algorithms also differ in their similarity function. Some employ symmetric functions, geared towards *paraphrasing* (bi-directional entailment), while others choose directional measures more suited for entailment (Bhagat et al., 2007). In this paper, We employ several such functions, such as *Lin* (Lin and Pantel, 2001), and *BInc* (Szpektor and Dagan, 2008).

Schoenmackers et al. (2010) recently used distributional similarity to learn rules between *typed predicates*, where the left-hand-side of the rule may contain more than a single predicate (horn clauses). In their work, they used Hearst-patterns (Hearst, 1992) to extract a set of 29 million *(argument, type)* pairs from a large web crawl. Then, they employed several filtering methods to clean this set and automatically produced a mapping of 1.1 million arguments into 156 types. Examples for *(argument, type)* pairs are *(EXODUS, book)*, *(CHINA, country)* and *(ASTHMA, disease)*. Schoenmackers et al. then utilized the types, the mapped arguments and tuples from TextRunner (Banko et al., 2007) to generate 10,672 typed predicates (such as *conquer(country,city)* and *common in(disease,place)*), and learn 30,000 rules between these predicates[2]. In this paper we will learn entailment rules over the same data set, which was generously provided by

[1] The resource can be downloaded from http://www.cs.tau.ac.il/~jonatha6/homepage_files/resources/ACL2011Resource.zip

[2] The rules and the mapping of arguments into types can be downloaded from http://www.cs.washington.edu/research/sherlock-hornclauses/

Schoenmackers et al.

As mentioned above, Berant et al. (2010) used global transitivity information to learn small *entailment graphs*. Transitivity was also used as an information source in other fields of NLP: Taxonomy Induction (Snow et al., 2006), Co-reference Resolution (Finkel and Manning, 2008), Temporal Information Extraction (Ling and Weld, 2010), and Unsupervised Ontology Induction (Poon and Domingos, 2010). Our proposed algorithm applies to any sparse transitive relation, and so might be applicable in these fields as well.

Last, we formulate our optimization problem as an Integer Linear Program (ILP). ILP is an optimization problem where a linear objective function over a set of integer variables is maximized under a set of linear constraints. Scaling ILP is challenging since it is an NP-complete problem. ILP has been extensively used in NLP lately (Clarke and Lapata, 2008; Martins et al., 2009; Do and Roth, 2010).

3 Typed Entailment Graphs

Given a set of typed predicates, entailment rules can only exist between predicates that share the same (unordered) pair of types (such as *place* and *country*)[3]. Hence, every pair of types defines a graph that describes the entailment relations between predicates sharing those types (Figure 1). Next, we show how to represent entailment rules between typed predicates in a structure termed *typed entailment graph*, which will be the learning goal of our algorithm.

A typed entailment graph is a directed graph where the nodes are *typed predicates*. A typed predicate is a triple $p(t_1, t_2)$ representing a predicate in natural language. p is the lexical realization of the predicate and the types t_1, t_2 are variables representing argument types. These are taken from a set of types T, where each type $t \in T$ is a bag of natural language words or phrases. Examples for typed predicates are: *conquer(country,city)* and *contain(product,material)*. An *instance* of a typed predicate is a triple $p(a_1, a_2)$, where $a_1 \in t_1$ and $a_2 \in t_2$ are termed *arguments*. For example, *be common in(ASTHMA,AUSTRALIA)* is an instance of *be common in(disease,place)*. For brevity, we refer to typed entailment graphs and typed predicates as *entailment graphs* and *predicates* respectively.

Edges in typed entailment graphs represent entailment rules: an edge (u, v) means that predicate u entails predicate v. If the type t_1 is different from the type t_2, mapping of arguments is straightforward, as in the rule '*be find in(material,product)* $\rightarrow$ *contain(product,material)*'. We term this *a two-types entailment graph*. When t_1 and t_2 are equal, mapping of arguments is ambiguous: we distinguish *direct-mapping edges* where the first argument on the left-hand-side (LHS) is mapped to the first argument on the right-hand-side (RHS), as in '*beat(team,team)* $\xrightarrow{d}$ *defeat(team,team)*', and *reversed-mapping edges* where the LHS first argument is mapped to the RHS second argument, as in '*beat(team,team)* $\xrightarrow{r}$ *lose to(team,team)*'. We term this *a single-type entailment graph*. Note that in single-type entailment graphs reversed-mapping loops are possible as in '*play(team,team)* $\xrightarrow{r}$ *play(team,team)*': if team A plays team B, then team B plays team A.

Since entailment is a transitive relation, typed-entailment graphs are transitive: if the edges (u, v) and (v, w) are in the graph so is the edge (u, w). Note that in single-type entailment graphs one needs to consider whether mapping of edges is direct or reversed: if mapping of both (u, v) and (v, w) is either direct or reversed, mapping of (u, w) is direct, otherwise it is reversed.

Typing plays an important role in rule transitivity: if predicates are ambiguous, transitivity does not necessarily hold. However, typing predicates helps disambiguate them and so the problem of ambiguity is greatly reduced.

4 Learning Typed Entailment Graphs

Our learning algorithm is composed of two steps: (1) Given a set of typed predicates and their instances extracted from a corpus, we train a (local) *entailment classifier* that estimates for every pair of predicates whether one entails the other. (2) Using the classifier scores we perform global optimization, i.e., learn the set of edges over the nodes that maximizes the global score of the graph under transitivity and background-knowledge constraints.

Section 4.1 describes the local classifier training

[3] Otherwise, the rule would contain unbound variables.

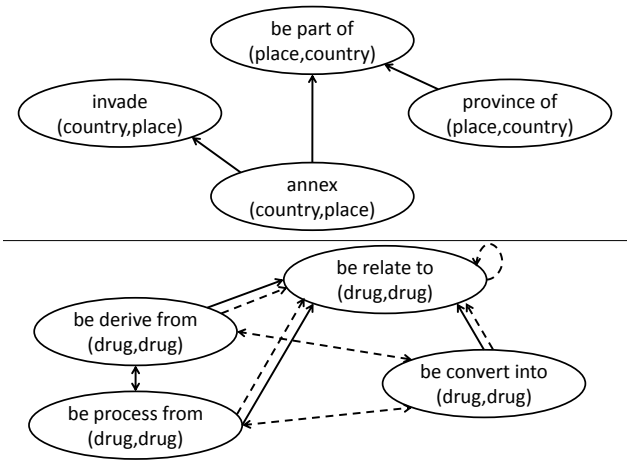


Figure 1: *Top:* A fragment of a two-types entailment graph. *bottom:* A fragment of a single-type entailment graph. Mapping of solid edges is direct and of dashed edges is reversed.

procedure. Section 4.2 gives an ILP formulation for the optimization problem. Sections 4.3 and 4.4 propose scaling techniques that exploit graph sparsity to optimally solve larger graphs.

4.1 Training an entailment classifier

Similar to the work of Berant et al. (2010), we use "distant supervision". Given a lexicographic resource (WordNet) and a set of predicates with their instances, we perform the following three steps (see Table 1):

1) Training set generation We use WordNet to generate positive and negative examples, where each example is a pair of predicates. Let P be the set of input typed predicates. For every predicate $p(t_1, t_2) \in P$ such that p is a single word, we extract from WordNet the set S of synonyms and direct hypernyms of p. For every $p' \in S$, if $p'(t_1, t_2) \in P$ then $p(t_1, t_2) \rightarrow p'(t_1, t_2)$ is taken as a positive example.

Negative examples are generated in a similar manner, with direct co-hyponyms of p (sister nodes in WordNet) and hyponyms at distance 2 instead of synonyms and direct hypernyms. We also generate negative examples by randomly sampling pairs of typed predicates that share the same types.

2) Feature representation Each example pair of predicates (p_1, p_2) is represented by a feature vector, where each feature is a specific distributional similarity score estimating whether p_1 entails p_2. We compute 11 distributional similarity scores for each pair of predicates based on the arguments appearing in the extracted arguments. The first 6 scores are computed by trying all combinations of the similarity functions *Lin* and *BInc* with the feature representations *unary*, *binary-DIRT* and *binary* (see Section 2). The other 5 scores were provided by Schoenmackers et al. (2010) and include *SR* (Schoenmackers et al., 2010), *LIME* (McCreath and Sharma, 1997), *M-estimate* (Dzeroski and Brakto, 1992), the standard *G-test* and a simple implementation of *Cover* (Weeds and Weir, 2003). Overall, the rationale behind this representation is that combining various scores will yield a better classifier than each single measure.

Type	example
hyper.	*beat(team,team)* $\rightarrow$ *play(team,team)*
syno.	*reach(team,game)* $\rightarrow$ *arrive at(team,game)*
cohypo.	*invade(country,city)* $\nrightarrow$ *bomb(country,city)*
hypo.	*defeat(city,city)* $\nrightarrow$ *eliminate(city,city)*
random	*hold(place,event)* $\nrightarrow$ *win(place,event)*

Table 1: Automatically generated training set examples.

3) Training We train over an equal number of positive and negative examples, as classifiers tend to perform poorly on the minority class when trained on imbalanced data (Van Hulse et al., 2007; Nikulin, 2008).

4.2 ILP formulation

Once the classifier is trained, we would like to learn all edges (entailment rules) of each typed entailment graph. Given a set of predicates V and an entailment score function $f : V \times V \rightarrow \mathbb{R}$ derived from the classifier, we want to find a graph $G = (V, E)$ that respects transitivity and maximizes the sum of edge weights $\sum_{(u,v)\in E} f(u, v)$. This problem is NP-hard by a reduction from the NP-hard *Transitive Subgraph* problem (Yannakakis, 1978). Thus, employing ILP is an appealing approach for obtaining an optimal solution.

For two-types entailment graphs the formulation is simple: The ILP variables are indicators X_{uv} denoting whether an edge (u, v) is in the graph, with the following ILP:

$$\hat{G} = arg\,max \sum_{u \neq v} f(u,v) \cdot X_{uv} \quad (1)$$

$$\text{s.t. } \forall_{u,v,w \in V}\ X_{uv} + X_{vw} - X_{uw} \leq 1 \quad (2)$$

$$\forall_{u,v \in A_{yes}}\ X_{uv} = 1 \quad (3)$$

$$\forall_{u,v \in A_{no}}\ X_{uv} = 0 \quad (4)$$

$$\forall_{u \neq v}\ X_{uv} \in \{0,1\} \quad (5)$$

The objective in Eq. 1 is a sum over the weights of the eventual edges. The constraint in Eq. 2 states that edges must respect transitivity. The constraints in Eq. 3 and 4 state that for known node pairs, defined by A_{yes} and A_{no}, we have background knowledge indicating whether entailment holds or not. We elaborate on how A_{yes} and A_{no} were constructed in Section 5. For a graph with n nodes we get $n(n-1)$ variables and $n(n-1)(n-2)$ transitivity constraints.

The simplest way to expand this formulation for single-type graphs is to duplicate each predicate node, with one node for each order of the types, and then the ILP is unchanged. However, this is inefficient as it results in an ILP with $2n(2n-1)$ variables and $2n(2n-1)(2n-2)$ transitivity constraints. Since our main goal is to scale the use of ILP, we modify it a little. We denote a direct-mapping edge (u,v) by the indicator X_{uv} and a reversed-mapping edge (u,v) by Y_{uv}. The functions f_d and f_r provide scores for direct and reversed mappings respectively. The objective in Eq. 1 and the constraint in Eq. 2 are replaced by (Eq. 3, 4 and 5 still exist and are carried over in a trivial manner):

$$arg\,max \sum_{u \neq v} f_d(u,v) X_{uv} + \sum_{u,v} f_r(u,v) Y_{uv} \quad (6)$$

$$\text{s.t. } \forall_{u,v,w \in V}\ X_{uv} + X_{vw} - X_{uw} \leq 1$$

$$\forall_{u,v,w \in V}\ X_{uv} + Y_{vw} - Y_{uw} \leq 1$$

$$\forall_{u,v,w \in V}\ Y_{uv} + X_{vw} - Y_{uw} \leq 1$$

$$\forall_{u,v,w \in V}\ Y_{uv} + Y_{vw} - X_{uw} \leq 1$$

The modified constraints capture the transitivity behavior of direct-mapping and reversed-mapping edges, as described in Section 3. This results in $2n^2 - n$ variables and about $4n^3$ transitivity constraints, cutting the ILP size in half.

Next, we specify how to derive the function f from the trained classifier using a probabilistic formulation[4]. Following Snow et al. (2006) and Berant et al. (2010), we utilize a probabilistic entailment classifier that computes the posterior $P_{uv} = P(X_{uv} = 1|F_{uv})$. We want to use P_{uv} to derive the posterior $P(G|F)$, where $F = \cup_{u \neq v} F_{uv}$ and F_{uv} is the feature vector for a node pair (u,v).

Since the classifier was trained on a balanced training set, the prior over the two entailment classes is uniform and so by Bayes rule $P_{uv} \propto P(F_{uv}|X_{uv} = 1)$. Using that and the exact same three independence assumptions described by Snow et al. (2006) and Berant et al. (2010) we can show that (for brevity, we omit the full derivation):

$$\hat{G} = arg\,max_G \log P(G|F) = \quad (7)$$

$$arg\,max \sum_{u \neq v} (\log \frac{P_{uv} \cdot P(X_{uv}=1)}{(1-P_{uv})P(X_{uv}=0)}) X_{uv}$$

$$= arg\,max \sum_{u \neq v} (\log \frac{P_{uv}}{1-P_{uv}}) X_{uv} + \log \eta \cdot |E|$$

where $\eta = \frac{P(X_{uv}=1)}{P(X_{uv}=0)}$ is the prior odds ratio for an edge in the graph. Comparing Eq. 1 and 7 we see that $f(u,v) = \log \frac{P_{uv} \cdot P(X_{uv}=1)}{(1-P_{uv})P(X_{uv}=0)}$. Note that f is composed of a likelihood component and an *edge prior* expressed by $P(X_{uv} = 1)$, which we assume to be some constant. This constant is a parameter that affects graph sparsity and controls the trade-off between recall and precision.

Next, we show how sparsity is exploited to scale the use of ILP solvers. We discuss two-types entailment graphs, but generalization is simple.

4.3 Graph decomposition

Though ILP solvers provide an optimal solution, they substantially restrict the size of graphs we can work with. The number of constraints is $O(n^3)$, and solving graphs of size > 50 is often not feasible. To overcome this, we take advantage of graph sparsity: most predicates in language do not entail one another. Thus, it might be possible to decompose graphs into small components and solve each

[4]We describe two-types graphs but extending to single-type graphs is straightforward.

Algorithm 1 Decomposed-ILP

Input: A set V and a function $f : V \times V \rightarrow \mathbf{R}$
Output: An optimal set of directed edges E^*

1: $E' = \{(u, v) : f(u, v) > 0 \vee f(v, u) > 0\}$
2: $V_1, V_2, ..., V_k \leftarrow$ connected components of $G' = (V, E')$
3: **for** $i = 1$ **to** k **do**
4: $\quad E_i \leftarrow$ ApplyILPSolve(V_i,f)
5: **end for**
6: $E^* \leftarrow \bigcup_{i=1}^{k} E_i$

component separately. This is formalized in the next proposition.

Proposition 1. *If we can partition a set of nodes V into disjoint sets U, W such that for any crossing edge (u, w) between them (in either direction), $f(u, w) < 0$, then the optimal set of edges E_{opt} does not contain any crossing edge.*

Proof Assume by contradiction that E_{opt} contains a set of crossing edges E_{cross}. We can construct $E_{new} = E_{opt} \setminus E_{cross}$. Clearly $\sum_{(u,v)\in E_{new}} f(u, v) > \sum_{(u,v)\in E_{opt}} f(u, v)$, as $f(u, v) < 0$ for any crossing edge.

Next, we show that E_{new} does not violate transitivity constraints. Assume it does, then the violation is caused by omitting the edges in E_{cross}. Thus, there must be a node $u \in U$ and $w \in W$ (w.l.o.g) such that for some node v, (u, v) and (v, w) are in E_{new}, but (u, w) is not. However, this means either (u, v) or (v, w) is a crossing edge, which is impossible since we omitted all crossing edges. Thus, E_{new} is a better solution than E_{opt}, contradiction. □

This proposition suggests a simple algorithm (see Algorithm 1): Add to the graph an undirected edge for any node pair with a positive score, then find the connected components, and apply an ILP solver over the nodes in each component. The edges returned by the solver provide an optimal (not approximate) solution to the optimization problem.

The algorithm's complexity is dominated by the ILP solver, as finding connected components takes $O(V^2)$ time. Thus, efficiency depends on whether the graph is sparse enough to be decomposed into small components. Note that the edge prior plays an important role: low values make the graph sparser and easier to solve. In Section 5 we empirically test how typed entailment graphs benefit from decomposition given different prior values.

From a more general perspective, this algorithm can be applied to any problem of learning a sparse transitive binary relation. Such problems include Co-reference Resolution (Finkel and Manning, 2008) and Temporal Information Extraction (Ling and Weld, 2010). Last, the algorithm can be easily parallelized by solving each component on a different core.

Algorithm 2 Incremental-ILP

Input: A set V and a function $f : V \times V \rightarrow \mathbf{R}$
Output: An optimal set of directed edges E^*

1: ACT,VIO $\leftarrow \phi$
2: **repeat**
3: $\quad E^* \leftarrow$ ApplyILPSolve(V,f,ACT)
4: $\quad$ VIO $\leftarrow$ violated(V, E^*)
5: $\quad$ ACT $\leftarrow$ ACT $\cup$ VIO
6: **until** |VIO| $= 0$

4.4 Incremental ILP

Another solution for scaling ILP is to employ incremental ILP, which has been used in dependency parsing (Riedel and Clarke, 2006). The idea is that even if we omit the transitivity constraints, we still expect most transitivity constraints to be satisfied, given a good local entailment classifier. Thus, it makes sense to avoid specifying the constraints ahead of time, but rather add them when they are violated. This is formalized in Algorithm 2.

Line 1 initializes an *active* set of constraints and a *violated* set of constraints (*ACT;VIO*). Line 3 applies the ILP solver with the active constraints. Lines 4 and 5 find the violated constraints and add them to the active constraints. The algorithm halts when no constraints are violated. The solution is clearly optimal since we obtain a maximal solution for a less-constrained problem.

A pre-condition for using incremental ILP is that computing the violated constraints (Line 4) is efficient, as it occurs in every iteration. We do that in a straightforward manner: For every node v, and edges (u, v) and (v, w), if $(u, w) \notin E^*$ we add (u, v, w) to the violated constraints. This is cubic in worst-case but assuming the degree of nodes is bounded by a constant it is linear, and performs very

fast in practice.

Combining *Incremental-ILP* and *Decomposed-ILP* is easy: We decompose any large graph into its components and apply Incremental ILP on each component. We applied this algorithm on our evaluation data set (Section 5) and found that it converges in at most 6 iterations and that the maximal number of active constraints in large graphs drops from $\sim 10^6$ to $\sim 10^3 - 10^4$.

5 Experimental Evaluation

In this section we empirically answer the following questions: (1) Does transitivity improve rule learning over typed predicates? (Section 5.1) (2) Do *Decomposed-ILP* and *Incremental-ILP* improve scalability? (Section 5.2)

5.1 Experiment 1

A data set of 1 million TextRunner tuples (Banko et al., 2007), mapped to 10,672 distinct typed predicates over 156 types was provided by Schoenmackers et al. (2010). Readers are referred to their paper for details on mapping of tuples to typed predicates. Since entailment only occurs between predicates that share the same types, we decomposed predicates by their types (e.g., all predicates with the types *place* and *disease*) into 2,303 *typed entailment graphs*. The largest graph contains 118 nodes and the total number of potential rules is 263,756.

We generated a training set by applying the procedure described in Section 4.1, yielding 2,644 examples. We used SVMperf (Joachims, 2005) to train a Gaussian kernel classifier and computed P_{uv} by projecting the classifier output score, S_{uv}, with the sigmoid function: $P_{uv} = \frac{1}{1+exp(-S_{uv})}$. We tuned two SVM parameters using 5-fold cross validation and a development set of two typed entailment graphs.

Next, we used our algorithm to learn rules. As mentioned in Section 4.2, we integrate background knowledge using the sets A_{yes} and A_{no} that contain predicate pairs for which we know whether entailment holds. A_{yes} was constructed with syntactic rules: We normalized each predicate by omitting the first word if it is a modal and turning passives to actives. If two normalized predicates are equal they are synonymous and inserted into A_{yes}. A_{no} was constructed from 3 sources (1) Predicates differing by a single pair of words that are WordNet antonyms (2) Predicates differing by a single word of negation (3) Predicates $p(t_1, t_2)$ and $p(t_2, t_1)$ where p is a transitive verb (e.g., *beat)* in VerbNet (Kipper-Schuler et al., 2000).

We compared our algorithm (termed *ILP*$_{scale}$) to the following baselines. First, to 10,000 rules released by Schoenmackers et al. (2010) (*Sherlock*), where the LHS contains a single predicate (Schoenmackers et al. released 30,000 rules but 20,000 of those have more than one predicate on the LHS, see Section 2), as we learn rules over the same data set. Second, to distributional similarity algorithms: (a) *SR:* the score used by Schoenmackers et al. as part of the *Sherlock* system. (b) *DIRT:* (Lin and Pantel, 2001) a widely-used rule learning algorithm. (c) *BInc:* (Szpektor and Dagan, 2008) a directional rule learning algorithm. Third, we compared to the entailment classifier with no transitivity constraints (*clsf*) to see if combining distributional similarity scores improves performance over single measures. Last, we added to all baselines background knowledge with A_{yes} and A_{no} (adding the subscript X_k to their name).

To evaluate performance we manually annotated all edges in 10 typed entailment graphs - 7 two-types entailment graphs containing 14, 22, 30, 53, 62, 86 and 118 nodes, and 3 single-type entailment graphs containing 7, 38 and 59 nodes. This annotation yielded 3,427 edges and 35,585 non-edges, resulting in an empirical edge density of 9%. We evaluate the algorithms by comparing the set of edges learned by the algorithms to the gold standard edges.

Figure 2 presents the precision-recall curve of the algorithms. The curve is formed by varying a score threshold in the baselines and varying the edge prior in ILP$_{scale}$[5]. For figure clarity, we omit *DIRT* and *SR*, since *BInc* outperforms them.

Table 2 shows micro-recall, precision and F_1 at the point of maximal F_1, and the Area Under the Curve (AUC) for recall in the range of 0-0.45 for all algorithms, given background knowledge (knowledge consistently improves performance by a few points for all algorithms). The table also shows results for the rules from *Sherlock*$_k$.

[5] we stop raising the prior when run time over the graphs exceeds 2 hours. Often when the solver does not terminate in 2 hours, it also does not terminate after 24 hours or more.

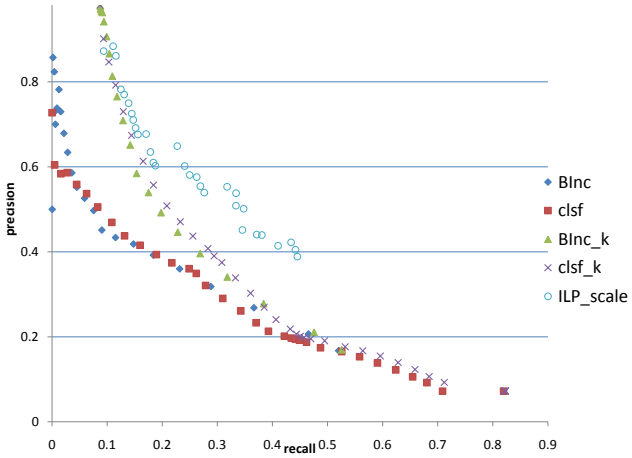


Figure 2: Precision-recall curve for the algorithms.

	micro-average			
	R (%)	P (%)	F_1 (%)	AUC
ILP_{scale}	**43.4**	42.2	**42.8**	**0.22**
$clsf_k$	30.8	37.5	33.8	0.17
$Sherlock_k$	20.6	**43.3**	27.9	N/A
$BInc_k$	31.8	34.1	32.9	0.17
SR_k	38.4	23.2	28.9	0.14
$DIRT_k$	25.7	31.0	28.1	0.13

Table 2: micro-average F_1 and AUC for the algorithms.

Results show that using global transitivity information substantially improves performance. ILP_{scale} is better than all other algorithms by a large margin starting from recall .2, and improves AUC by 29% and the maximal F_1 by 27%. Moreover, ILP_{scale} doubles recall comparing to the rules from the *Sherlock* resource, while maintaining comparable precision.

5.2 Experiment 2

We want to test whether using our scaling techniques, *Decomposed-ILP* and *Incremental-ILP*, allows us to reach the optimal solution in graphs that otherwise we could not solve, and consequently increase the number of learned rules and the overall recall. To check that, we run ILP_{scale}, with and without these scaling techniques (termed ILP^{-}).

We used the same data set as in Experiment 1 and learned edges for all 2,303 entailment graphs in the data set. If the ILP solver was unable to hold the ILP in memory or took more than 2 hours for some graph, we did not attempt to learn its edges. We ran ILP_{scale} and ILP^{-} in three density modes to examine the behavior of the algorithms for different graph densities: (a) $\log \eta = -0.6$: the configuration that achieved the best recall/precision/F_1 of 43.4/42.2/42.8. (b) $\log \eta = -1$ with recall/precision/F_1 of 31.8/55.3/40.4. (c) $\log \eta = -1.75$: A high precision configuration with recall/precision/F_1 of 0.15/0.75/0.23 [6].

$\log \eta$	# unlearned	# rules	$\triangle$	Red.
-1.75	9/0	6,242 / 7,466	20%	75%
-1	9/1	16,790 / 19,396	16%	29%
-0.6	9/3	26,330 / 29,732	13%	14%

Table 3: Impact of scaling techinques (ILP^{-}/ILP_{scale}).

In each run we counted the number of graphs that could not be learned and the number of rules learned by each algorithm. In addition, we looked at the 20 largest graphs in our data (49-118 nodes) and measured the ratio r between the size of the largest component after applying *Decomposed-ILP* and the original size of the graph. We then computed the average $1-r$ over the 20 graphs to examine how graph size drops due to decomposition.

Table 3 shows the results. Column *# unlearned* and *# rules* describe the number of unlearned graphs and the number of learned rules. Column $\triangle$ shows relative increase in the number of rules learned and column *Red.* shows the average $1-r$.

ILP_{scale} increases the number of graphs that we are able to learn: in our best configuration ($\log \eta = -0.6$) only 3 graphs could not be handled comparing to 9 graphs when omitting our scaling techniques. Since the unlearned graphs are among the largest in the data set, this adds 3,500 additional rules. We compared the precision of rules learned only by ILP_{scale} with that of the rules learned by both, by randomly sampling 100 rules from each and found precision to be comparable. Thus, the additional rules learned translate into a 13% increase in relative recall without harming precision.

Also note that as density increases, the number of rules learned grows and the effectiveness of decomposition decreases. This shows how *Decomposed-ILP* is especially useful for sparse graphs. We re-

[6] Experiment was run on an Intel i5 CPU with 4GB RAM.

lease the 29,732 rules learned by the configuration $\log \eta = -0.6$ as a resource.

To sum up, our scaling techniques allow us to learn rules from graphs that standard ILP can not handle and thus considerably increase recall without harming precision.

6 Conclusions and Future Work

This paper proposes two contributions over two recent works: In the first, Berant et al. (2010) presented a *global* optimization procedure to learn entailment rules between predicates using transitivity, and applied this algorithm over small graphs where all predicates have one argument instantiated by a target concept. Consequently, the rules they learn are of limited applicability. In the second, Schoenmackers et al. learned rules of wider applicability by using *typed predicates*, but utilized a *local* approach.

In this paper we developed an algorithm that uses *global* optimization to learn widely-applicable entailment rules between typed predicates (where *both* arguments are variables). This was achieved by appropriately defining entailment graphs for typed predicates, formulating an ILP representation for them, and introducing scaling techniques that include graph decomposition and incremental ILP. Our algorithm is guaranteed to provide an optimal solution and we have shown empirically that it substantially improves performance over Schoenmackers et al.'s recent resource and over several baselines.

In future work, we aim to scale the algorithm further and learn entailment rules between untyped predicates. This would require explicit modeling of predicate ambiguity and using approximation techniques when an optimal solution cannot be attained.

Acknowledgments

This work was performed with financial support from the Turing Center at The University of Washington during a visit of the first author (NSF grant IIS-0803481). We deeply thank Oren Etzioni and Stefan Schoenmackers for providing us with the data sets for this paper and for numerous helpful discussions. We would also like to thank the anonymous reviewers for their useful comments. This work was developed under the collaboration of FBK-irst/University of Haifa and was partially supported by the Israel Science Foundation grant 1112/08. The first author is grateful to IBM for the award of an IBM Fellowship, and has carried out this research in partial fulllment of the requirements for the Ph.D. degree.

References

J. Fillmore Baker, C. F. and J. B. Lowe. 1998. The Berkeley framenet project. In *Proc. of COLING-ACL*.

Michele Banko, Michael Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of IJCAI*.

Roni Ben Aharon, Idan Szpektor, and Ido Dagan. 2010. Generating entailment rules from framenet. In *Proceedings of ACL*.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2010. Global learning of focused entailment graphs. In *Proceedings of ACL*.

Rahul Bhagat, Patrick Pantel, and Eduard Hovy. 2007. LEDIR: An unsupervised algorithm for learning directionality of inference rules. In *Proceedings of EMNLP-CoNLL*.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:273–381.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15(4):1–17.

Quang Do and Dan Roth. 2010. Constraints based taxonomic relation classification. In *Proceedings of EMNLP*.

Saso Dzeroski and Ivan Brakto. 1992. Handling noise in inductive logic programming. In *Proceedings of the International Workshop on Inductive Logic Programming*.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press.

Jenny Rose Finkel and Christopher D. Manning. 2008. Enforcing transitivity in coreference resolution. In *Proceedings of ACL-08: HLT, Short Papers*.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING*.

Thorsten Joachims. 2005. A support vector method for multivariate performance measures. In *Proceedings of ICML*.

Karin Kipper-Schuler, Hoa Trand Dang, and Martha Palmer. 2000. Class-based construction of verb lexicon. In *Proceedings of AAAI/IAAI*.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.

Xiao Ling and Daniel S. Weld. 2010. Temporal information extraction. In *Proceedings of AAAI*.

Catherine Macleod, Ralph Grishman, Adam Meyers, Leslie Barrett, and Ruth Reeves. 1998. NOMLEX: A lexicon of nominalizations. In *Proceedings of COLING*.

Andre Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of ACL*.

Eric McCreath and Arun Sharma. 1997. ILP with noise and fixed example size: a bayesian approach. In *Proceedings of the Fifteenth international joint conference on artificial intelligence - Volume 2*.

Vladimir Nikulin. 2008. Classification of imbalanced data with random sets and mean-variance filtering. *IJDWM*, 4(2):63–78.

Hoifung Poon and Pedro Domingos. 2010. Unsupervised ontology induction from text. In *Proceedings of ACL*.

Sebastian Riedel and James Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proceedings of EMNLP*.

Stefan Schoenmackers, Oren Etzioni Jesse Davis, and Daniel S. Weld. 2010. Learning first-order horn clauses from web text. In *Proceedings of EMNLP*.

Satoshi Sekine. 2005. Automatic paraphrase discovery based on context and keywords between ne pairs. In *Proceedings of IWP*.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of ACL*.

Idan Szpektor and Ido Dagan. 2008. Learning entailment rules for unary templates. In *Proceedings of COLING*.

Idan Szpektor and Ido Dagan. 2009. Augmenting wordnet-based inference with argument mapping. In *Proceedings of TextInfer*.

Idan Szpektor, Hristo Tanev, Ido Dagan, and Bonaventura Coppola. 2004. Scaling web-based acquisition of entailment relations. In *Proceedings of EMNLP*.

Jason Van Hulse, Taghi Khoshgoftaar, and Amri Napolitano. 2007. Experimental perspectives on learning from imbalanced data. In *Proceedings of ICML*.

Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *Proceedings of EMNLP*.

Mihalis Yannakakis. 1978. Node-and edge-deletion NP-complete problems. In *STOC '78: Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 253–264, New York, NY, USA. ACM.

Alexander Yates and Oren Etzioni. 2009. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34:255–296.

Incremental Syntactic Language Models for Phrase-based Translation

Lane Schwartz
Air Force Research Laboratory
Wright-Patterson AFB, OH USA
lane.schwartz@wpafb.af.mil

Chris Callison-Burch
Johns Hopkins University
Baltimore, MD USA
ccb@cs.jhu.edu

William Schuler
Ohio State University
Columbus, OH USA
schuler@ling.ohio-state.edu

Stephen Wu
Mayo Clinic
Rochester, MN USA
wu.stephen@mayo.edu

Abstract

This paper describes a novel technique for incorporating syntactic knowledge into phrase-based machine translation through incremental syntactic parsing. Bottom-up and top-down parsers typically require a completed string as input. This requirement makes it difficult to incorporate them into phrase-based translation, which generates partial hypothesized translations from left-to-right. Incremental syntactic language models score sentences in a similar left-to-right fashion, and are therefore a good mechanism for incorporating syntax into phrase-based translation. We give a formal definition of one such linear-time syntactic language model, detail its relation to phrase-based decoding, and integrate the model with the Moses phrase-based translation system. We present empirical results on a constrained Urdu-English translation task that demonstrate a significant BLEU score improvement and a large decrease in perplexity.

1 Introduction

Early work in statistical machine translation viewed translation as a noisy channel process comprised of a translation model, which functioned to posit adequate translations of source language words, and a target language model, which guided the fluency of generated target language strings (Brown et al., 1990). Drawing on earlier successes in speech recognition, research in statistical machine translation has effectively used n-gram word sequence models as language models.

Modern phrase-based translation using large scale n-gram language models generally performs well in terms of lexical choice, but still often produces ungrammatical output. Syntactic parsing may help produce more grammatical output by better modeling structural relationships and long-distance dependencies. Bottom-up and top-down parsers typically require a completed string as input; this requirement makes it difficult to incorporate these parsers into phrase-based translation, which generates hypothesized translations incrementally, from left-to-right.[1] As a workaround, parsers can rerank the translated output of translation systems (Och et al., 2004).

On the other hand, incremental parsers (Roark, 2001; Henderson, 2004; Schuler et al., 2010; Huang and Sagae, 2010) process input in a straightforward left-to-right manner. We observe that incremental parsers, used as structured language models, provide an appropriate algorithmic match to incremental phrase-based decoding. We directly integrate incremental syntactic parsing into phrase-based translation. This approach re-exerts the role of the language model as a mechanism for encouraging syntactically fluent translations.

The contributions of this work are as follows:

- A novel method for integrating syntactic LMs into phrase-based translation (§3)
- A formal definition of an incremental parser for

This research was supported by NSF CAREER/PECASE award 0447685, NSF grant IIS-0713448, and the European Commission through the EuroMatrixPlus project. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the sponsors or the United States Air Force. Cleared for public release (Case Number 88ABW-2010-6489) on 10 Dec 2010.

[1] While not all languages are written left-to-right, we will refer to incremental processing which proceeds from the beginning of a sentence as left-to-right.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 620–631,
Portland, Oregon, June 19-24, 2011.

statistical MT that can run in linear-time (§4)

- Integration with Moses (§5) along with empirical results for perplexity and significant translation score improvement on a constrained Urdu-English task (§6)

2 Related Work

Neither phrase-based (Koehn et al., 2003) nor hierarchical phrase-based translation (Chiang, 2005) take explicit advantage of the syntactic structure of either source or target language. The translation models in these techniques define *phrases* as contiguous word sequences (with gaps allowed in the case of hierarchical phrases) which may or may not correspond to any linguistic constituent. Early work in statistical phrase-based translation considered whether restricting translation models to use only syntactically well-formed constituents might improve translation quality (Koehn et al., 2003) but found such restrictions failed to improve translation quality.

Significant research has examined the extent to which syntax can be usefully incorporated into statistical tree-based translation models: string-to-tree (Yamada and Knight, 2001; Gildea, 2003; Imamura et al., 2004; Galley et al., 2004; Graehl and Knight, 2004; Melamed, 2004; Galley et al., 2006; Huang et al., 2006; Shen et al., 2008), tree-to-string (Liu et al., 2006; Liu et al., 2007; Mi et al., 2008; Mi and Huang, 2008; Huang and Mi, 2010), tree-to-tree (Abeillé et al., 1990; Shieber and Schabes, 1990; Poutsma, 1998; Eisner, 2003; Shieber, 2004; Cowan et al., 2006; Nesson et al., 2006; Zhang et al., 2007; DeNeefe et al., 2007; DeNeefe and Knight, 2009; Liu et al., 2009; Chiang, 2010), and treelet (Ding and Palmer, 2005; Quirk et al., 2005) techniques use syntactic information to inform the translation model. Recent work has shown that parsing-based machine translation using syntax-augmented (Zollmann and Venugopal, 2006) hierarchical translation grammars with rich nonterminal sets can demonstrate substantial gains over hierarchical grammars for certain language pairs (Baker et al., 2009). In contrast to the above tree-based translation models, our approach maintains a standard (non-syntactic) phrase-based translation model. Instead, we incorporate syntax into the language model.

Traditional approaches to language models in speech recognition and statistical machine translation focus on the use of n-grams, which provide a simple finite-state model approximation of the target language. Chelba and Jelinek (1998) proposed that syntactic structure could be used as an alternative technique in language modeling. This insight has been explored in the context of speech recognition (Chelba and Jelinek, 2000; Collins et al., 2005). Hassan et al. (2007) and Birch et al. (2007) use supertag n-gram LMs. Syntactic language models have also been explored with tree-based translation models. Charniak et al. (2003) use syntactic language models to rescore the output of a tree-based translation system. Post and Gildea (2008) investigate the integration of parsers as syntactic language models during binary bracketing transduction translation (Wu, 1997); under these conditions, both syntactic phrase-structure and dependency parsing language models were found to improve oracle-best translations, but did not improve actual translation results. Post and Gildea (2009) use tree substitution grammar parsing for language modeling, but do not use this language model in a translation system. Our work, in contrast to the above approaches, explores the use of incremental syntactic language models in conjunction with phrase-based translation models.

Our syntactic language model fits into the family of linear-time dynamic programming parsers described in (Huang and Sagae, 2010). Like (Galley and Manning, 2009) our work implements an incremental syntactic language model; our approach differs by calculating syntactic LM scores over all available phrase-structure parses at each hypothesis instead of the 1-best dependency parse.

The syntax-driven reordering model of Ge (2010) uses syntax-driven features to influence word order within standard phrase-based translation. The syntactic cohesion features of Cherry (2008) encourages the use of syntactically well-formed translation phrases. These approaches are fully orthogonal to our proposed incremental syntactic language model, and could be applied in concert with our work.

3 Parser as Syntactic Language Model in Phrase-Based Translation

Parsing is the task of selecting the representation $\hat{\tau}$ (typically a tree) that best models the structure of

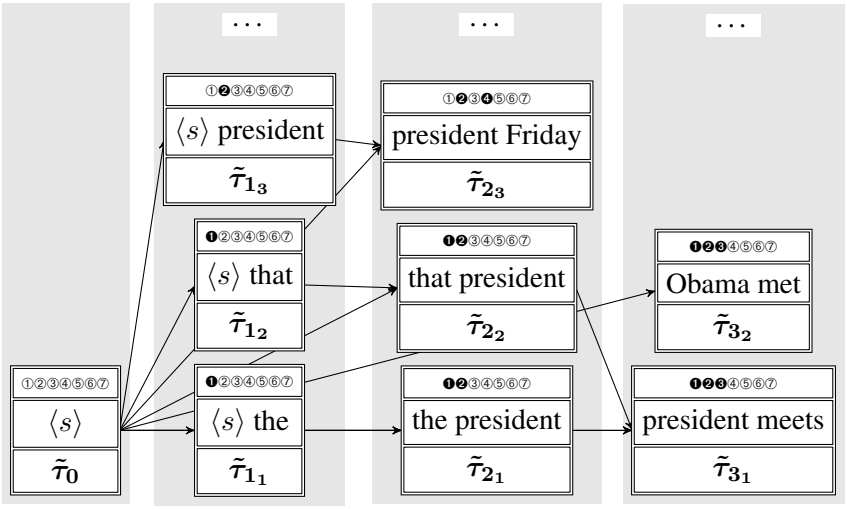


Figure 1: Partial decoding lattice for standard phrase-based decoding stack algorithm translating the German sentence *Der Präsident trifft am Freitag den Vorstand*. Each node h in decoding stack t represents the application of a translation option, and includes the source sentence coverage vector, target language n-gram state, and syntactic language model state $\tilde{\boldsymbol{\tau}}_{\boldsymbol{t_h}}$. Hypothesis combination is also shown, indicating where lattice paths with identical n-gram histories converge. We use the English translation *The president meets the board on Friday* as a running example throughout all Figures.

sentence $\mathbf{e}$, out of all such possible representations $\boldsymbol{\tau}$. This set of representations may be all phrase structure trees or all dependency trees allowed by the parsing model. Typically, tree $\hat{\tau}$ is taken to be:

$$\hat{\tau} = \underset{\tau}{\operatorname{argmax}}\, \mathsf{P}(\tau \,|\, \mathbf{e}) \tag{1}$$

We define a syntactic language model $\mathsf{P}(\mathbf{e})$ based on the total probability mass over all possible trees for string $\mathbf{e}$. This is shown in Equation 2 and decomposed in Equation 3.

$$\mathsf{P}(\mathbf{e}) = \sum_{\tau \in \boldsymbol{\tau}} \mathsf{P}(\tau, \mathbf{e}) \tag{2}$$

$$\mathsf{P}(\mathbf{e}) = \sum_{\tau \in \boldsymbol{\tau}} \mathsf{P}(\mathbf{e} \,|\, \tau)\mathsf{P}(\tau) \tag{3}$$

3.1 Incremental syntactic language model

An incremental parser processes each token of input sequentially from the beginning of a sentence to the end, rather than processing input in a top-down (Earley, 1968) or bottom-up (Cocke and Schwartz, 1970; Kasami, 1965; Younger, 1967) fashion. After processing the tth token in string $\mathbf{e}$, an incremental parser has some internal representation of possible hypothesized (incomplete) trees, $\boldsymbol{\tau_t}$. The syntactic language model probability of a partial sentence $e_1 ... e_t$ is defined:

$$\mathsf{P}(e_1 ... e_t) = \sum_{\tau \in \boldsymbol{\tau_t}} \mathsf{P}(e_1 ... e_t \,|\, \tau)\mathsf{P}(\tau) \tag{4}$$

In practice, a parser may constrain the set of trees under consideration to $\tilde{\boldsymbol{\tau}}_{\boldsymbol{t}}$, that subset of analyses or partial analyses that remains after any pruning is performed. An incremental syntactic language model can then be defined by a probability mass function (Equation 5) and a transition function δ (Equation 6). The role of δ is explained in §3.3 below. Any parser which implements these two functions can serve as a syntactic language model.

$$\mathsf{P}(e_1 ... e_t) \approx \mathsf{P}(\tilde{\boldsymbol{\tau}}_t) = \sum_{\tau \in \tilde{\boldsymbol{\tau}}_t} \mathsf{P}(e_1 ... e_t \,|\, \tau)\mathsf{P}(\tau) \tag{5}$$

$$\delta(e_t, \tilde{\boldsymbol{\tau}}_{t-1}) \rightarrow \tilde{\boldsymbol{\tau}}_t \tag{6}$$

3.2 Decoding in phrase-based translation

Given a source language input sentence $\mathbf{f}$, a trained source-to-target translation model, and a target language model, the task of translation is to find the maximally probable translation $\hat{\boldsymbol{e}}$ using a linear combination of j feature functions $\boldsymbol{h}$ weighted according to tuned parameters $\boldsymbol{\lambda}$ (Och and Ney, 2002).

$$\hat{\boldsymbol{e}} = \underset{\mathbf{e}}{\operatorname{argmax}} \exp(\sum_{j} \lambda_j h_j(\boldsymbol{e}, \boldsymbol{f})) \quad (7)$$

Phrase-based translation constructs a set of translation options — hypothesized translations for contiguous portions of the source sentence — from a trained phrase table, then incrementally constructs a lattice of partial target translations (Koehn, 2010). To prune the search space, lattice nodes are organized into beam stacks (Jelinek, 1969) according to the number of source words translated. An n-gram language model history is also maintained at each node in the translation lattice. The search space is further trimmed with hypothesis recombination, which collapses lattice nodes that share a common coverage vector and n-gram state.

3.3 Incorporating a Syntactic Language Model

Phrase-based translation produces target language words in an incremental left-to-right fashion, generating words at the beginning of a translation first and words at the end of a translation last. Similarly, incremental parsers process sentences in an incremental fashion, analyzing words at the beginning of a sentence first and words at the end of a sentence last. As such, an incremental parser with transition function δ can be incorporated into the phrase-based decoding process in a straightforward manner. Each node in the translation lattice is augmented with a syntactic language model state $\tilde{\boldsymbol{\tau}}_t$.

The hypothesis at the root of the translation lattice is initialized with $\tilde{\boldsymbol{\tau}}_0$, representing the internal state of the incremental parser before any input words are processed. The phrase-based translation decoding process adds nodes to the lattice; each new node contains one or more target language words. Each node contains a backpointer to its parent node, in which $\tilde{\boldsymbol{\tau}}_{t-1}$ is stored. Given a new target language word e_t and $\tilde{\boldsymbol{\tau}}_{t-1}$, the incremental parser's transition function δ calculates $\tilde{\boldsymbol{\tau}}_t$. Figure 1 illustrates

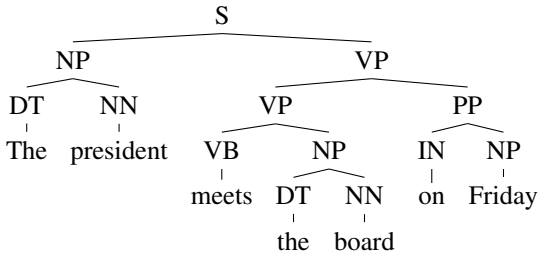


Figure 2: Sample binarized phrase structure tree.

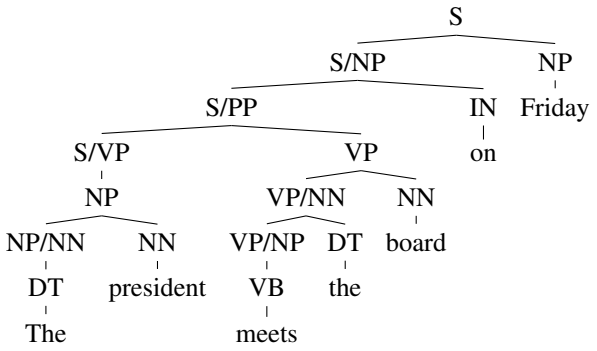


Figure 3: Sample binarized phrase structure tree after application of right-corner transform.

a sample phrase-based decoding lattice where each translation lattice node is augmented with syntactic language model state $\tilde{\boldsymbol{\tau}}_t$.

In phrase-based translation, many translation lattice nodes represent multi-word target language phrases. For such translation lattice nodes, δ will be called once for each newly hypothesized target language word in the node. Only the final syntactic language model state in such sequences need be stored in the translation lattice node.

4 Incremental Bounded-Memory Parsing with a Time Series Model

Having defined the framework by which any incremental parser may be incorporated into phrase-based translation, we now formally define a specific incremental parser for use in our experiments.

The parser must process target language words incrementally as the phrase-based decoder adds hypotheses to the translation lattice. To facilitate this incremental processing, ordinary phrase-structure trees can be transformed into right-corner recur-

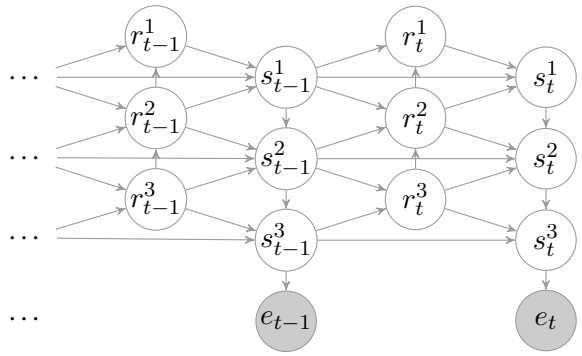


Figure 4: Graphical representation of the dependency structure in a standard Hierarchic Hidden Markov Model with $D = 3$ hidden levels that can be used to parse syntax. Circles denote random variables, and edges denote conditional dependencies. Shaded circles denote variables with observed values.

sive phrase structure trees using the tree transforms in Schuler et al. (2010). Constituent nonterminals in right-corner transformed trees take the form of *incomplete constituents* $c_\eta/c_{\eta\iota}$ consisting of an 'active' constituent c_η lacking an 'awaited' constituent $c_{\eta\iota}$ yet to come, similar to non-constituent categories in a Combinatory Categorial Grammar (Ades and Steedman, 1982; Steedman, 2000). As an example, the parser might consider VP/NN as a possible category for input "meets the".

A sample phrase structure tree is shown before and after the right-corner transform in Figures 2 and 3. Our parser operates over a right-corner transformed probabilistic context-free grammar (PCFG). Parsing runs in linear time on the length of the input. This model of incremental parsing is implemented as a Hierarchical Hidden Markov Model (HHMM) (Murphy and Paskin, 2001), and is equivalent to a probabilistic pushdown automaton with a bounded pushdown store. The parser runs in O(n) time, where n is the number of words in the input. This model is shown graphically in Figure 4 and formally defined in §4.1 below.

The incremental parser assigns a probability (Eq. 5) for a partial target language hypothesis, using a bounded store of incomplete constituents $c_\eta/c_{\eta\iota}$. The phrase-based decoder uses this probability value as the syntactic language model feature score.

4.1 Formal Parsing Model: Scoring Partial Translation Hypotheses

This model is essentially an extension of an HHMM, which obtains a most likely sequence of hidden store states, $\hat{s}^{1..D}_{1..T}$, of some length T and some maximum depth D, given a sequence of observed tokens (e.g. generated target language words), $e_{1..T}$, using HHMM state transition model θ_A and observation symbol model θ_B (Rabiner, 1990):

$$\hat{s}^{1..D}_{1..T} \stackrel{\text{def}}{=} \underset{s^{1..D}_{1..T}}{\operatorname{argmax}} \prod_{t=1}^{T} \mathsf{P}_{\theta_\text{A}}(s^{1..D}_t \mid s^{1..D}_{t-1}) \cdot \mathsf{P}_{\theta_\text{B}}(e_t \mid s^{1..D}_t) \tag{8}$$

The HHMM parser is equivalent to a probabilistic pushdown automaton with a bounded pushdown store. The model generates each successive store (using store model θ_S) only after considering whether each nested sequence of incomplete constituents has completed and reduced (using reduction model θ_R):

$$\begin{aligned}\mathsf{P}_{\theta_\text{A}}(s^{1..D}_t \mid s^{1..D}_{t-1}) \stackrel{\text{def}}{=} \\ \sum_{r^1_t..r^D_t} \prod_{d=1}^{D} & \mathsf{P}_{\theta_\text{R}}(r^d_t \mid r^{d+1}_t s^d_{t-1} s^{d-1}_{t-1}) \\ & \cdot \mathsf{P}_{\theta_\text{S}}(s^d_t \mid r^{d+1}_t r^d_t\, s^d_{t-1} s^{d-1}_t)\end{aligned} \tag{9}$$

Store elements are defined to contain only the active (c_η) and awaited ($c_{\eta\iota}$) constituent categories necessary to compute an incomplete constituent probability:

$$s^d_t \stackrel{\text{def}}{=} \langle c_\eta, c_{\eta\iota} \rangle \tag{10}$$

Reduction states are defined to contain only the complete constituent category $c_{r^d_t}$ necessary to compute an inside likelihood probability, as well as a flag $f_{r^d_t}$ indicating whether a reduction has taken place (to end a sequence of incomplete constituents):

$$r^d_t \stackrel{\text{def}}{=} \langle c_{r^d_t}, f_{r^d_t} \rangle \tag{11}$$

The model probabilities for these store elements and reduction states can then be defined (from Murphy and Paskin 2001) to expand a new incomplete constituent after a reduction has taken place ($f_{r^d_t} = 1$; using depth-specific store state expansion model $\theta_{\text{S-E},d}$), transition along a sequence of store elements

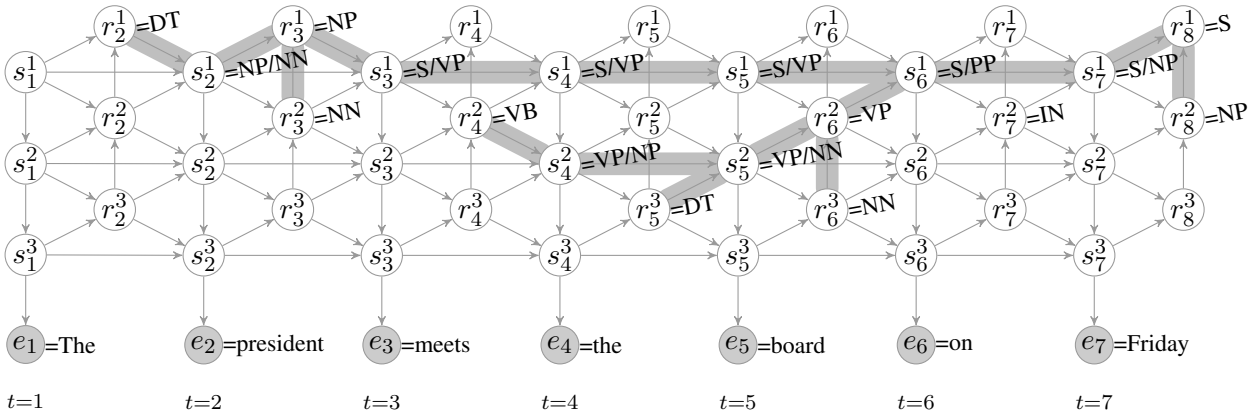


Figure 5: Graphical representation of the Hierarchic Hidden Markov Model after parsing input sentence *The president meets the board on Friday*. The shaded path through the parse lattice illustrates the recognized right-corner tree structure of Figure 3.

if no reduction has taken place ($f_{r_t^d}=0$; using depth-specific store state transition model $\theta_{\text{S-T},d}$): [2]

$$\mathsf{P}_{\theta_{\text{S}}}(s_t^d \mid r_t^{d+1} r_t^d \; s_{t-1}^d s_t^{d-1}) \stackrel{\text{def}}{=} \begin{cases} \text{if } f_{r_t^{d+1}}=1,\; f_{r_t^d}=1 : \mathsf{P}_{\theta_{\text{S-E},d}}(s_t^d \mid s_t^{d-1}) \\ \text{if } f_{r_t^{d+1}}=1,\; f_{r_t^d}=0 : \mathsf{P}_{\theta_{\text{S-T},d}}(s_t^d \mid r_t^{d+1} r_t^d \; s_{t-1}^d s_t^{d-1}) \\ \text{if } f_{r_t^{d+1}}=0,\; f_{r_t^d}=0 : [\![s_t^d = s_{t-1}^d]\!] \end{cases} \tag{12}$$

and possibly reduce a store element (terminate a sequence) if the store state below it has reduced ($f_{r_t^{d+1}}=1$; using depth-specific reduction model $\theta_{\text{R},d}$):

$$\mathsf{P}_{\theta_{\text{R}}}(r_t^d \mid r_t^{d+1} s_{t-1}^d s_{t-1}^{d-1}) \stackrel{\text{def}}{=} \begin{cases} \text{if } f_{r_t^{d+1}}=0 : [\![r_t^d = \mathbf{r}_\perp]\!] \\ \text{if } f_{r_t^{d+1}}=1 : \mathsf{P}_{\theta_{\text{R},d}}(r_t^d \mid r_t^{d+1} \; s_{t-1}^d \; s_{t-1}^{d-1}) \end{cases} \tag{13}$$

where $\mathbf{r}_\perp$ is a null state resulting from the failure of an incomplete constituent to complete, and constants are defined for the edge conditions of s_t^0 and r_t^{D+1}. Figure 5 illustrates this model in action.

These pushdown automaton operations are then refined for right-corner parsing (Schuler, 2009), distinguishing *active* transitions (model $\theta_{\text{S-T-A},d}$, in which an incomplete constituent is completed, but not reduced, and then immediately expanded to a new incomplete constituent in the same store element) from *awaited* transitions (model $\theta_{\text{S-T-W},d}$, which involve no completion):

$$\mathsf{P}_{\theta_{\text{S-T},d}}(s_t^d \mid r_t^{d+1} r_t^d \; s_{t-1}^d s_t^{d-1}) \stackrel{\text{def}}{=} \begin{cases} \text{if } r_t^d \neq \mathbf{r}_\perp : \mathsf{P}_{\theta_{\text{S-T-A},d}}(s_t^d \mid s_t^{d-1} r_t^d) \\ \text{if } r_t^d = \mathbf{r}_\perp : \mathsf{P}_{\theta_{\text{S-T-W},d}}(s_t^d \mid s_{t-1}^d r_t^{d+1}) \end{cases} \tag{14}$$

$$\mathsf{P}_{\theta_{\text{R},d}}(r_t^d \mid r_t^{d+1} s_{t-1}^d s_{t-1}^{d-1}) \stackrel{\text{def}}{=} \begin{cases} \text{if } c_{r_t^{d+1}} \neq x_t : [\![r_t^d = \mathbf{r}_\perp]\!] \\ \text{if } c_{r_t^{d+1}} = x_t : \mathsf{P}_{\theta_{\text{R-R},d}}(r_t^d \mid s_{t-1}^d s_{t-1}^{d-1}) \end{cases} \tag{15}$$

These HHMM right-corner parsing operations are then defined in terms of branch- and depth-specific PCFG probabilities $\theta_{\text{G-R},d}$ and $\theta_{\text{G-L},d}$: [3]

$$\mathsf{E}_{\theta_{\text{G-RL}^*,d}}(c_\eta \xrightarrow{0} c_{\eta 0} \ldots) \stackrel{\text{def}}{=} \sum_{c_{\eta 1}} \mathsf{P}_{\theta_{\text{G-R},d}}(c_\eta \rightarrow c_{\eta 0} \; c_{\eta 1}) \tag{16}$$

$$\mathsf{E}_{\theta_{\text{G-RL}^*,d}}(c_\eta \xrightarrow{k} c_{\eta 0^k 0} \ldots) \stackrel{\text{def}}{=} \sum_{c_{\eta 0^k}} \mathsf{E}_{\theta_{\text{G-RL}^*,d}}(c_\eta \xrightarrow{k-1} c_{\eta 0^k} \ldots) \cdot \sum_{c_{\eta 0^k 1}} \mathsf{P}_{\theta_{\text{G-L},d}}(c_{\eta 0^k} \rightarrow c_{\eta 0^k 0} \; c_{\eta 0^k 1}) \tag{17}$$

$$\mathsf{E}_{\theta_{\text{G-RL}^*,d}}(c_\eta \xrightarrow{*} c_{\eta\iota} \ldots) \stackrel{\text{def}}{=} \sum_{k=0}^{\infty} \mathsf{E}_{\theta_{\text{G-RL}^*,d}}(c_\eta \xrightarrow{k} c_{\eta\iota} \ldots) \tag{18}$$

$$\mathsf{E}_{\theta_{\text{G-RL}^*,d}}(c_\eta \xrightarrow{+} c_{\eta\iota} \ldots) \stackrel{\text{def}}{=} \mathsf{E}_{\theta_{\text{G-RL}^*,d}}(c_\eta \xrightarrow{*} c_{\eta\iota} \ldots) - \mathsf{E}_{\theta_{\text{G-RL}^*,d}}(c_\eta \xrightarrow{0} c_{\eta\iota} \ldots) \tag{19}$$

[2] An indicator function $[\![\cdot]\!]$ is used to denote deterministic probabilities: $[\![\phi]\!] = 1$ if ϕ is true, 0 otherwise.

[3] Model probabilities are also defined in terms of left-progeny probability distribution $\mathsf{E}_{\theta_{\text{G-RL}^*,d}}$ which is itself defined in terms of PCFG probabilities:

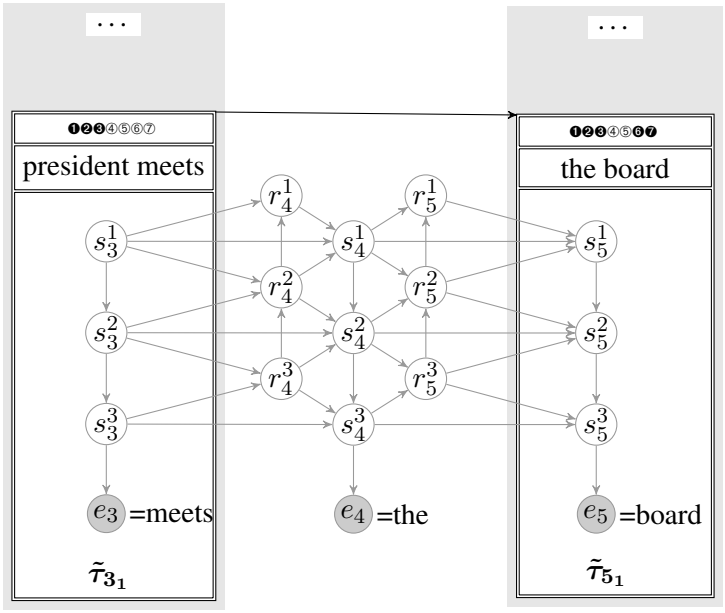


Figure 6: A hypothesis in the phrase-based decoding lattice from Figure 1 is expanded using translation option *the board* of source phrase *den Vorstand*. Syntactic language model state $\tilde{\boldsymbol{\tau}}_{\mathbf{3_1}}$ contains random variables $s_3^{1..3}$; likewise $\tilde{\boldsymbol{\tau}}_{\mathbf{5_1}}$ contains $s_5^{1..3}$. The intervening random variables $r_4^{1..3}$, $s_4^{1..3}$, and $r_5^{1..3}$ are calculated by transition function δ (Eq. 6, as defined by §4.1), but are not stored. Observed random variables ($e_3..e_5$) are shown for clarity, but are not explicitly stored in any syntactic language model state.

- for expansions:

$$\mathsf{P}_{\theta_{\text{S-E},d}}(\langle c_{\eta\iota}, c'_{\eta\iota}\rangle \mid \langle -, c_\eta\rangle) \stackrel{\text{def}}{=} \mathsf{E}_{\theta_{\text{G-RL}^*,d}}(c_\eta \xrightarrow{*} c_{\eta\iota}\ ...) \cdot [\![x_{\eta\iota} = c'_{\eta\iota} = c_{\eta\iota}]\!] \quad (20)$$

- for awaited transitions:

$$\mathsf{P}_{\theta_{\text{S-T-W},d}}(\langle c_\eta, c_{\eta\iota 1}\rangle \mid \langle c'_\eta, c_{\eta\iota}\rangle\ c_{\eta\iota 0}) \stackrel{\text{def}}{=} [\![c_\eta = c'_\eta]\!] \cdot \frac{\mathsf{P}_{\theta_{\text{G-R},d}}(c_{\eta\iota} \rightarrow c_{\eta\iota 0}\ c_{\eta\iota 1})}{\mathsf{E}_{\theta_{\text{G-RL}^*,d}}(c_{\eta\iota} \xrightarrow{0} c_{\eta\iota 0}\ ...)} \quad (21)$$

- for active transitions:

$$\mathsf{P}_{\theta_{\text{S-T-A},d}}(\langle c_{\eta\iota}, c_{\eta\iota 1}\rangle \mid \langle -, c_\eta\rangle\ c_{\eta\iota 0}) \stackrel{\text{def}}{=} \frac{\mathsf{E}_{\theta_{\text{G-RL}^*,d}}(c_\eta \xrightarrow{*} c_{\eta\iota}\ ...) \cdot \mathsf{P}_{\theta_{\text{G-L},d}}(c_{\eta\iota} \rightarrow c_{\eta\iota 0}\ c_{\eta\iota 1})}{\mathsf{E}_{\theta_{\text{G-RL}^*,d}}(c_\eta \xrightarrow{+} c_{\eta\iota 0}\ ...)} \quad (22)$$

- for cross-element reductions:

$$\mathsf{P}_{\theta_{\text{R-R},d}}(c_{\eta\iota}, \mathbf{1} \mid \langle -, c_\eta\rangle\ \langle c'_{\eta\iota}, -\rangle) \stackrel{\text{def}}{=} [\![c_{\eta\iota} = c'_{\eta\iota}]\!] \cdot \frac{\mathsf{E}_{\theta_{\text{G-RL}^*,d}}(c_\eta \xrightarrow{0} c_{\eta\iota}\ ...)}{\mathsf{E}_{\theta_{\text{G-RL}^*,d}}(c_\eta \xrightarrow{*} c_{\eta\iota}\ ...)} \quad (23)$$

- for in-element reductions:

$$\mathsf{P}_{\theta_{\text{R-R},d}}(c_{\eta\iota}, \mathbf{0} \mid \langle -, c_\eta\rangle\ \langle c'_{\eta\iota}, -\rangle) \stackrel{\text{def}}{=} [\![c_{\eta\iota} = c'_{\eta\iota}]\!] \cdot \frac{\mathsf{E}_{\theta_{\text{G-RL}^*,d}}(c_\eta \xrightarrow{+} c_{\eta\iota}\ ...)}{\mathsf{E}_{\theta_{\text{G-RL}^*,d}}(c_\eta \xrightarrow{*} c_{\eta\iota}\ ...)} \quad (24)$$

We use the parser implementation of (Schuler, 2009; Schuler et al., 2010).

5 Phrase Based Translation with an Incremental Syntactic Language Model

The phrase-based decoder is augmented by adding additional state data to each hypothesis in the de-

coder's hypothesis stacks. Figure 1 illustrates an excerpt from a standard phrase-based translation lattice. Within each decoder stack t, each hypothesis h is augmented with a syntactic language model state $\tilde{\boldsymbol{\tau}}_{\boldsymbol{t_h}}$. Each syntactic language model state is a random variable store, containing a slice of random variables from the HHMM. Specifically, $\tilde{\boldsymbol{\tau}}_{\boldsymbol{t_h}}$ contains those random variables $s_t^{1..D}$ that maintain distributions over syntactic elements.

By maintaining these syntactic random variable stores, each hypothesis has access to the current language model probability for the partial translation ending at that hypothesis, as calculated by an incremental syntactic language model defined by the HHMM. Specifically, the random variable store at hypothesis h provides $\mathsf{P}(\tilde{\boldsymbol{\tau}}_{\boldsymbol{t_h}}) = \mathsf{P}(e_{1..t}^h, s_{1..t}^{1..D})$, where $e_{1..t}^h$ is the sequence of words in a partial hypothesis ending at h which contains t target words, and where there are D syntactic random variables in each random variable store (Eq. 5).

During stack decoding, the phrase-based decoder progressively constructs new hypotheses by extending existing hypotheses. New hypotheses are placed in appropriate hypothesis stacks. In the simplest case, a new hypothesis extends an existing hypothesis by exactly one target word. As the new hypothesis is constructed by extending an existing stack element, the store and reduction state random variables are processed, along with the newly hypothesized word. This results in a new store of syntactic random variables (Eq. 6) that are associated with the new stack element.

When a new hypothesis extends an existing hypothesis by more than one word, this process is first carried out for the first new word in the hypothesis. It is then repeated for the remaining words in the hypothesis extension. Once the final word in the hypothesis has been processed, the resulting random variable store is associated with that hypothesis. The random variable stores created for the non-final words in the extending hypothesis are discarded, and need not be explicitly retained.

Figure 6 illustrates this process, showing how a syntactic language model state $\tilde{\boldsymbol{\tau}}_{\mathbf{5_1}}$ in a phrase-based decoding lattice is obtained from a previous syntactic language model state $\tilde{\boldsymbol{\tau}}_{\mathbf{3_1}}$ (from Figure 1) by parsing the target language words from a phrase-based translation option.

LM	In-domain WSJ 23 ppl	Out-of-domain ur-en dev ppl
WSJ 1-gram	1973.57	3581.72
WSJ 2-gram	349.18	1312.61
WSJ 3-gram	262.04	1264.47
WSJ 4-gram	244.12	1261.37
WSJ 5-gram	**232.08**	1261.90
WSJ HHMM	384.66	**529.41**
Interpolated WSJ 5-gram + HHMM	***209.13***	225.48
Giga 5-gram	258.35	312.28
Interp. Giga 5-gr + WSJ HHMM	222.39	***123.10***
Interp. Giga 5-gr + WSJ 5-gram	174.88	321.05

Figure 7: Average per-word perplexity values. HHMM was run with beam size of 2000. **Bold** indicates best single-model results for LMs trained on WSJ sections 2-21. Best overall in ***italics***.

Our syntactic language model is integrated into the current version of Moses (Koehn et al., 2007).

6 Results

As an initial measure to compare language models, average per-word perplexity, ppl, reports how surprised a model is by test data. Equation 25 calculates ppl using log base b for a test set of T tokens.

$$ppl = b^{\frac{-log_b \mathsf{P}(e_1 \ldots e_T)}{T}} \quad (25)$$

We trained the syntactic language model from §4 (HHMM) and an interpolated n-gram language model with modified Kneser-Ney smoothing (Chen and Goodman, 1998); models were trained on sections 2-21 of the Wall Street Journal (WSJ) treebank (Marcus et al., 1993). The HHMM outperforms the n-gram model in terms of out-of-domain test set perplexity when trained on the same WSJ data; the best perplexity results for in-domain and out-of-domain test sets[4] are found by interpolating

[4]In-domain is WSJ Section 23. Out-of-domain are the English reference translations of the dev section , set aside in (Baker et al., 2009) for parameter tuning, of the NIST Open MT 2008 Urdu-English task.

Sentence length	Moses	+HHMM beam=50	+HHMM beam=2000
10	0.21	533	1143
20	0.53	1193	2562
30	0.85	1746	3749
40	1.13	2095	4588

Figure 8: Mean per-sentence decoding time (in seconds) for dev set using Moses with and without syntactic language model. HHMM parser beam sizes are indicated for the syntactic LM.

HHMM and n-gram LMs (Figure 7). To show the effects of training an LM on more data, we also report perplexity results on the 5-gram LM trained for the GALE Arabic-English task using the English Gigaword corpus. In all cases, including the HHMM significantly reduces perplexity.

We trained a phrase-based translation model on the full NIST Open MT08 Urdu-English translation model using the full training data. We trained the HHMM and n-gram LMs on the WSJ data in order to make them as similar as possible. During tuning, Moses was first configured to use just the n-gram LM, then configured to use both the n-gram LM and the syntactic HHMM LM. MERT consistently assigned positive weight to the syntactic LM feature, typically slightly less than the n-gram LM weight.

In our integration with Moses, incorporating a syntactic language model dramatically slows the decoding process. Figure 8 illustrates a slowdown around three orders of magnitude. Although speed remains roughly linear to the size of the source sentence (ruling out exponential behavior), it is with an extremely large constant time factor. Due to this slowdown, we tuned the parameters using a constrained dev set (only sentences with 1-20 words), and tested using a constrained devtest set (only sentences with 1-20 words). Figure 9 shows a statistically significant improvement to the BLEU score when using the HHMM and the n-gram LMs together on this reduced test set.

Moses LM(s)	BLEU
n-gram only	18.78
HHMM + n-gram	**19.78**

Figure 9: Results for Ur-En devtest (only sentences with 1-20 words) with HHMM beam size of 2000 and Moses settings of distortion limit 10, stack size 200, and ttable_limit 20.

7 Discussion

This paper argues that incremental syntactic languages models are a straightforward and appropriate algorithmic fit for incorporating syntax into phrase-based statistical machine translation, since both process sentences in an incremental left-to-right fashion. This means incremental syntactic LM scores can be calculated during the decoding process, rather than waiting until a complete sentence is posited, which is typically necessary in top-down or bottom-up parsing.

We provided a rigorous formal definition of incremental syntactic languages models, and detailed what steps are necessary to incorporate such LMs into phrase-based decoding. We integrated an incremental syntactic language model into Moses. The translation quality significantly improved on a constrained task, and the perplexity improvements suggest that interpolating between n-gram and syntactic LMs may hold promise on larger data sets.

The use of very large n-gram language models is typically a key ingredient in the best-performing machine translation systems (Brants et al., 2007). Our n-gram model trained only on WSJ is admittedly small. Our future work seeks to incorporate large-scale n-gram language models in conjunction with incremental syntactic language models.

The added decoding time cost of our syntactic language model is very high. By increasing the beam size and distortion limit of the baseline system, future work may examine whether a baseline system with comparable runtimes can achieve comparable translation quality.

A more efficient implementation of the HHMM parser would speed decoding and make more extensive and conclusive translation experiments possible. Various additional improvements could include caching the HHMM LM calculations, and exploiting properties of the right-corner transform that limit the number of decisions between successive time steps.

References

Anne Abeillé, Yves Schabes, and Aravind K. Joshi. 1990. Using lexicalized tree adjoining grammars for machine translation. In *Proceedings of the 13th International Conference on Computational Linguistics*.

Anthony E. Ades and Mark Steedman. 1982. On the order of words. *Linguistics and Philosophy*, 4:517–558.

Kathy Baker, Steven Bethard, Michael Bloodgood, Ralf Brown, Chris Callison-Burch, Glen Coppersmith, Bonnie Dorr, Wes Filardo, Kendall Giles, Anni Irvine, Mike Kayser, Lori Levin, Justin Martineau, Jim Mayfield, Scott Miller, Aaron Phillips, Andrew Philpot, Christine Piatko, Lane Schwartz, and David Zajic. 2009. Semantically informed machine translation (SIMT). SCALE summer workshop final report, Human Language Technology Center Of Excellence.

Alexandra Birch, Miles Osborne, and Philipp Koehn. 2007. CCG supertags in factored statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 9–16.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Peter Brown, John Cocke, Stephen Della Pietra, Vincent Della Pietra, Frederick Jelinek, John Lafferty, Robert Mercer, and Paul Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Eugene Charniak, Kevin Knight, and Kenji Yamada. 2003. Syntax-based language models for statistical machine translation. In *Proceedings of the Ninth Machine Translation Summit of the International Association for Machine Translation*.

Ciprian Chelba and Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 225–231.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech and Language*, 14(4):283–332.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report, Harvard University.

Colin Cherry. 2008. Cohesive phrase-based decoding for statistical machine translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 72–80.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 263–270.

David Chiang. 2010. Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1443–1452.

John Cocke and Jacob Schwartz. 1970. Programming languages and their compilers. Technical report, Courant Institute of Mathematical Sciences, New York University.

Michael Collins, Brian Roark, and Murat Saraclar. 2005. Discriminative syntactic language modeling for speech recognition. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 507–514.

Brooke Cowan, Ivona Kučerová, and Michael Collins. 2006. A discriminative model for tree-to-tree translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 232–241.

Steve DeNeefe and Kevin Knight. 2009. Synchronous tree adjoining machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 727–736.

Steve DeNeefe, Kevin Knight, Wei Wang, and Daniel Marcu. 2007. What can syntax-based MT learn from phrase-based MT? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 755–763.

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

Jay Earley. 1968. *An efficient context-free parsing algorithm*. Ph.D. thesis, Department of Computer Science, Carnegie Mellon University.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, pages 205–208.

Michel Galley and Christopher D. Manning. 2009. Quadratic-time dependency parsing for machine translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 773–781.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In

Daniel Marcu Susan Dumais and Salim Roukos, editors, *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 273–280.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 961–968.

Niyu Ge. 2010. A direct syntax-driven reordering model for phrase-based machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 849–857.

Daniel Gildea. 2003. Loosely tree-based alignment for machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 80–87.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 105–112.

Hany Hassan, Khalil Sima'an, and Andy Way. 2007. Supertagged phrase-based statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 288–295.

James Henderson. 2004. Lookahead in deterministic left-corner parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 26–33.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 273–283.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the 7th Biennial conference of the Association for Machine Translation in the Americas*.

Kenji Imamura, Hideo Okuma, Taro Watanabe, and Eiichiro Sumita. 2004. Example-based machine translation based on syntactic transfer with statistical models. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 99–105.

Frederick Jelinek. 1969. Fast sequential decoding algorithm using a stack. *IBM Journal of Research and Development*, pages 675–685.

T. Kasami. 1965. An efficient recognition and syntax analysis algorithm for context free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Laboratory.

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133.

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

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 609–616.

Yang Liu, Yun Huang, Qun Liu, and Shouxun Lin. 2007. Forest-to-string statistical translation rules. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 704–711.

Yang Liu, Yajuan Lü, and Qun Liu. 2009. Improving tree-to-tree translation with packed forests. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 558–566.

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

I. Dan Melamed. 2004. Statistical machine translation by parsing. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pages 653–660.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 206–214.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 192–199.

Kevin P. Murphy and Mark A. Paskin. 2001. Linear time inference in hierarchical HMMs. In *Proceedings of Neural Information Processing Systems*, pages 833–840.

Rebecca Nesson, Stuart Shieber, and Alexander Rush. 2006. Induction of probabilistic synchronous tree-insertion grammars for machine translation. In *Proceedings of the 7th Biennial conference of the Association for Machine Translation in the Americas*, pages 128–137.

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

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 161–168.

Matt Post and Daniel Gildea. 2008. Parsers as language models for statistical machine translation. In *Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas*, pages 172–181.

Matt Post and Daniel Gildea. 2009. Language modeling with tree substitution grammars. In *NIPS workshop on Grammar Induction, Representation of Language, and Language Learning*.

Arjen Poutsma. 1998. Data-oriented translation. In *Ninth Conference of Computational Linguistics in the Netherlands*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 271–279.

Lawrence R. Rabiner. 1990. A tutorial on hidden Markov models and selected applications in speech recognition. *Readings in speech recognition*, 53(3):267–296.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

William Schuler, Samir AbdelRahman, Tim Miller, and Lane Schwartz. 2010. Broad-coverage incremental parsing using human-like memory constraints. *Computational Linguistics*, 36(1):1–30.

William Schuler. 2009. Positive results for parsing with a bounded stack using a model-based right-corner transform. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 344–352.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 577–585.

Stuart M. Shieber and Yves Schabes. 1990. Synchronous tree adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics*.

Stuart M. Shieber. 2004. Synchronous grammars as tree transducers. In *Proceedings of the Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms*.

Mark Steedman. 2000. *The syntactic process*. MIT Press/Bradford Books, Cambridge, MA.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530.

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

Min Zhang, Hongfei Jiang, Ai Ti Aw, Jun Sun, Seng Li, and Chew Lim Tan. 2007. A tree-to-tree alignment-based model for statistical machine translation. In *Proceedings of the 11th Machine Translation Summit of the International Association for Machine Translation*, pages 535–542.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 138–141.

An Unsupervised Model for Joint Phrase Alignment and Extraction

Graham Neubig[1,2] **Taro Watanabe**[2], **Eiichiro Sumita**[2], **Shinsuke Mori**[1], **Tatsuya Kawahara**[1]
[1]Graduate School of Informatics, Kyoto University
Yoshida Honmachi, Sakyo-ku, Kyoto, Japan
[2]National Institute of Information and Communication Technology
3-5 Hikari-dai, Seika-cho, Soraku-gun, Kyoto, Japan

Abstract

We present an unsupervised model for joint phrase alignment and extraction using nonparametric Bayesian methods and inversion transduction grammars (ITGs). The key contribution is that phrases of many granularities are included directly in the model through the use of a novel formulation that memorizes phrases generated not only by terminal, but also non-terminal symbols. This allows for a completely probabilistic model that is able to create a phrase table that achieves competitive accuracy on phrase-based machine translation tasks directly from unaligned sentence pairs. Experiments on several language pairs demonstrate that the proposed model matches the accuracy of traditional two-step word alignment/phrase extraction approach while reducing the phrase table to a fraction of the original size.

1 Introduction

The training of translation models for phrase-based statistical machine translation (SMT) systems (Koehn et al., 2003) takes unaligned bilingual training data as input, and outputs a scored table of phrase pairs. This phrase table is traditionally generated by going through a pipeline of two steps, first generating word (or minimal phrase) alignments, then extracting a phrase table that is consistent with these alignments.

However, as DeNero and Klein (2010) note, this two step approach results in word alignments that are not optimal for the final task of generating phrase tables that are used in translation. As a solution to this, they proposed a supervised discriminative model that performs joint word alignment and phrase extraction, and found that joint estimation of word alignments and extraction sets improves both word alignment accuracy and translation results.

In this paper, we propose the first *unsupervised* approach to joint alignment and extraction of phrases at multiple granularities. This is achieved by constructing a generative model that includes phrases at many levels of granularity, from minimal phrases all the way up to full sentences. The model is similar to previously proposed phrase alignment models based on inversion transduction grammars (ITGs) (Cherry and Lin, 2007; Zhang et al., 2008; Blunsom et al., 2009), with one important change: ITG symbols and phrase pairs are generated in the opposite order. In traditional ITG models, the branches of a biparse tree are generated from a non-terminal distribution, and each leaf is generated by a word or phrase pair distribution. As a result, only minimal phrases are directly included in the model, while larger phrases must be generated by heuristic extraction methods. In the proposed model, at each branch in the tree, we first attempt to generate a phrase pair from the phrase pair distribution, falling back to ITG-based divide and conquer strategy to generate phrase pairs that do not exist (or are given low probability) in the phrase distribution.

We combine this model with the Bayesian nonparametric Pitman-Yor process (Pitman and Yor, 1997; Teh, 2006), realizing ITG-based divide and conquer through a novel formulation where the Pitman-Yor process uses two copies of itself as a

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 632–641,
Portland, Oregon, June 19-24, 2011.

base measure. As a result of this modeling strategy, phrases of multiple granularities are generated, and thus memorized, by the Pitman-Yor process. This makes it possible to directly use probabilities of the phrase model as a replacement for the phrase table generated by heuristic extraction techniques.

Using this model, we perform machine translation experiments over four language pairs. We observe that the proposed joint phrase alignment and extraction approach is able to meet or exceed results attained by a combination of GIZA++ and heuristic phrase extraction with significantly smaller phrase table size. We also find that it achieves superior BLEU scores over previously proposed ITG-based phrase alignment approaches.

2 A Probabilistic Model for Phrase Table Extraction

The problem of SMT can be defined as finding the most probable target sentence $\mathbf{e}$ for the source sentence $\mathbf{f}$ given a parallel training corpus $\langle \mathcal{E}, \mathcal{F} \rangle$

$$\hat{\mathbf{e}} = \underset{\mathbf{e}}{\text{argmax}} P(\mathbf{e}|\mathbf{f}, \langle \mathcal{E}, \mathcal{F} \rangle).$$

We assume that there is a hidden set of parameters θ learned from the training data, and that $\mathbf{e}$ is conditionally independent from the training corpus given θ. We take a Bayesian approach, integrating over all possible values of the hidden parameters:

$$P(\mathbf{e}|\mathbf{f}, \langle \mathcal{E}, \mathcal{F} \rangle) = \int_{\theta} P(\mathbf{e}|\mathbf{f}, \theta) P(\theta|\langle \mathcal{E}, \mathcal{F} \rangle). \quad (1)$$

If θ takes the form of a scored phrase table, we can use traditional methods for phrase-based SMT to find $P(\mathbf{e}|\mathbf{f}, \theta)$ and concentrate on creating a model for $P(\theta|\langle \mathcal{E}, \mathcal{F} \rangle)$. We decompose this posterior probability using Bayes law into the corpus likelihood and parameter prior probabilities

$$P(\theta|\langle \mathcal{E}, \mathcal{F} \rangle) \propto P(\langle \mathcal{E}, \mathcal{F} \rangle|\theta) P(\theta).$$

In Section 3 we describe an existing method, and in Section 4 we describe our proposed method for modeling these two probabilities.

3 Flat ITG Model

There has been a significant amount of work in many-to-many alignment techniques (Marcu and Wong (2002), DeNero et al. (2008), *inter alia*), and in particular a number of recent works (Cherry and Lin, 2007; Zhang et al., 2008; Blunsom et al., 2009) have used the formalism of inversion transduction grammars (ITGs) (Wu, 1997) to learn phrase alignments. By slightly limit reordering of words, ITGs make it possible to exactly calculate probabilities of phrasal alignments in polynomial time, which is a computationally hard problem when arbitrary reordering is allowed (DeNero and Klein, 2008).

The traditional flat ITG generative probability for a particular phrase (or sentence) pair $P_{flat}(\langle e, f \rangle; \theta_x, \theta_t)$ is parameterized by a phrase table θ_t and a symbol distribution θ_x. We use the following generative story as a representative of the flat ITG model.

1. Generate symbol x from the multinomial distribution $P_x(x; \theta_x)$. x can take the values TERM, REG, or INV.

2. According to the x take the following actions.

   (a) If $x =$ TERM, generate a phrase pair from the phrase table $P_t(\langle e, f \rangle; \theta_t)$.

   (b) If $x =$ REG, a regular ITG rule, generate phrase pairs $\langle e_1, f_1 \rangle$ and $\langle e_2, f_2 \rangle$ from P_{flat}, and concatenate them into a single phrase pair $\langle e_1 e_2, f_1 f_2 \rangle$.

   (c) If $x =$ INV, an inverted ITG rule, follows the same process as (b), but concatenate f_1 and f_2 in reverse order $\langle e_1 e_2, f_2 f_1 \rangle$.

By taking the product of P_{flat} over every sentence in the corpus, we are able to calculate the likelihood

$$P(\langle \mathcal{E}, \mathcal{F} \rangle|\theta) = \prod_{\langle e,f \rangle \in \langle \mathcal{E}, \mathcal{F} \rangle} P_{flat}(\langle e, f \rangle; \theta).$$

We will refer to this model as FLAT.

3.1 Bayesian Modeling

While the previous formulation can be used as-is in maximum likelihood training, this leads to a degenerate solution where every sentence is memorized as a single phrase pair. Zhang et al. (2008) and others propose dealing with this problem by putting a prior probability $P(\theta_x, \theta_t)$ on the parameters.

We assign θ_x a Dirichlet prior[1], and assign the phrase table parameters θ_t a prior using the Pitman-Yor process (Pitman and Yor, 1997; Teh, 2006), which is a generalization of the Dirichlet process prior used in previous research. It is expressed as

$$\theta_t \sim PY(d, s, P_{base}) \quad (2)$$

where d is the discount parameter, s is the strength parameter, and P_{base} is the base measure. The discount d is subtracted from observed counts, and when it is given a large value (close to one), less frequent phrase pairs will be given lower relative probability than more common phrase pairs. The strength s controls the overall sparseness of the distribution, and when it is given a small value the distribution will be sparse. P_{base} is the prior probability of generating a particular phrase pair, which we describe in more detail in the following section.

Non-parametric priors are well suited for modeling the phrase distribution because every time a phrase is generated by the model, it is "memorized" and given higher probability. Because of this, common phrase pairs are more likely to be re-used (the *rich-get-richer* effect), which results in the induction of phrase tables with fewer, but more helpful phrases. It is important to note that only phrases generated by P_t are actually memorized and given higher probability by the model. In FLAT, only minimal phrases generated after P_x outputs the terminal symbol TERM are generated from P_t, and thus only minimal phrases are memorized by the model.

While the Dirichlet process is simply the Pitman-Yor process with $d = 0$, it has been shown that the discount parameter allows for more effective modeling of the long-tailed distributions that are often found in natural language (Teh, 2006). We confirmed in preliminary experiments (using the data described in Section 7) that the Pitman-Yor process with automatically adjusted parameters results in superior alignment results, outperforming the sparse Dirichlet process priors used in previous research[2]. The average gain across all data sets was approximately 0.8 BLEU points.

[1]The value of α had little effect on the results, so we arbitrarily set $\alpha = 1$.

[2]We put weak priors on s ($Gamma(\alpha = 2, \beta = 1)$) and d ($Beta(\alpha = 2, \beta = 2)$) for the Pitman-Yor process, and set $\alpha = 1^{-10}$ for the Dirichlet process.

3.2 Base Measure

P_{base} in Equation (2) indicates the prior probability of phrase pairs according to the model. By choosing this probability appropriately, we can incorporate prior knowledge of what phrases tend to be aligned to each other. We calculate P_{base} by first choosing whether to generate an unaligned phrase pair (where $|e| = 0$ or $|f| = 0$) according to a fixed probability p_u[3], then generating from P_{ba} for aligned phrase pairs, or P_{bu} for unaligned phrase pairs.

For P_{ba}, we adopt a base measure similar to that used by DeNero et al. (2008):

$$P_{ba}(\langle e, f \rangle) = M_0(\langle e, f \rangle) P_{pois}(|e|; \lambda) P_{pois}(|f|; \lambda)$$
$$M_0(\langle e, f \rangle) = (P_{m1}(f|e) P_{uni}(e) P_{m1}(e|f) P_{uni}(f))^{\frac{1}{2}}.$$

P_{pois} is the Poisson distribution with the average length parameter λ. As long phrases lead to sparsity, we set λ to a relatively small value to allow us to bias against overly long phrases[4]. P_{m1} is the word-based Model 1 (Brown et al., 1993) probability of one phrase given the other, which incorporates word-based alignment information as prior knowledge in the phrase translation probability. We take the geometric mean[5]of the Model 1 probabilities in both directions to encourage alignments that are supported by both models (Liang et al., 2006). It should be noted that while Model 1 probabilities are used, they are only soft constraints, compared with the hard constraint of choosing a single word alignment used in most previous phrase extraction approaches.

For P_{bu}, if g is the non-null phrase in e and f, we calculate the probability as follows:

$$P_{bu}(\langle e, f \rangle) = P_{uni}(g) P_{pois}(|g|; \lambda)/2.$$

Note that P_{bu} is divided by 2 as the probability is considering null alignments in both directions.

4 Hierarchical ITG Model

While in FLAT only minimal phrases were memorized by the model, as DeNero et al. (2008) note

[3]We choose 10^{-2}, 10^{-3}, or 10^{-10} based on which value gave the best accuracy on the development set.

[4]We tune λ to 1, 0.1, or 0.01 based on which value gives the best performance on the development set.

[5]The probabilities of the geometric mean do not add to one, but we found empirically that even when left unnormalized, this provided much better results than the using the arithmetic mean, which is more theoretically correct.

and we confirm in the experiments in Section 7, using only minimal phrases leads to inferior translation results for phrase-based SMT. Because of this, previous research has combined FLAT with heuristic phrase extraction, which exhaustively combines all adjacent phrases permitted by the word alignments (Och et al., 1999). We propose an alternative, fully statistical approach that directly models phrases at multiple granularities, which we will refer to as HIER. By doing so, we are able to do away with heuristic phrase extraction, creating a fully probabilistic model for phrase probabilities that still yields competitive results.

Similarly to FLAT, HIER assigns a probability $P_{hier}(\langle e, f\rangle; \theta_x, \theta_t)$ to phrase pairs, and is parameterized by a phrase table θ_t and a symbol distribution θ_x. The main difference from the generative story of the traditional ITG model is that symbols and phrase pairs are generated in the opposite order. While FLAT first generates branches of the derivation tree using P_x, then generates leaves using the phrase distribution P_t, HIER first attempts to generate the full sentence as a single phrase from P_t, then falls back to ITG-style derivations to cope with sparsity. We allow for this within the Bayesian ITG context by defining a new base measure P_{dac} ("divide-and-conquer") to replace P_{base} in Equation (2), resulting in the following distribution for θ_t.

$$\theta_t \sim PY(d, s, P_{dac}) \tag{3}$$

P_{dac} essentially breaks the generation of a single longer phrase into two generations of shorter phrases, allowing even phrase pairs for which $c(\langle e, f\rangle) = 0$ to be given some probability. The generative process of P_{dac}, similar to that of P_{flat} from the previous section, is as follows:

1. Generate symbol x from $P_x(x; \theta_x)$. x can take the values BASE, REG, or INV.

2. According to x take the following actions.

   (a) If $x =$ BASE, generate a new phrase pair directly from P_{base} of Section 3.2.

   (b) If $x =$ REG, generate $\langle e_1, f_1\rangle$ and $\langle e_2, f_2\rangle$ from P_{hier}, and concatenate them into a single phrase pair $\langle e_1 e_2, f_1 f_2\rangle$.

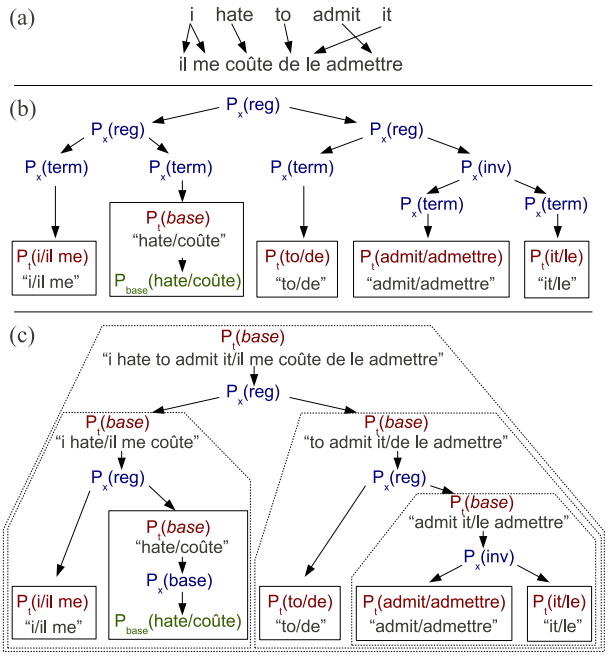


Figure 1: A word alignment (a), and its derivations according to FLAT (b), and HIER (c). Solid and dotted lines indicate minimal and non-minimal pairs respectively, and phrases are written under their corresponding instance of P_t. The pair hate/coûte is generated from P_{base}.

   (c) If $x =$ INV, follow the same process as (b), but concatenate f_1 and f_2 in reverse order $\langle e_1 e_2, f_2 f_1\rangle$.

A comparison of derivation trees for FLAT and HIER is shown in Figure 1. As previously described, FLAT first generates from the symbol distribution P_x, then from the phrase distribution P_t, while HIER generates directly from P_t, which falls back to divide-and-conquer based on P_x when necessary. It can be seen that while P_t in FLAT only generates minimal phrases, P_t in HIER generates (and thus memorizes) phrases at all levels of granularity.

4.1 Length-based Parameter Tuning

There are still two problems with HIER, one theoretical, and one practical. Theoretically, HIER contains itself as its base measure, and stochastic process models that include themselves as base measures are deficient, as noted in Cohen et al. (2010). Practically, while the Pitman-Yor process in HIER shares the parameters s and d over all phrase pairs in the model, long phrase pairs are much more sparse

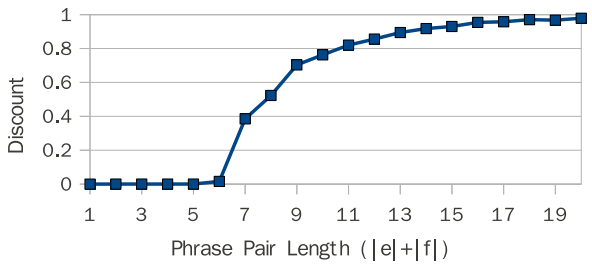


Figure 2: Learned discount values by phrase pair length.

than short phrase pairs, and thus it is desirable to appropriately adjust the parameters of Equation (2) according to phrase pair length.

In order to solve these problems, we reformulate the model so that each phrase length $l = |f|+|e|$ has its own phrase parameters $\theta_{t,l}$ and symbol parameters $\theta_{x,l}$, which are given separate priors:

$$\theta_{t,l} \sim PY(s, d, P_{dac,l})$$
$$\theta_{x,l} \sim Dirichlet(\alpha)$$

We will call this model HLEN.

The generative story is largely similar to HIER with a few minor changes. When we generate a sentence, we first choose its length l according to a uniform distribution over all possible sentence lengths

$$l \sim Uniform(1, L),$$

where L is the size $|e| + |f|$ of the longest sentence in the corpus. We then generate a phrase pair from the probability $P_{t,l}(\langle e, f \rangle)$ for length l. The base measure for HLEN is identical to that of HIER, with one minor change: when we fall back to two shorter phrases, we choose the length of the left phrase from $l_l \sim Uniform(1, l-1)$, set the length of the right phrase to $l_r = l - l_l$, and generate the smaller phrases from P_{t,l_l} and P_{t,l_r} respectively.

It can be seen that phrases at each length are generated from different distributions, and thus the parameters for the Pitman-Yor process will be different for each distribution. Further, as l_l and l_r must be smaller than l, $P_{t,l}$ no longer contains itself as a base measure, and is thus not deficient.

An example of the actual discount values learned in one of the experiments described in Section 7 is shown in Figure 2. It can be seen that, as expected, the discounts for short phrases are lower than those of long phrases. In particular, phrase pairs of length up to six (for example, $|e| = 3$, $|f| = 3$) are given discounts of nearly zero while larger phrases are more heavily discounted. We conjecture that this is related to the observation by Koehn et al. (2003) that using phrases where $\max(|e|, |f|) \leq 3$ cause significant improvements in BLEU score, while using larger phrases results in diminishing returns.

4.2 Implementation

Previous research has used a variety of sampling methods to learn Bayesian phrase based alignment models (DeNero et al., 2008; Blunsom et al., 2009; Blunsom and Cohn, 2010). All of these techniques are applicable to the proposed model, but we choose to apply the sentence-based blocked sampling of Blunsom and Cohn (2010), which has desirable convergence properties compared to sampling single alignments. As exhaustive sampling is too slow for practical purpose, we adopt the beam search algorithm of Saers et al. (2009), and use a probability beam, trimming spans where the probability is at least 10^{10} times smaller than that of the best hypothesis in the bucket.

One important implementation detail that is different from previous models is the management of phrase counts. As a phrase pair t_a may have been generated from two smaller component phrases t_b and t_c, when a sample containing t_a is removed from the distribution, it may also be necessary to decrement the counts of t_b and t_c as well. The Chinese Restaurant Process representation of P_t (Teh, 2006) lends itself to a natural and easily implementable solution to this problem. For each table representing a phrase pair t_a, we maintain not only the number of customers sitting at the table, but also the identities of phrases t_b and t_c that were originally used when generating the table. When the count of the table t_a is reduced to zero and the table is removed, the counts of t_b and t_c are also decremented.

5 Phrase Extraction

In this section, we describe both traditional heuristic phrase extraction, and the proposed model-based extraction method.

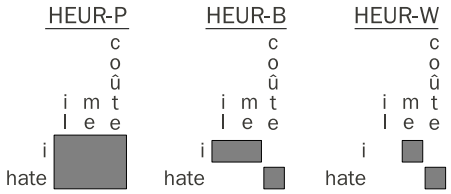


Figure 3: The phrase, block, and word alignments used in heuristic phrase extraction.

5.1 Heuristic Phrase Extraction

The traditional method for heuristic phrase extraction from word alignments exhaustively enumerates all phrases up to a certain length consistent with the alignment (Och et al., 1999). Five features are used in the phrase table: the conditional phrase probabilities in both directions estimated using maximum likelihood $P_{ml}(f|e)$ and $P_{ml}(e|f)$, lexical weighting probabilities (Koehn et al., 2003), and a fixed penalty for each phrase. We will call this heuristic extraction from word alignments HEUR-W. These word alignments can be acquired through the standard GIZA++ training regimen.

We use the combination of our ITG-based alignment with traditional heuristic phrase extraction as a second baseline. An example of these alignments is shown in Figure 3. In model HEUR-P, minimal phrases generated from P_t are treated as aligned, and we perform phrase extraction on these alignments. However, as the proposed models tend to align relatively large phrases, we also use two other techniques to create smaller alignment chunks that prevent sparsity. We perform regular sampling of the trees, but if we reach a minimal phrase generated from P_t, we continue traveling down the tree until we reach either a one-to-many alignment, which we will call HEUR-B as it creates alignments similar to the block ITG, or an at-most-one alignment, which we will call HEUR-W as it generates word alignments. It should be noted that forcing alignments smaller than the model suggests is only used for generating alignments for use in heuristic extraction, and does not affect the training process.

5.2 Model-Based Phrase Extraction

We also propose a method for phrase table extraction that directly utilizes the phrase probabilities $P_t(\langle e, f\rangle)$. Similarly to the heuristic phrase tables, we use conditional probabilities $P_t(f|e)$ and $P_t(e|f)$, lexical weighting probabilities, and a phrase penalty. Here, instead of using maximum likelihood, we calculate conditional probabilities directly from P_t probabilities:

$$P_t(f|e) = P_t(\langle e, f\rangle) / \sum_{\{\tilde{f}: c(\langle e,\tilde{f}\rangle)\geq 1\}} P_t(\langle e, \tilde{f}\rangle)$$

$$P_t(e|f) = P_t(\langle e, f\rangle) / \sum_{\{\tilde{e}: c(\langle \tilde{e},f\rangle)\geq 1\}} P_t(\langle \tilde{e}, f\rangle).$$

To limit phrase table size, we include only phrase pairs that are aligned at least once in the sample.

We also include two more features: the phrase pair joint probability $P_t(\langle e, f\rangle)$, and the average posterior probability of each span that generated $\langle e, f\rangle$ as computed by the inside-outside algorithm during training. We use the span probability as it gives a hint about the reliability of the phrase pair. It will be high for common phrase pairs that are generated directly from the model, and also for phrases that, while not directly included in the model, are composed of two high probability child phrases.

It should be noted that while for FLAT and HIER P_t can be used directly, as HLEN learns separate models for each length, we must combine these probabilities into a single value. We do this by setting

$$P_t(\langle e, f\rangle) = P_{t,l}(\langle e, f\rangle) c(l) / \sum_{\tilde{l}=1}^{L} c(\tilde{l})$$

for every phrase pair, where $l = |e| + |f|$ and $c(l)$ is the number of phrases of length l in the sample.

We call this model-based extraction method MOD.

5.3 Sample Combination

As has been noted in previous works, (Koehn et al., 2003; DeNero et al., 2006) exhaustive phrase extraction tends to out-perform approaches that use syntax or generative models to limit phrase boundaries. DeNero et al. (2006) state that this is because generative models choose only a single phrase segmentation, and thus throw away many good phrase pairs that are in conflict with this segmentation.

Luckily, in the Bayesian framework it is simple to overcome this problem by combining phrase tables

from multiple samples. This is equivalent to approximating the integral over various parameter configurations in Equation (1). In MOD, we do this by taking the average of the joint probability and span probability features, and re-calculating the conditional probabilities from the averaged joint probabilities.

6 Related Work

In addition to the previously mentioned phrase alignment techniques, there has also been a significant body of work on phrase extraction (Moore and Quirk (2007), Johnson et al. (2007a), *inter alia*). DeNero and Klein (2010) presented the first work on joint phrase alignment and extraction at multiple levels. While they take a supervised approach based on discriminative methods, we present a fully unsupervised generative model.

A generative probabilistic model where longer units are built through the binary combination of shorter units was proposed by de Marcken (1996) for monolingual word segmentation using the minimum description length (MDL) framework. Our work differs in that it uses Bayesian techniques instead of MDL, and works on two languages, not one.

Adaptor grammars, models in which nonterminals memorize subtrees that lie below them, have been used for word segmentation or other monolingual tasks (Johnson et al., 2007b). The proposed method could be thought of as synchronous adaptor grammars over two languages. However, adaptor grammars have generally been used to specify only two or a few levels as in the FLAT model in this paper, as opposed to recursive models such as HIER or many-leveled models such as HLEN. One exception is the variational inference method for adaptor grammars presented by Cohen et al. (2010) that is applicable to recursive grammars such as HIER. We plan to examine variational inference for the proposed models in future work.

7 Experimental Evaluation

We evaluate the proposed method on translation tasks from four languages, French, German, Spanish, and Japanese, into English.

	de-en	es-en	fr-en	ja-en
TM (en)	1.80M	1.62M	1.35M	2.38M
TM (other)	1.85M	1.82M	1.56M	2.78M
LM (en)	52.7M	52.7M	52.7M	44.7M
Tune (en)	49.8k	49.8k	49.8k	68.9k
Tune (other)	47.2k	52.6k	55.4k	80.4k
Test (en)	65.6k	65.6k	65.6k	40.4k
Test (other)	62.7k	68.1k	72.6k	48.7k

Table 1: The number of words in each corpus for TM and LM training, tuning, and testing.

7.1 Experimental Setup

The data for French, German, and Spanish are from the 2010 Workshop on Statistical Machine Translation (Callison-Burch et al., 2010). We use the news commentary corpus for training the TM, and the news commentary and Europarl corpora for training the LM. For Japanese, we use data from the NTCIR patent translation task (Fujii et al., 2008). We use the first 100k sentences of the parallel corpus for the TM, and the whole parallel corpus for the LM. Details of both corpora can be found in Table 1. Corpora are tokenized, lower-cased, and sentences of over 40 words on either side are removed for TM training. For both tasks, we perform weight tuning and testing on specified development and test sets.

We compare the accuracy of our proposed method of joint phrase alignment and extraction using the FLAT, HIER and HLEN models, with a baseline of using word alignments from GIZA++ and heuristic phrase extraction. Decoding is performed using Moses (Koehn and others, 2007) using the phrase tables learned by each method under consideration, as well as standard bidirectional lexical reordering probabilities (Koehn et al., 2005). Maximum phrase length is limited to 7 in all models, and for the LM we use an interpolated Kneser-Ney 5-gram model.

For GIZA++, we use the standard training regimen up to Model 4, and combine alignments with `grow-diag-final-and`. For the proposed models, we train for 100 iterations, and use the final sample acquired at the end of the training process for our experiments using a single sample[6]. In addition,

[6]For most models, while likelihood continued to increase gradually for all 100 iterations, BLEU score gains plateaued after 5-10 iterations, likely due to the strong prior information

			de-en		es-en		fr-en		ja-en	
Align	Extract	# Samp.	BLEU	Size	BLEU	Size	BLEU	Size	BLEU	Size
GIZA++	HEUR-W	1	**16.62**	4.91M	**22.00**	4.30M	21.35	4.01M	**23.20**	4.22M
FLAT	MOD	1	13.48	136k	19.15	125k	17.97	117k	16.10	89.7k
HIER	MOD	1	**16.58**	1.02M	**21.79**	859k	**21.50**	751k	**23.23**	723k
HLEN	MOD	1	**16.49**	1.17M	21.57	930k	21.31	860k	**23.19**	820k
HIER	MOD	10	**16.53**	3.44M	**21.84**	2.56M	**21.57**	2.63M	**23.12**	2.21M
HLEN	MOD	10	**16.51**	3.74M	**21.69**	3.00M	**21.53**	3.09M	**23.20**	2.70M

Table 2: BLEU score and phrase table size by alignment method, extraction method, and samples combined. Bold numbers are not significantly different from the best result according to the sign test ($p < 0.05$) (Collins et al., 2005).

we also try averaging the phrase tables from the last ten samples as described in Section 5.3.

7.2 Experimental Results

The results for these experiments can be found in Table 2. From these results we can see that when using a single sample, the combination of using HIER and model probabilities achieves results approximately equal to GIZA++ and heuristic phrase extraction. This is the first reported result in which an unsupervised phrase alignment model has built a phrase table directly from model probabilities and achieved results that compare to heuristic phrase extraction. It can also be seen that the phrase table created by the proposed method is approximately 5 times smaller than that obtained by the traditional pipeline.

In addition, HIER significantly outperforms FLAT when using the model probabilities. This confirms that phrase tables containing only minimal phrases are not able to achieve results that compete with phrase tables that use multiple granularities.

Somewhat surprisingly, HLEN consistently slightly underperforms HIER. This indicates potential gains to be provided by length-based parameter tuning were outweighed by losses due to the increased complexity of the model. In particular, we believe the necessity to combine probabilities from multiple $P_{t,l}$ models into a single phrase table may have resulted in a distortion of the phrase probabilities. In addition, the assumption that phrase lengths are generated from a uniform distribution is likely too strong, and further gains could likely be achieved by more accurate modeling of phrase lengths. We leave further adjustments to the HLEN model to future work.

provided by P_{base}. As iterations took 1.3 hours on a single processor, good translation results can be achieved in approximately 13 hours, which could further reduced using distributed sampling (Newman et al., 2009; Blunsom et al., 2009).

	FLAT		HIER	
MOD	17.97	117k	21.50	751k
HEUR-W	21.52	5.65M	21.68	5.39M
HEUR-B	21.45	4.93M	21.41	2.61M
HEUR-P	21.56	4.88M	21.47	1.62M

Table 3: Translation results and phrase table size for various phrase extraction techniques (French-English).

It can also be seen that combining phrase tables from multiple samples improved the BLEU score for HLEN, but not for HIER. This suggests that for HIER, most of the useful phrase pairs discovered by the model are included in every iteration, and the increased recall obtained by combining multiple samples does not consistently outweigh the increased confusion caused by the larger phrase table.

We also evaluated the effectiveness of model-based phrase extraction compared to heuristic phrase extraction. Using the alignments from HIER, we created phrase tables using model probabilities (MOD), and heuristic extraction on words (HEUR-W), blocks (HEUR-B), and minimal phrases (HEUR-P) as described in Section 5. The results of these experiments are shown in Table 3. It can be seen that model-based phrase extraction using HIER outperforms or insignificantly underperforms heuristic phrase extraction over all experimental settings, while keeping the phrase table to a fraction of the size of most heuristic extraction methods.

Finally, we varied the size of the parallel corpus for the Japanese-English task from 50k to 400k sen-

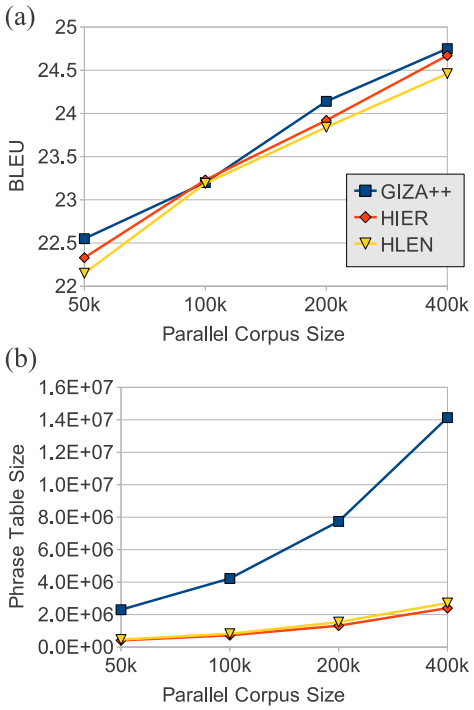


Figure 4: The effect of corpus size on the accuracy (a) and phrase table size (b) for each method (Japanese-English).

tences and measured the effect of corpus size on translation accuracy. From the results in Figure 4 (a), it can be seen that at all corpus sizes, the results from all three methods are comparable, with insignificant differences between GIZA++ and HIER at all levels, and HLEN lagging slightly behind HIER. Figure 4 (b) shows the size of the phrase table induced by each method over the various corpus sizes. It can be seen that the tables created by GIZA++ are significantly larger at all corpus sizes, with the difference being particularly pronounced at larger corpus sizes.

8 Conclusion

In this paper, we presented a novel approach to joint phrase alignment and extraction through a hierarchical model using non-parametric Bayesian methods and inversion transduction grammars. Machine translation systems using phrase tables learned directly by the proposed model were able to achieve accuracy competitive with the traditional pipeline of word alignment and heuristic phrase extraction, the first such result for an unsupervised model.

For future work, we plan to refine HLEN to use a more appropriate model of phrase length than the uniform distribution, particularly by attempting to bias against phrase pairs where one of the two phrases is much longer than the other. In addition, we will test probabilities learned using the proposed model with an ITG-based decoder. We will also examine the applicability of the proposed model in the context of hierarchical phrases (Chiang, 2007), or in alignment using syntactic structure (Galley et al., 2006). It is also worth examining the plausibility of variational inference as proposed by Cohen et al. (2010) in the alignment context.

Acknowledgments

This work was performed while the first author was supported by the JSPS Research Fellowship for Young Scientists.

References

Phil Blunsom and Trevor Cohn. 2010. Inducing synchronous grammars with slice sampling. In *Proceedings of the Human Language Technology: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A Gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics*, pages 782–790.

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

Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar F. Zaidan. 2010. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the Joint 5th Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53.

Colin Cherry and Dekang Lin. 2007. Inversion transduction grammar for joint phrasal translation modeling. In *Proceedings of the NAACL Workshop on Syntax and Structure in Machine Translation*.

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

Shay B. Cohen, David M. Blei, and Noah A. Smith. 2010. Variational inference for adaptor grammars. In *Proceedings of the Human Language Technology: The*

11th Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 564–572.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 531–540.

Carl de Marcken. 1996. *Unsupervised Language Acquisition*. Ph.D. thesis, Massachusetts Institute of Technology.

John DeNero and Dan Klein. 2008. The complexity of phrase alignment problems. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 25–28.

John DeNero and Dan Klein. 2010. Discriminative modeling of extraction sets for machine translation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1453–1463.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. 2006. Why generative phrase models underperform surface heuristics. In *Proceedings of the 1st Workshop on Statistical Machine Translation*, pages 31–38.

John DeNero, Alex Bouchard-Côté, and Dan Klein. 2008. Sampling alignment structure under a Bayesian translation model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 314–323.

Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, and Takehito Utsuro. 2008. Overview of the patent translation task at the NTCIR-7 workshop. In *Proceedings of the 7th NTCIR Workshop Meeting on Evaluation of Information Access Technologies*, pages 389–400.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 961–968.

J. Howard Johnson, Joel Martin, George Foster, and Roland Kuhn. 2007a. Improving translation quality by discarding most of the phrasetable. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007b. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. *Advances in Neural Information Processing Systems*, 19:641.

Philipp Koehn et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Phillip Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Human Language Technology Conference (HLT-NAACL)*, pages 48–54.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proceedings of the International Workshop on Spoken Language Translation*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the Human Language Technology Conference - North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL)*, pages 104–111.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. pages 133–139.

Robert C. Moore and Chris Quirk. 2007. An iteratively-trained segmentation-free phrase translation model for statistical machine translation. In *Proceedings of the 2nd Workshop on Statistical Machine Translation*, pages 112–119.

David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2009. Distributed algorithms for topic models. *Journal of Machine Learning Research*, 10:1801–1828.

Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *Proceedings of the 4th Conference on Empirical Methods in Natural Language Processing*, pages 20–28.

Jim Pitman and Marc Yor. 1997. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, 25(2):855–900.

Markus Saers, Joakim Nivre, and Dekai Wu. 2009. Learning stochastic bracketing inversion transduction grammars with a cubic time biparsing algorithm. In *Proceedings of the The 11th International Workshop on Parsing Technologies*.

Yee Whye Teh. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Hao Zhang, Chris Quirk, Robert C. Moore, and Daniel Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 97–105.

Learning Hierarchical Translation Structure with Linguistic Annotations

Markos Mylonakis
ILLC
University of Amsterdam
m.mylonakis@uva.nl

Khalil Sima'an
ILLC
University of Amsterdam
k.simaan@uva.nl

Abstract

While it is generally accepted that many translation phenomena are correlated with linguistic structures, employing linguistic syntax for translation has proven a highly non-trivial task. The key assumption behind many approaches is that translation is guided by the source and/or target language parse, employing rules extracted from the parse tree or performing tree transformations. These approaches enforce strict constraints and might overlook important translation phenomena that cross linguistic constituents. We propose a novel flexible modelling approach to introduce linguistic information of varying granularity from the source side. Our method induces joint probability synchronous grammars and estimates their parameters, by selecting and weighing together linguistically motivated rules according to an objective function directly targeting generalisation over future data. We obtain statistically significant improvements across 4 different language pairs with English as source, mounting up to +1.92 BLEU for Chinese as target.

1 Introduction

Recent advances in Statistical Machine Translation (SMT) are widely centred around two concepts: (a) hierarchical translation processes, frequently employing Synchronous Context Free Grammars (SCFGs) and (b) transduction or synchronous rewrite processes over a linguistic syntactic tree. SCFGs in the form of the Inversion-Transduction Grammar (ITG) were first introduced by (Wu, 1997) as a formalism to recursively describe the translation process. The Hiero system (Chiang, 2005) utilised an ITG-flavour which focused on hierarchical phrase-pairs to capture context-driven translation and reordering patterns with 'gaps', offering competitive performance particularly for language pairs with extensive reordering. As Hiero uses a single non-terminal and concentrates on overcoming translation lexicon sparsity, it barely explores the recursive nature of translation past the lexical level. Nevertheless, the successful employment of SCFGs for phrase-based SMT brought translation models assuming latent syntactic structure to the spotlight.

Simultaneously, mounting efforts have been directed towards SMT models employing linguistic syntax on the source side (Yamada and Knight, 2001; Quirk et al., 2005; Liu et al., 2006), target side (Galley et al., 2004; Galley et al., 2006) or both (Zhang et al., 2008; Liu et al., 2009; Chiang, 2010). Hierarchical translation was combined with target side linguistic annotation in (Zollmann and Venugopal, 2006). Interestingly, early on (Koehn et al., 2003) exemplified the difficulties of integrating linguistic information in translation systems. Syntax-based MT often suffers from inadequate constraints in the translation rules extracted, or from striving to combine these rules together towards a full derivation. Recent research tries to address these issues, by re-structuring training data parse trees to better suit syntax-based SMT training (Wang et al., 2010), or by moving from linguistically motivated synchronous grammars to systems where linguistic plausibility of the translation is assessed through additional features in a phrase-based system (Venugopal et al., 2009; Chiang et al., 2009), obscuring the impact of higher level syntactic processes.

While it is assumed that linguistic structure does correlate with some translation phenomena, in this

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 642–652,
Portland, Oregon, June 19-24, 2011.

work we do not employ it as the backbone of translation. In place of linguistically *constrained* translation imposing syntactic parse structure, we opt for linguistically *motivated* translation. We learn latent hierarchical structure, taking advantage of linguistic annotations but shaped and trained for translation.

We start by labelling each phrase-pair span in the word-aligned training data with multiple linguistically motivated categories, offering multi-grained abstractions from its lexical content. These phrase-pair label charts are the input of our learning algorithm, which extracts the linguistically motivated rules and estimates the probabilities for a stochastic SCFG, without arbitrary constraints such as phrase or span sizes. Estimating such grammars under a Maximum Likelihood criterion is known to be plagued by strong overfitting leading to degenerate estimates (DeNero et al., 2006). In contrast, our learning objective not only avoids overfitting the training data but, most importantly, learns joint stochastic synchronous grammars which directly aim at generalisation towards yet unseen instances.

By advancing from structures which mimic linguistic syntax, to learning linguistically aware latent recursive structures targeting translation, we achieve significant improvements in translation quality for 4 different language pairs in comparison with a strong hierarchical translation baseline.

Our key contributions are presented in the following sections. Section 2 discusses the weak independence assumptions of SCFGs and introduces a joint translation model which addresses these issues and separates hierarchical translation structure from phrase-pair emission. In section 3 we consider a chart over phrase-pair spans filled with source-language linguistically motivated labels. We show how we can employ this crucial input to extract and train a hierarchical translation structure model with millions of rules. Section 4 demonstrates decoding with the model by constraining derivations to linguistic hints of the source sentence and presents our empirical results. We close with a discussion of related work and our conclusions.

2 Joint Translation Model

Our model is based on a probabilistic Synchronous CFG (Wu, 1997; Chiang, 2005). SCFGs define a language over string pairs, which are generated beginning from a start symbol S and recursively expanding pairs of linked non-terminals across the two strings using the grammar's rule set. By crossing the links between the non-terminals of the two sides reordering phenomena are captured. We employ binary SCFGs, i.e. grammars with a maximum of two non-terminals on the right-hand side. Also, for this work we only used grammars with either purely lexical or purely abstract rules involving one or two non-terminal pairs. An example can be seen in Figure 1, using an ITG-style notation and assuming the same non-terminal labels for both sides.

$$\text{SBAR} \rightarrow [\text{WHNP}\ \ \text{SBAR}\backslash\text{WHNP}] \quad \text{(a)}$$
$$\text{SBAR}\backslash\text{WHNP} \rightarrow \langle \text{VP/NP}^{\mathbf{L}}\ \ \text{NP}^{\mathbf{R}} \rangle \quad \text{(b)}$$
$$\text{NP}^{\mathbf{R}} \rightarrow [\text{NP}\ \ \text{PP}] \quad \text{(c)}$$
$$\text{WHNP} \rightarrow \text{WHNP}_{\mathbf{P}} \quad \text{(d)}$$
$$\text{WHNP}_{\mathbf{P}} \rightarrow \text{which} \ / \ \text{der} \quad \text{(e)}$$
$$\text{VP/NP}^{\mathbf{L}} \rightarrow \text{VP/NP}^{\mathbf{L}}_{\mathbf{P}} \quad \text{(f)}$$
$$\text{VP/NP}^{\mathbf{L}}_{\mathbf{P}} \rightarrow \text{is} \ / \ \text{ist} \quad \text{(g)}$$
$$\text{NP}^{\mathbf{R}} \rightarrow \text{NP}^{\mathbf{R}}_{\mathbf{P}} \quad \text{(h)}$$
$$\text{NP}^{\mathbf{R}}_{\mathbf{P}} \rightarrow \text{the solution} \ / \ \text{die Lösung} \quad \text{(i)}$$
$$\text{NP} \rightarrow \text{NP}_{\mathbf{P}} \quad \text{(j)}$$
$$\text{NP}_{\mathbf{P}} \rightarrow \text{the solution} \ / \ \text{die Lösung} \quad \text{(k)}$$
$$\text{PP} \rightarrow \text{PP}_{\mathbf{P}} \quad \text{(l)}$$
$$\text{PP}_{\mathbf{P}} \rightarrow \text{to the problem} \ / \ \text{für das Problem} \quad \text{(m)}$$

Figure 1: English-German SCFG rules for the relative clause(s) 'which is the solution (to the problem) / der die Lösung (für das Problem) ist', [ ] signify monotone translation, ⟨ ⟩ a swap reordering.

We utilise *probabilistic* SCFGs, where each rule is assigned a conditional probability of expanding the left-hand side symbol with the rule's right-hand side. Phrase-pairs are emitted jointly and the overall probabilistic SCFG is a *joint* model over parallel strings.

2.1 SCFG Reordering Weaknesses

An interesting feature of all probabilistic SCFGs (i.e. not only binary ones), which has received surprisingly little attention, is that the reordering pat-

tern between the non-terminal pairs (or in the case of ITGs the choice between monotone and swap expansion) are not conditioned on any other part of a derivation. The result is that, the reordering pattern with the highest probability will *always* be preferred (e.g. in the Viterbi derivation) over the rest, irrespective of lexical or abstract context. As an example, a probabilistic SCFG will always assign a higher probability to derivations swapping or monotonically translating nouns and adjectives between English and French, only depending on which of the two rules $NP \rightarrow [NN \;\; JJ]$, $NP \rightarrow \langle NN \;\; JJ \rangle$ has a higher probability. The rest of the (sometimes thousands of) rule-specific features usually added to SCFG translation models do not directly help either, leaving reordering decisions disconnected from the rest of the derivation.

While in a decoder this is somehow mitigated by the use of a language model, we believe that the weakness of straightforward applications of SCFGs to model reordering structure at the sentence level misses a chance to learn this crucial part of the translation process during grammar induction. As (Mylonakis and Sima'an, 2010) note, 'plain' SCFGs seem to perform worse than the grammars described next, mainly due to wrong long-range reordering decisions for which the language model can hardly help.

2.2 Hierarchical Reordering SCFG

We address the weaknesses mentioned above by relying on an SCFG grammar design that is similar to the 'Lexicalised Reordering' grammar of (Mylonakis and Sima'an, 2010). As in the rules of Figure 1, we separate non-terminals according to the reordering patterns in which they participate. Non-terminals such as $\mathrm{B}^{\mathbf{L}}$, $\mathrm{C}^{\mathbf{R}}$ take part only in swapping right-hand sides $\langle \mathrm{B}^{\mathbf{L}} \;\; \mathrm{C}^{\mathbf{R}} \rangle$ (with $\mathrm{B}^{\mathbf{L}}$ swapping from the source side's left to the target side's right, $\mathrm{C}^{\mathbf{R}}$ swapping in the opposite direction), while non-terminals such as B, C take part solely in monotone right-hand side expansions [B C]. These non-terminal categories can appear also on the left-hand side of a rule, as in rule (c) of Figure 1.

In contrast with (Mylonakis and Sima'an, 2010), monotone and swapping non-terminals do not emit phrase-pairs themselves. Rather, each non-terminal NT is expanded to a dedicated phrase-pair emitting non-terminal $\mathrm{NT}_{\mathbf{P}}$, which generates all phrase-pairs for it and nothing more. In this way, the preference of non-terminals to either expand towards a (long) phrase-pair or be further analysed recursively is explicitly modelled. Furthermore, this set of *pre-terminals* allows us to separate the higher order translation structure from the process that emits phrase-pairs, a feature we employ next.

$$
\begin{array}{rr}
\mathrm{A} \rightarrow [\mathrm{B} \;\; \mathrm{C}] & \mathrm{A} \rightarrow \langle \mathrm{B}^{\mathbf{L}} \;\; \mathrm{C}^{\mathbf{R}} \rangle \\
\mathrm{A}^{\mathbf{L}} \rightarrow [\mathrm{B} \;\; \mathrm{C}] & \mathrm{A}^{\mathbf{L}} \rightarrow \langle \mathrm{B}^{\mathbf{L}} \;\; \mathrm{C}^{\mathbf{R}} \rangle \\
\mathrm{A}^{\mathbf{R}} \rightarrow [\mathrm{B} \;\; \mathrm{C}] & \mathrm{A}^{\mathbf{R}} \rightarrow \langle \mathrm{B}^{\mathbf{L}} \;\; \mathrm{C}^{\mathbf{R}} \rangle \\
\mathrm{A} \rightarrow \mathrm{A}_{\mathbf{P}} & \mathrm{A}_{\mathbf{P}} \rightarrow \alpha \;/\; \beta \\
\mathrm{A}^{\mathbf{L}} \rightarrow \mathrm{A}^{\mathbf{L}}_{\mathbf{P}} & \mathrm{A}^{\mathbf{L}}_{\mathbf{P}} \rightarrow \alpha \;/\; \beta \\
\mathrm{A}^{\mathbf{R}} \rightarrow \mathrm{A}^{\mathbf{R}}_{\mathbf{P}} & \mathrm{A}^{\mathbf{R}}_{\mathbf{P}} \rightarrow \alpha \;/\; \beta
\end{array}
$$

Figure 2: Recursive Reordering Grammar rule categories; A, B, C non-terminals; α, β source and target strings respectively.

In (Mylonakis and Sima'an, 2010) this grammar design mainly contributed to model lexical reordering preferences. While we retain this function, for the rich linguistically-motivated grammars used in this work this design effectively propagates reordering preferences above and below the current rule application (e.g. Figure 1, rules (a)-(c)), allowing to learn and apply complex reordering patterns.

The different types of grammar rules are summarised in abstract form in Figure 2. We will subsequently refer to this grammar structure as Hierarchical Reordering SCFG (HR-SCFG).

2.3 Generative Model

We arrive at a probabilistic SCFG model which jointly generates source $\mathbf{e}$ and target $\mathbf{f}$ strings, by augmenting each grammar rule with a probability, summing up to one for every left-hand side. The probability of a derivation D of tuple $\langle \mathbf{e}, \mathbf{f} \rangle$ beginning from start symbol S is equal to the product of the probabilities of the rules used to recursively generate it.

We separate the structural part of the derivation D, down to the pre-terminals $\mathrm{NT}_{\mathbf{P}}$, from the phrase-emission part. The grammar rules pertaining to the

<table>
<tr><td colspan="4">X, SBAR, WHNP+VP, WHNP+VBZ+NP</td></tr>
<tr><td></td><td colspan="3">X, VBZ+NP, VP, SBAR\WHNP</td></tr>
<tr><td colspan="3">X, SBAR/NN, WHNP+VBZ+DT</td><td></td></tr>
<tr><td></td><td colspan="2">X, VBZ+DT, VP/NN</td><td></td></tr>
<tr><td colspan="2">X, WHNP+VBZ, SBAR/NP</td><td colspan="2">X, NP, VP\VBZ</td></tr>
<tr><td>X, WHNP, SBAR/VP</td><td>X, VBZ, VP/NP</td><td>X, DT, NP/NN</td><td>X, NN, NP\DT</td></tr>
<tr><td>which</td><td>is</td><td>the</td><td>problem</td></tr>
</table>

Figure 3: The label chart for the source fragment 'which is the problem'. Only a sample of the entries is listed.

structural part and their associated probabilities define a model $p(\sigma)$ over the latent variable σ determining the recursive, reordering and phrase-pair segmenting structure of translation, as in Figure 4. Given σ, the phrase-pair emission part merely generates the phrase-pairs utilising distributions from every $\text{NT}_\mathbf{P}$ to the phrase-pairs that it covers, thereby defining a model over all sentence-pairs generated given each translation structure. The probabilities of a derivation and of a sentence-pair are then as follows:

$$p(D) = p(\sigma)p(\mathbf{e}, \mathbf{f}|\sigma) \quad (1)$$

$$p(\mathbf{e}, \mathbf{f}) = \sum_{D:D\overset{*}{\Rightarrow}\langle \mathbf{e},\mathbf{f}\rangle} p(D) \quad (2)$$

By splitting the joint model in a hierarchical structure model and a lexical emission one we facilitate estimating the two models separately. The following section discusses this.

3 Learning Translation Structure

3.1 Phrase-Pair Label Chart

The input to our learning algorithm is a word-aligned parallel corpus. We consider as phrase-pair spans those that obey the word-alignment constraints of (Koehn et al., 2003). For every training sentence-pair, we also input a chart containing one or more labels for every synchronous span, such as that of Figure 3. Each label describes different properties of the phrase pair (syntactic, semantic etc.), possibly in relation to its context, or supplying varying levels of abstraction (phrase-pair, determiner with noun, noun-phrase, sentence etc.). We aim to induce a recursive translation structure explaining the joint generation of the source and target sentence taking advantage of these phrase-pair span labels.

For this work we employ the linguistically motivated labels of (Zollmann and Venugopal, 2006), albeit for the source language. Given a parse of the source sentence, each span is assigned the following kind of labels:

Phrase-Pair All phrase-pairs are assigned the X label

Constituent Source phrase is a constituent A

Concatenation of Constituents Source phrase labelled A+B as a concatenation of constituents A and B, similarly for 3 constituents.

Partial Constituents Categorial grammar (Bar-Hillel, 1953) inspired labels A/B, A\B, indicating a partial constituent A missing constituent B right or left respectively.

An important point is that we assign all applicable labels to every span. In this way, each label set captures the features of the source side's parse-tree without being bounded by the actual parse structure, as well as provides a coarse to fine-grained view of the source phrase.

3.2 Grammar Extraction

From every word-aligned sentence-pair and its label chart, we extract SCFG rules as those of Figure 2. Binary rules are extracted from adjoining synchronous spans up to the whole sentence-pair level, with the non-terminals of both left and right-hand side derived from the label names plus their reordering function (monotone, left/right swapping) in the span examined. A single unary rule per non-terminal NT generates the phrase-pair emitting $\text{NT}_\mathbf{P}$. Unary rules $\text{NT}_\mathbf{P} \rightarrow \alpha \ / \ \beta$ generating the phrase-pair are created for all the labels covering it.

While we label the phrase-pairs similarly to (Zollmann and Venugopal, 2006), the extracted grammar is rather different. We do not employ rules that are grounded to lexical context ('gap' rules), relying instead on the reordering-aware non-terminal set and related unary and binary rules. The result is a grammar which can both capture a rich array of translation phenomena based on linguistic and lexical grounds and explicitly model the balance between

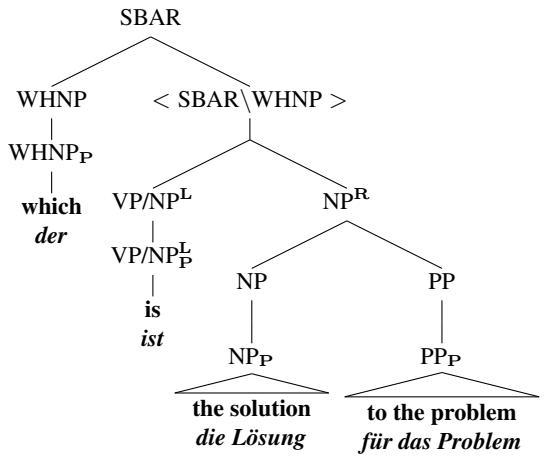


Figure 4: A derivation of a sentence fragment with the grammar of Figure 1.

memorising long phrase-pairs and generalising over yet unseen ones, as shown in the next example.

The derivation in Figure 4 illustrates some of the formalism's features. A preference to reorder based on lexical *content* is applied for is / ist. Noun phrase NP$^{\mathbf{R}}$ is recursively constructed with a preference to constitute the right branch of an order swapping non-terminal expansion. This is matched with VP/NP$^{\mathbf{L}}$ which reorders in the opposite direction. The labels VP/NP and SBAR\WHNP allow linguistic syntax *context* to influence the lexical and reordering translation choices. Crucially, *all* these lexical, attachment and reordering preferences (as encoded in the model's rules and probabilities) must be matched together to arrive at the analysis in Figure 4.

3.3 Parameter Estimation

We estimate the parameters for the phrase-emission model $p(\mathbf{e}, \mathbf{f}|\sigma)$ using Relative Frequency Estimation (RFE) on the *label charts* induced for the training sentence-pairs, after the labels have been augmented by the reordering indications. In the RFE estimate, every rule $\mathrm{NT}_{\mathbf{P}} \rightarrow \alpha \ / \ \beta$ receives a probability in proportion with the times that $\alpha \ / \ \beta$ was covered by the NT label.

On the other hand, estimating the parameters under Maximum-Likelihood Estimation (MLE) for the latent translation structure model $p(\sigma)$ is bound to overfit towards memorising whole sentence-pairs as discussed in (Mylonakis and Sima'an, 2010), with the resulting grammar estimate not being able to generalise past the training data. However, apart from overfitting towards long phrase-pairs, a grammar with millions of structural rules is also liable to overfit towards degenerate latent structures which, while fitting the training data well, have limited applicability to unseen sentences.

We avoid both pitfalls by estimating the grammar probabilities with the Cross-Validating Expectation-Maximization algorithm (CV-EM) (Mylonakis and Sima'an, 2008; Mylonakis and Sima'an, 2010). CV-EM is a cross-validating instance of the well known EM algorithm (Dempster et al., 1977). It works iteratively on a partition of the training data, climbing the likelihood of the training data while cross-validating the latent variable values, considering for every training data point only those which can be produced by models built from the rest of the data excluding the current part. As a result, the estimation process simulates maximising future data likelihood, using the training data to directly aim towards strong generalisation of the estimate.

For our probabilistic SCFG-based translation structure variable σ, implementing CV-EM boils down to a synchronous version of the Inside-Outside algorithm, modified to enforce the CV criterion. In this way we arrive at cross-validated ML estimate of the σ parameters while keeping the phrase-emission parameters of $p(\mathbf{e}, \mathbf{f}|\sigma)$ fixed. The CV-criterion, apart from avoiding overfitting, results in discarding the structural rules which are only found in a single part of the training corpus, leading to a more compact grammar while still retaining millions of structural rules that are more hopeful to generalise.

Unravelling the joint generative process, by modelling latent hierarchical structure separately from phrase-pair emission, allows us to concentrate our inference efforts towards the hidden, higher-level translation mechanism.

4 Experiments

4.1 Decoding Model

The induced joint translation model can be used to recover $\arg\max_{\mathbf{e}} p(\mathbf{e}|\mathbf{f})$, as it is equal to $\arg\max_{\mathbf{e}} p(\mathbf{e}, \mathbf{f})$. We employ the induced probabilistic HR-SCFG $\mathbf{G}$ as the backbone of a log-linear, feature based translation model, with the derivation probability $p(D)$ under the grammar estimate being

one of the features. This is augmented with a small number n of additional smoothing features ϕ_i for derivation rules r: (a) conditional phrase translation probabilities, (b) lexical phrase translation probabilities, (c) word generation penalty, and (d) a count of swapping reordering operations. Features (a), (b) and (c) are applicable to phrase-pair emission rules and features for both translation directions are used, while (d) is only triggered by structural rules.

These extra features assess translation quality past the synchronous grammar derivation and learning general reordering or word emission preferences for the language pair. As an example, while our probabilistic HR-SCFG maintains a separate joint phrase-pair emission distribution per non-terminal, the smoothing features (a) above assess the conditional translation of surface phrases irrespective of any notion of recursive translation structure.

The final feature is the language model score for the target sentence, mounting up to the following model used at decoding time, with the feature weights λ trained by Minimum Error Rate Training (MERT) (Och, 2003) on a development corpus.

$$p(D \stackrel{*}{\Rightarrow} \langle \mathbf{e}, \mathbf{f} \rangle) \propto p(\mathbf{e})^{\lambda_{lm}} p_{\mathbf{G}}(D)^{\lambda_{\mathbf{G}}} \prod_{i=1}^{n} \prod_{r \in D} \phi_i(r)^{\lambda_i}$$

4.2 Decoding Modifications

We use a customised version of the Joshua SCFG decoder (Li et al., 2009) to translate, with the following modifications:

Source Labels Constraints As for this work the phrase-pair labels used to extract the grammar are based on the linguistic analysis of the source side, we can construct the label chart for every input sentence from its parse. We subsequently use it to consider only derivations with synchronous spans which are covered by non-terminals matching one of the labels for those spans. This applies both for the non-terminals covering phrase-pairs as well as the higher level parts of the derivation.

In this manner we not only constrain the translation hypotheses resulting in faster decoding time, but, more importantly, we may ground the hypotheses more closely to the available linguistic information of the source sentence. This is of particular interest as we move up the derivation tree, where an initial wrong choice below could propagate towards hypotheses wildly diverging from the input sentence's linguistic annotation.

Per Non-Terminal Pruning The decoder uses a combination of beam and cube-pruning (Huang and Chiang, 2007). As our grammar uses non-terminals in the hundreds of thousands, it is important not to prune away prematurely non-terminals covering smaller spans and to leave more options to be considered as we move up the derivation tree.

For this, for every cell in the decoder's chart, we keep a separate bin per non-terminal and prune together hypotheses leading to the same non-terminal covering a cell. This allows full derivations to be found for all input sentences, as well as avoids aggressive pruning at an early stage. Given the source label constraint discussed above, this does not increase running times or memory demands considerably as we allow only up to a few tens of non-terminals per span.

Expected Counts Rule Pruning To compact the hierarchical structure part of the grammar prior to decoding, we prune rules that fail to accumulate 10^{-8} expected counts during the last CV-EM iteration. For English to German, this brings the structural rules from 15M down to 1.2M. Note that we do not prune the phrase-pair emitting rules. Overall, we consider this a much more informed pruning criterion than those based on probability values (that are not comparable across left-hand sides) or right-hand side counts (frequent symbols need many more expansions than a highly specialised one).

4.3 Experimental Setting & Baseline

We evaluate our method on four different language pairs with English as the source language and French, German, Dutch and Chinese as target. The data for the first three language pairs are derived from parliament proceedings sourced from the Europarl corpus (Koehn, 2005), with WMT-07 development and test data for French and German. The data for the English to Chinese task is composed of parliament proceedings and news articles. For all language pairs we employ 200K and 400K sentence pairs for training, 2K for development and 2K for testing (single reference per source sentence). Both the baseline and our method decode

Training set size	English to	French		German		Dutch		Chinese	
		BLEU	NIST	BLEU	NIST	BLEU	NIST	BLEU	NIST
200K	josh-base	29.20	7.2123	18.65	5.8047	21.97	6.2469	22.34	6.5540
	lts	**29.43**	**7.2611****	**19.10****	**5.8714****	**22.31***	**6.2903***	**23.67****	**6.6595****
400K	josh-base	29.58	7.3033	18.86	5.8818	22.25	6.2949	23.24	6.7402
	lts	**29.83**	**7.4000****	**19.49****	**5.9374****	**22.92****	**6.3727****	**25.16****	**6.9005****

Table 1: Experimental results for training sets of 200K and 400K sentence pairs. Statistically significant score improvements from the baseline at the 95% confidence level are labelled with a single star, at the 99% level with two.

with a 3-gram language model smoothed with modified Knesser-Ney discounting (Chen and Goodman, 1998), trained on around 1M sentences per target language. The parses of the source sentences employed by our system during training and decoding are created with the Charniak parser (Charniak, 2000).

We compare against a state-of-the-art hierarchical translation (Chiang, 2005) baseline, based on the Joshua translation system under the default training and decoding settings (`josh-base`). Apart of evaluating against a state-of-the-art system, especially on the English-Chinese language pair, the comparison has an added interesting aspect. The heuristically trained baseline takes advantage of 'gap rules' to reorder based on lexical context cues, but makes very limited use of the hierarchical structure above the lexical surface. In contrast, our method induces a grammar with no such rules, relying on lexical content and the strength of a higher level translation structure instead.

4.4 Training & Decoding Details

To train our Latent Translation Structure (LTS) system, we used the following settings. CV-EM cross-validated on a 10-part partition of the training data and performed 10 iterations. The structural rule probabilities were initialised to uniform per left-hand side.

The decoder does not employ any 'glue grammar' as is usual with hierarchical translation systems to limit reordering up to a certain cut-off length. Instead, we rely on our LTS grammar to reorder and construct the translation output up to the full sentence length.

In summary, our system's experimental pipeline is as follows. All input sentences are parsed and label charts are created from these parses. The Hierarchical Reordering SCFG is extracted and its parameters are estimated employing CV-EM. The structural rules of the estimate are pruned according to their expected counts and smoothing features are added to all rules. We train the feature weights under MERT and decode with the resulting log-linear model.

The overall training and decoding setup is appealing also regarding computational demands. On an 8-core 2.3GHz system, training on 200K sentence-pairs demands 4.5 hours while decoding runs on 25 sentences per minute.

4.5 Results

Table 1 presents the results for the baseline and our method for the 4 language pairs, for training sets of both 200K and 400K sentence pairs. Our system (`lts`) outperforms the baseline for all 4 language pairs for both BLEU and NIST scores, by a margin which scales up to +1.92 BLEU points for English to Chinese translation when training on the 400K set. In addition, increasing the size of the training data from 200K to 400K sentence pairs widens the performance margin between the baseline and our system, in some cases considerably. All but one of the performance improvements are found to be statistically significant (Koehn, 2004) at the 95% confidence level, most of them also at the 99% level.

We selected an array of target languages of increasing reordering complexity with English as source. Examining the results across the target languages, LTS performance gains increase the more challenging the sentence structure of the target language is in relation to the source's, highlighted when translating to Chinese. Even for Dutch and German, which pose additional challenges such as compound words and morphology which we do not explicitly treat in the current system, LTS still delivers significant improvements in performance. Additionally,

	System	200K	400K
(a)	lts-nolabels	22.50	24.24
	lts	**23.67****	**25.16****
(b)	josh-base-lm4	23.81	24.77
	lts-lm4	**24.48****	**26.35****

Table 2: Additional experiments for English to Chinese translation examining (a) the impact of the linguistic annotations in the LTS system (lts), when compared with an instance not employing such annotations (lts-nolabels) and (b) decoding with a 4th-order language model (-lm4). BLEU scores for 200K and 400K training sentence pairs.

the robustness of our system is exemplified by delivering significant performance increases for all language pairs.

For the English to Chinese translation task, we performed further experiments along two axes. We first investigate the contribution of the linguistic annotations, by comparing our complete system (lts) with an otherwise identical implementation (lts-nolabels) which does not employ any linguistically motivated labels. The latter system then uses a labels chart as that of Figure 3, which however labels all phrase-pair spans solely with the generic X label. The results in Table 2(a) indicate that a large part of the performance improvement can be attributed to the use of the linguistic annotations extracted from the source parse trees, indicating the potential of the LTS system to take advantage of such additional annotations to deliver better translations.

The second additional experiment relates to the impact of employing a stronger language model during decoding, which may increase performance but slows down decoding speed. Notably, as can be seen in Table 2(b), switching to a 4-gram LM results in performance gains for both the baseline and our system and while the margin between the two systems decreases, our system continues to deliver a considerable and significant improvement in translation BLEU scores.

5 Related Work

In this work, we focus on the combination of learning latent structure with syntax and linguistic annotations, exploring the crossroads of machine learning, linguistic syntax and machine translation. Training a joint probability model was first discussed in (Marcu and Wong, 2002). We show that a translation system based on such a joint model can perform competitively in comparison with conditional probability models, when it is augmented with a rich latent hierarchical structure trained adequately to avoid overfitting.

Earlier approaches for linguistic syntax-based translation such as (Yamada and Knight, 2001; Galley et al., 2006; Huang et al., 2006; Liu et al., 2006) focus on memorising and reusing parts of the structure of the source and/or target parse trees and constraining decoding by the input parse tree. In contrast to this approach, we choose to employ linguistic annotations in the form of unambiguous synchronous span labels, while discovering ambiguous translation structure taking advantage of them.

Later work (Marton and Resnik, 2008; Venugopal et al., 2009; Chiang et al., 2009) takes a more flexible approach, influencing translation output using linguistically motivated features, or features based on source-side linguistically-guided latent syntactic categories (Huang et al., 2010). A feature-based approach and ours are not mutually exclusive, as we also employ a limited set of features next to our trained model during decoding. We find augmenting our system with a more extensive feature set an interesting research direction for the future.

An array of recent work (Chiang, 2010; Zhang et al., 2008; Liu et al., 2009) sets off to utilise source *and* target syntax for translation. While for this work we constrain ourselves to source language syntax annotations, our method can be directly applied to employ labels taking advantage of linguistic annotations from both sides of translation. The decoding constraints of section 4.2 can then still be applied on the source part of hybrid source-target labels.

For the experiments in this paper we employ a label set similar to the non-terminals set of (Zollmann and Venugopal, 2006). However, the synchronous grammars we learn share few similarities with those that they heuristically extract. The HR-SCFG we adopt allows capturing more complex reordering phenomena and, in contrast to both (Chiang, 2005; Zollmann and Venugopal, 2006), is not exposed to the issues highlighted in section 2.1. Nevertheless, our results underline the capacity of linguistic anno-

tations similar to those of (Zollmann and Venugopal, 2006) as part of latent translation variables.

Most of the aforementioned work does concentrate on *learning* hierarchical, linguistically motivated translation models. Cohn and Blunsom (2009) sample rules of the form proposed in (Galley et al., 2004) from a Bayesian model, employing Dirichlet Process priors favouring smaller rules to avoid overfitting. Their grammar is however also based on the target parse-tree structure, with their system surpassing a weak baseline by a small margin. In contrast to the Bayesian approach which imposes external priors to lead estimation away from degenerate solutions, we take a *data-driven* approach to arrive to estimates which generalise well. The rich linguistically motivated latent variable learnt by our method delivers translation performance that compares favourably to a state-of-the-art system.

Mylonakis and Sima'an (2010) also employ the CV-EM algorithm to estimate the parameters of an SCFG, albeit a much simpler one based on a handful of non-terminals. In this work we employ some of their grammar design principles for an immensely more complex grammar with millions of hierarchical latent structure rules and show how such grammar can be learnt and applied taking advantage of source language linguistic annotations.

6 Conclusions

In this work we contribute a method to learn and apply latent hierarchical translation structure. To this end, we take advantage of source-language linguistic annotations to motivate instead of constrain the translation process. An input chart over phrase-pair spans, with each cell filled with multiple linguistically motivated labels, is coupled with the HR-SCFG design to arrive at a rich synchronous grammar with millions of structural rules and the capacity to capture complex linguistically conditioned translation phenomena. We address overfitting issues by cross-validating climbing the likelihood of the training data and propose solutions to increase the efficiency and accuracy of decoding.

An interesting aspect of our work is delivering competitive performance for difficult language pairs such as English-Chinese with a joint probability generative model and an SCFG without 'gap rules'. Instead of employing hierarchical phrase-pairs, we invest in learning the higher-order hierarchical synchronous structure behind translation, up to the full sentence length. While these choices and the related results challenge current MT research trends, they are not mutually exclusive with them. Future work directions include investigating the impact of hierarchical phrases for our models as well as any gains from additional features in the log-linear decoding model.

Smoothing the HR-SCFG grammar estimates could prove a possible source of further performance improvements. Learning translation and reordering behaviour with respect to linguistic cues is facilitated in our approach by keeping separate phrase-pair emission distributions per emitting non-terminal and reordering pattern, while the employment of the generic X non-terminals already allows backing off to more coarse-grained rules. Nevertheless, we still believe that further smoothing of these sparse distributions, e.g. by interpolating them with less sparse ones, could in the future lead to an additional increase in translation quality.

Finally, we discuss in this work how our method can already utilise hundreds of thousands of phrase-pair labels and millions of structural rules. A further promising direction is broadening this set with labels taking advantage of both source and target-language linguistic annotation or categories exploring additional phrase-pair properties past the parse trees such as semantic annotations.

Acknowledgments

Both authors are supported by a VIDI grant (nr. 639.022.604) from The Netherlands Organization for Scientific Research (NWO). The authors would like to thank Maxim Khalilov for helping with experimental data and Andreas Zollmann and the anonymous reviewers for their valuable comments.

References

Yehoshua Bar-Hillel. 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29(1):47–58.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the North American Association for Computational Linguistics (HLT/NAACL)*, Seattle, Washington, USA, April.

Stanley Chen and Joshua Goodman. 1998. *An empirical study of smoothing techniques for language modeling*. Technical Report TR-10-98, Harvard University, August.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 218–226, Boulder, Colorado, June. Association for Computational Linguistics.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL 2005*, pages 263–270.

David Chiang. 2010. Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1443–1452, Uppsala, Sweden, July. Association for Computational Linguistics.

Trevor Cohn and Phil Blunsom. 2009. A Bayesian model of syntax-directed tree to string grammar induction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 352–361, Singapore, August. Association for Computational Linguistics.

A.P. Dempster, N.M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. 2006. Why generative phrase models underperform surface heuristics. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 31–38, New York City. Association for Computational Linguistics.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 273–280, Boston, Massachusetts, USA, May. Association for Computational Linguistics.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 961–968, Sydney, Australia, July. Association for Computational Linguistics.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151, Prague, Czech Republic, June. Association for Computational Linguistics.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas (AMTA)*, Boston, MA, USA.

Zhongqiang Huang, Martin Cmejrek, and Bowen Zhou. 2010. Soft syntactic constraints for hierarchical phrase-based translation using latent syntactic distributions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 138–147, Cambridge, MA, October. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *HLT-NAACL 2003*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *MT Summit 2005*.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar Zaidan. 2009. Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 135–139, Athens, Greece, March. Association for Computational Linguistics.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 609–616, Sydney, Australia, July. Association for Computational Linguistics.

Yang Liu, Yajuan Lü, and Qun Liu. 2009. Improving tree-to-tree translation with packed forests. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 558–566, Suntec, Singapore, August. Association for Computational Linguistics.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of Empirical methods in natural language processing*, pages 133–139. Association for Computational Linguistics.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of ACL-08: HLT*, pages 1003–1011,

Columbus, Ohio, June. Association for Computational Linguistics.

Markos Mylonakis and Khalil Sima'an. 2008. Phrase translation probabilities with ITG priors and smoothing as learning objective. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 630–639, Honolulu, USA, October.

Markos Mylonakis and Khalil Sima'an. 2010. Learning probabilistic synchronous CFGs for phrase-based translation. In *Fourteenth Conference on Computational Natural Language Learning*, Uppsala, Sweden, July.

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

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *Proceedings of 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, Michigan, USA, June.

Ashish Venugopal, Andreas Zollmann, Noah A. Smith, and Stephan Vogel. 2009. Preference grammars: Softening syntactic constraints to improve statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 236–244, Boulder, Colorado, June. Association for Computational Linguistics.

Wei Wang, Jonathan May, Kevin Knight, and Daniel Marcu. 2010. Re-structuring, re-labeling, and re-aligning for syntax-based machine translation. *Computational Linguistics*, 36(2):247–277.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530, Toulouse, France, July. Association for Computational Linguistics.

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008. A tree sequence alignment-based tree-to-tree translation model. In *Proceedings of ACL-08: HLT*, pages 559–567, Columbus, Ohio, June. Association for Computational Linguistics.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 138–141, New York City, June. Association for Computational Linguistics.

Phrase-Based Translation Model for Question Retrieval in Community Question Answer Archives

Guangyou Zhou, Li Cai, Jun Zhao*, and Kang Liu
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
95 Zhongguancun East Road, Beijing 100190, China
{gyzhou,lcai,jzhao,kliu}@nlpr.ia.ac.cn

Abstract

Community-based question answer (Q&A) has become an important issue due to the popularity of Q&A archives on the web. This paper is concerned with the problem of question retrieval. Question retrieval in Q&A archives aims to find historical questions that are semantically equivalent or relevant to the queried questions. In this paper, we propose a novel phrase-based translation model for question retrieval. Compared to the traditional word-based translation models, the phrase-based translation model is more effective because it captures contextual information in modeling the translation of phrases as a whole, rather than translating single words in isolation. Experiments conducted on real Q&A data demonstrate that our proposed phrase-based translation model significantly outperforms the state-of-the-art word-based translation model.

1 Introduction

Over the past few years, large scale question and answer (Q&A) archives have become an important information resource on the Web. These include the traditional Frequently Asked Questions (FAQ) archives and the emerging community-based Q&A services, such as Yahoo! Answers[1], Live QnA[2], and Baidu Zhidao[3].

*Correspondence author: jzhao@nlpr.ia.ac.cn

[1]http://answers.yahoo.com/

[2]http://qna.live.com/

[3]http://zhidao.baidu.com/

Community-based Q&A services can directly return answers to the queried questions instead of a list of relevant documents, thus provide an effective alternative to the traditional adhoc information retrieval. To make full use of the large scale archives of question-answer pairs, it is critical to have functionality helping users to retrieve historical answers (Duan et al., 2008). Therefore, it is a meaningful task to retrieve the questions that are semantically equivalent or relevant to the queried questions. For example in Table 1, given question Q_1, Q_2 can be returned and their answers will then be used to answer Q_1 because the answer of Q_2 is expected to partially satisfy the queried question Q_1. This is what we called *question retrieval* in this paper.

The major challenge for Q&A retrieval, as for most information retrieval models, such as vector space model (VSM) (Salton et al., 1975), Okapi model (Robertson et al., 1994), language model (LM) (Ponte and Croft, 1998), is the *lexical gap* (or *lexical chasm*) between the queried questions and the historical questions in the archives (Jeon et al., 2005; Xue et al., 2008). For example in Table 1, Q_1 and Q_2 are two semantically similar questions, but they have very few words in common. This prob-

Query:
Q_1: How to get rid of **stuffy nose**?
Expected:
Q_2: What is the best way to prevent a cold?
Not Expected:
Q_3: How do I air out my stuffy room?
Q_4: How do you make a nose bleed stop quicker?

Table 1: An example on question retrieval

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 653–662,
Portland, Oregon, June 19-24, 2011.

lem is more serious for Q&A retrieval, since the question-answer pairs are usually short and there is little chance of finding the same content expressed using different wording (Xue et al., 2008). To solve the lexical gap problem, most researchers regarded the question retrieval task as a statistical machine translation problem by using IBM model 1 (Brown et al., 1993) to learn the word-to-word translation probabilities (Berger and Lafferty, 1999; Jeon et al., 2005; Xue et al., 2008; Lee et al., 2008; Bernhard and Gurevych, 2009). Experiments consistently reported that the word-based translation models could yield better performance than the traditional methods (e.g., VSM. Okapi and LM). However, all these existing approaches are considered to be *context independent* in that they do not take into account any contextual information in modeling word translation probabilities. For example in Table 1, although neither of the individual word pair (e.g., "stuffy"/"cold" and "nose"/"cold") might have a high translation probability, the sequence of words "stuffy nose" can be easily translated from a single word "cold" in Q_2 with a relative high translation probability.

In this paper, we argue that it is beneficial to capture contextual information for question retrieval. To this end, inspired by the phrase-based statistical machine translation (SMT) systems (Koehn et al., 2003; Och and Ney, 2004), we propose a phrase-based translation model (P-Trans) for question retrieval, and we assume that question retrieval should be performed at the phrase level. This model learns the probability of translating one sequence of words (e.g., phrase) into another sequence of words, e.g., translating a phrase in a historical question into another phrase in a queried question. Compared to the traditional word-based translation models that account for translating single words in isolation, the phrase-based translation model is potentially more effective because it captures some contextual information in modeling the translation of phrases as a whole. More precise translation can be determined for phrases than for words. It is thus reasonable to expect that using such phrase translation probabilities as ranking features is likely to improve the question retrieval performance, as we will show in our experiments.

Unlike the general natural language translation, the parallel sentences between questions and answers in community-based Q&A have very different lengths, leaving many words in answers unaligned to any word in queried questions. Following (Berger and Lafferty, 1999), we restrict our attention to those phrase translations consistent with a good word-level alignment.

Specifically, we make the following contributions:

- we formulate the question retrieval task as a phrase-based translation problem by modeling the contextual information (in Section 3.1).
- we linearly combine the phrase-based translation model for the question part and answer part (in Section 3.2).
- we propose a linear ranking model framework for question retrieval in which different models are incorporated as features because the phrase-based translation model cannot be interpolated with a unigram language model (in Section 3.3).
- finally, we conduct the experiments on community-based Q&A data for question retrieval. The results show that our proposed approach significantly outperforms the baseline methods (in Section 4).

The remainder of this paper is organized as follows. Section 2 introduces the existing state-of-the-art methods. Section 3 describes our phrase-based translation model for question retrieval. Section 4 presents the experimental results. In Section 5, we conclude with ideas for future research.

2 Preliminaries

2.1 Language Model

The unigram language model has been widely used for question retrieval on community-based Q&A data (Jeon et al., 2005; Xue et al., 2008; Cao et al., 2010). To avoid zero probability, we use Jelinek-Mercer smoothing (Zhai and Lafferty, 2001) due to its good performance and cheap computational cost. So the ranking function for the query likelihood language model with Jelinek-Mercer smoothing can be

written as:

$$Score(\mathbf{q}, D) = \prod_{w \in \mathbf{q}} (1-\lambda) P_{ml}(w|D) + \lambda P_{ml}(w|C) \quad (1)$$

$$P_{ml}(w|D) = \frac{\#(w, D)}{|D|}, \;\; P_{ml}(w|C) = \frac{\#(w, C)}{|C|} \quad (2)$$

where $\mathbf{q}$ is the queried question, D is a document, C is background collection, λ is smoothing parameter. $\#(t, D)$ is the frequency of term t in D, $|D|$ and $|C|$ denote the length of D and C respectively.

2.2 Word-Based Translation Model

Previous work (Berger et al., 2000; Jeon et al., 2005; Xue et al., 2008) consistently reported that the word-based translation models (Trans) yielded better performance than the traditional methods (VSM, Okapi and LM) for question retrieval. These models exploit the word translation probabilities in a language modeling framework. Following Jeon et al. (2005) and Xue et al. (2008), the ranking function can be written as:

$$Score(\mathbf{q}, D) = \prod_{w \in \mathbf{q}} (1-\lambda) P_{tr}(w|D) + \lambda P_{ml}(w|C) \quad (3)$$

$$P_{tr}(w|D) = \sum_{t \in D} P(w|t) P_{ml}(t|D), \;\; P_{ml}(t|D) = \frac{\#(t, D)}{|D|} \quad (4)$$

where $P(w|t)$ denotes the translation probability from word t to word w.

2.3 Word-Based Translation Language Model

Xue et al. (2008) proposed to linearly mix two different estimations by combining language model and word-based translation model into a unified framework, called TransLM. The experiments show that this model gains better performance than both the language model and the word-based translation model. Following Xue et al. (2008), this model can be written as:

$$Score(\mathbf{q}, D) = \prod_{w \in \mathbf{q}} (1-\lambda) P_{mx}(w|D) + \lambda P_{ml}(w|C) \quad (5)$$

$$P_{mx}(w|D) = \alpha \sum_{t \in D} P(w|t) P_{ml}(t|D) + (1-\alpha) P_{ml}(w|D) \quad (6)$$

D:	… for good cold home remedies …	*document*
E:	[for, good, cold, home remedies]	*segmentation*
F:	[for$_1$, best$_2$, stuffy nose$_3$, home remedy$_4$]	*translation*
M:	(1→3, 2→1, 3→4, 4→2)	*permutation*
$\mathbf{q}$:	best home remedy for stuffy nose	*queried question*

Figure 1: Example describing the generative procedure of the phrase-based translation model.

3 Our Approach: Phrase-Based Translation Model for Question Retrieval

3.1 Phrase-Based Translation Model

Phrase-based machine translation models (Koehn et al., 2003; D. Chiang, 2005; Och and Ney, 2004) have shown superior performance compared to word-based translation models. In this paper, the goal of phrase-based translation model is to translate a document[4] D into a queried question $\mathbf{q}$. Rather than translating single words in isolation, the phrase-based model translates one sequence of words into another sequence of words, thus incorporating contextual information. For example, we might learn that the phrase "stuffy nose" can be translated from "cold" with relative high probability, even though neither of the individual word pairs (e.g., "stuffy"/"cold" and "nose"/"cold") might have a high word translation probability. Inspired by the work of (Sun et al., 2010; Gao et al., 2010), we assume the following generative process: first the document D is broken into K non-empty word sequences $\mathbf{t}_1, \ldots, \mathbf{t}_K$, then each $\mathbf{t}$ is translated into a new non-empty word sequence $\mathbf{w}_1, \ldots, \mathbf{w}_K$, and finally these phrases are permutated and concatenated to form the queried questions $\mathbf{q}$, where $\mathbf{t}$ and $\mathbf{w}$ denote the phrases or consecutive sequence of words.

To formulate this generative process, let E denote the segmentation of D into K phrases $\mathbf{t}_1, \ldots, \mathbf{t}_K$, and let F denote the K translation phrases $\mathbf{w}_1, \ldots, \mathbf{w}_K$ —we refer to these $(\mathbf{t}_i, \mathbf{w}_i)$ pairs as *bi-phrases*. Finally, let M denote a permutation of K elements representing the final reordering step. Figure 1 describes an example of the generative procedure.

Next let us place a probability distribution over rewrite pairs. Let $B(D, \mathbf{q})$ denote the set of E,

[4] In this paper, a document has the same meaning as a historical question-answer pair in the Q&A archives.

F, M triples that translate D into $\mathbf{q}$. Here we assume a uniform probability over segmentations, so the phrase-based translation model can be formulated as:

$$P(\mathbf{q}|D) \propto \sum_{\substack{(E,F,M)\in \\ B(D,\mathbf{q})}} P(F|D,E) \cdot P(M|D,E,F) \quad (7)$$

As is common practice in SMT, we use the maximum approximation to the sum:

$$P(\mathbf{q}|D) \approx \max_{\substack{(E,F,M)\in \\ B(D,\mathbf{q})}} P(F|D,E) \cdot P(M|D,E,F) \quad (8)$$

Although we have defined a generative model for translating D into $\mathbf{q}$, our goal is to calculate the ranking score function over existing $\mathbf{q}$ and D, rather than generating new queried questions. Equation (8) cannot be used directly for document ranking because $\mathbf{q}$ and D are often of very different lengths, leaving many words in D unaligned to any word in $\mathbf{q}$. This is the key difference between the community-based question retrieval and the general natural language translation. As pointed out by Berger and Lafferty (1999) and Gao et al. (2010), document-query translation requires a *distillation* of the document, while translation of natural language tolerates little being thrown away.

Thus we attempt to extract the *key document words* that form the distillation of the document, and assume that a queried question is translated only from the *key document words*. In this paper, the key document words are identified via word alignment. We introduce the "hidden alignments" $A = a_1 \dots a_j \dots a_J$, which describe the mapping from a word position j in queried question to a document word position $i = a_j$. The different alignment models we present provide different decompositions of $P(\mathbf{q}, A|D)$. We assume that the position of the *key document words* are determined by the Viterbi alignment, which can be obtained using IBM model 1 as follows:

$$\begin{aligned}\hat{A} &= \arg\max_{A} P(\mathbf{q}, A|D) \\ &= \arg\max_{A} \Big\{ P(J|I) \prod_{j=1}^{J} P(w_j|t_{a_j}) \Big\} \\ &= \Big[\arg\max_{a_j} P(w_j|t_{a_j}) \Big]_{j=1}^{J} \end{aligned} \quad (9)$$

Given $\hat{A}$, when scoring a given Q&A pair, we restrict our attention to those E, F, M triples that are consistent with $\hat{A}$, which we denote as $B(D, \mathbf{q}, \hat{A})$. Here, consistency requires that if two words are aligned in $\hat{A}$, then they must appear in the same bi-phrase $(\mathbf{t}_i, \mathbf{w}_i)$. Once the word alignment is fixed, the final permutation is uniquely determined, so we can safely discard that factor. Thus equation (8) can be written as:

$$P(\mathbf{q}|D) \approx \max_{(E,F,M)\in B(D,\mathbf{q},\hat{A})} P(F|D,E) \quad (10)$$

For the sole remaining factor $P(F|D,E)$, we make the assumption that a segmented queried question $F = \mathbf{w}_1, \dots, \mathbf{w}_K$ is generated from left to right by translating each phrase $\mathbf{t}_1, \dots, \mathbf{t}_K$ independently:

$$P(F|D,E) = \prod_{k=1}^{K} P(\mathbf{w}_k|\mathbf{t}_k) \quad (11)$$

where $P(\mathbf{w}_k|\mathbf{t}_k)$ is a phrase translation probability, the estimation will be described in Section 3.3.

To find the maximum probability assignment efficiently, we use a dynamic programming approach, somewhat similar to the monotone decoding algorithm described in (Och, 2002). We define α_j to be the probability of the most likely sequence of phrases covering the first j words in a queried question, then the probability can be calculated using the following recursion:

(1) Initialization:

$$\alpha_0 = 1 \quad (12)$$

(2) Induction:

$$\alpha_j = \sum_{j'<j, \mathbf{w}=w_{j'+1}\dots w_j} \Big\{ \alpha_{j'} P(\mathbf{w}|\mathbf{t}_{\mathbf{w}}) \Big\} \quad (13)$$

(3) Total:

$$P(\mathbf{q}|D) = \alpha_J \quad (14)$$

3.2 Phrase-Based Translation Model for Question Part and Answer Part

In Q&A, a document D is decomposed into $(\bar{\mathbf{q}}, \bar{\mathbf{a}})$, where $\bar{\mathbf{q}}$ denotes the question part of the historical question in the archives and $\bar{\mathbf{a}}$ denotes the answer part. Although it has been shown that doing Q&A retrieval based solely on the answer part does not perform well (Jeon et al., 2005; Xue et al., 2008), the answer part should provide additional evidence about relevance and, therefore, it should be combined with the estimation based on the question part.

In this combined model, $P(\mathbf{q}|\bar{\mathbf{q}})$ and $P(\mathbf{q}|\bar{\mathbf{a}})$ are calculated with equations (12) to (14). So $P(\mathbf{q}|D)$ will be written as:

$$P(\mathbf{q}|D) = \mu_1 P(\mathbf{q}|\bar{\mathbf{q}}) + \mu_2 P(\mathbf{q}|\bar{\mathbf{a}}) \quad (15)$$

where $\mu_1 + \mu_2 = 1$.

In equation (15), the relative importance of question part and answer part is adjusted through μ_1 and μ_2. When $\mu_1 = 1$, the retrieval model is based on phrase-based translation model for the question part. When $\mu_2 = 1$, the retrieval model is based on phrase-based translation model for the answer part.

3.3 Parameter Estimation

3.3.1 Parallel Corpus Collection

In Q&A archives, question-answer pairs can be considered as a type of parallel corpus, which is used for estimating the translation probabilities. Unlike the bilingual machine translation, the questions and answers in a Q&A archive are written in the same language, the translation probability can be calculated through setting either as the source and the other as the target. In this paper, $P(\bar{\mathbf{a}}|\bar{\mathbf{q}})$ is used to denote the translation probability with the question as the source and the answer as the target. $P(\bar{\mathbf{q}}|\bar{\mathbf{a}})$ is used to denote the opposite configuration.

For a given word or phrase, the related words or phrases differ when it appears in the question or in the answer. Following Xue et al. (2008), a pooling strategy is adopted. First, we pool the question-answer pairs used to learn $P(\bar{\mathbf{a}}|\bar{\mathbf{q}})$ and the answer-question pairs used to learn $P(\bar{\mathbf{q}}|\bar{\mathbf{a}})$, and then use IBM model 1 (Brown et al., 1993) to learn the combined translation probabilities. Suppose we use the collection $\{(\bar{\mathbf{q}}, \bar{\mathbf{a}})_1, \ldots, (\bar{\mathbf{q}}, \bar{\mathbf{a}})_m\}$ to learn $P(\bar{\mathbf{a}}|\bar{\mathbf{q}})$ and use the collection $\{(\bar{\mathbf{a}}, \bar{\mathbf{q}})_1, \ldots, (\bar{\mathbf{a}}, \bar{\mathbf{q}})_m\}$ to learn $P(\bar{\mathbf{q}}|\bar{\mathbf{a}})$, then $\{(\bar{\mathbf{q}}, \bar{\mathbf{a}})_1, \ldots, (\bar{\mathbf{q}}, \bar{\mathbf{a}})_m, (\bar{\mathbf{a}}, \bar{\mathbf{q}})_1, \ldots, (\bar{\mathbf{a}}, \bar{\mathbf{q}})_m\}$ is used here to learn the combination translation probability $P_{pool}(w_i|t_j)$.

3.3.2 Parallel Corpus Preprocessing

Unlike the bilingual parallel corpus used in SMT, our parallel corpus is collected from Q&A archives, which is more noisy. Directly using the IBM model 1 can be problematic, it is possible for translation model to contain "unnecessary" translations (Lee et al., 2008). In this paper, we adopt a variant of TextRank algorithm (Mihalcea and Tarau, 2004) to identify and eliminate unimportant words from parallel corpus, assuming that a word in a question or answer is unimportant if it holds a relatively low significance in the parallel corpus.

Following (Lee et al., 2008), the ranking algorithm proceeds as follows. First, all the words in a given document are added as vertices in a graph G. Then edges are added between words if the words co-occur in a fixed-sized window. The number of co-occurrences becomes the weight of an edge. When the graph is constructed, the score of each vertex is initialized as 1, and the PageRank-based ranking algorithm is run on the graph iteratively until convergence. The TextRank score of a word w in document D at kth iteration is defined as follows:

$$R^k_{w,D} = (1-d) + d \cdot \sum_{\forall j:(i,j)\in G} \frac{e_{i,j}}{\sum_{\forall l:(j,l)\in G} e_{j,l}} R^{k-1}_{w,D} \quad (16)$$

where d is a damping factor usually set to 0.85, and $e_{i,j}$ is an edge weight between i and j.

We use average *TextRank score* as threshold: words are removed if their scores are lower than the average score of all words in a document.

3.3.3 Translation Probability Estimation

After preprocessing the parallel corpus, we will calculate $P(\mathbf{w}|\mathbf{t})$, following the method commonly used in SMT (Koehn et al., 2003; Och, 2002) to extract bi-phrases and estimate their translation probabilities.

First, we learn the word-to-word translation probability using IBM model 1 (Brown et al., 1993). Then, we perform Viterbi word alignment according to equation (9). Finally, the bi-phrases that are consistent with the word alignment are extracted using the heuristics proposed in (Och, 2002). We set the maximum phrase length to five in our experiments.

After gathering all such bi-phrases from the training data, we can estimate conditional relative frequency estimates without smoothing:

$$P(\mathbf{w}|\mathbf{t}) = \frac{N(\mathbf{t}, \mathbf{w})}{N(\mathbf{t})} \quad (17)$$

where $N(\mathbf{t}, \mathbf{w})$ is the number of times that $\mathbf{t}$ is aligned to $\mathbf{w}$ in training data. These estimates are

source	stuffy nose	internet explorer
1	stuffy nose	internet explorer
2	cold	ie
3	stuffy	internet browser
4	sore throat	explorer
5	sneeze	browser

Table 2: Phrase translation probability examples. Each column shows the top 5 target phrases learned from the word-aligned question-answer pairs.

useful for contextual lexical selection with sufficient training data, but can be subject to data sparsity issues (Sun et al., 2010; Gao et al., 2010). An alternate translation probability estimate not subject to data sparsity is the so-called *lexical weight* estimate (Koehn et al., 2003). Let $P(w|t)$ be the word-to-word translation probability, and let A be the word alignment between $\mathbf{w}$ and $\mathbf{t}$. Here, the word alignment contains (i, j) pairs, where $i \in 1 \ldots |\mathbf{w}|$ and $j \in 0 \ldots |\mathbf{t}|$, with 0 indicating a null word. Then we use the following estimate:

$$P_t(\mathbf{w}|\mathbf{t}, A) = \prod_{i=1}^{|\mathbf{w}|} \frac{1}{|\{j|(j,i) \in A\}|} \sum_{\forall (i,j) \in A} P(w_i|t_j) \tag{18}$$

We assume that for each position in $\mathbf{w}$, there is either a single alignment to 0, or multiple alignments to non-zero positions in $\mathbf{t}$. In fact, equation (18) computes a product of per-word translation scores; the per-word scores are the averages of all the translations for the alignment links of that word. The word translation probabilities are calculated using IBM 1, which has been widely used for question retrieval (Jeon et al., 2005; Xue et al., 2008; Lee et al., 2008; Bernhard and Gurevych, 2009). These word-based scores of bi-phrases, though not as effective in contextual selection, are more robust to noise and sparsity.

A sample of the resulting phrase translation examples is shown in Table 2, where the top 5 target phrases are translated from the source phrases according to the phrase-based translation model. For example, the term "explorer" used alone, most likely refers to a person who engages in scientific exploration, while the phrase "internet explorer" has a very different meaning.

3.4 Ranking Candidate Historical Questions

Unlike the word-based translation models, the phrase-based translation model cannot be interpolated with a unigram language model. Following (Sun et al., 2010; Gao et al., 2010), we resort to a linear ranking framework for question retrieval in which different models are incorporated as features.

We consider learning a relevance function of the following general, linear form:

$$Score(\mathbf{q}, D) = \boldsymbol{\theta}^T \cdot \boldsymbol{\Phi}(\mathbf{q}, D) \tag{19}$$

where the feature vector $\boldsymbol{\Phi}(\mathbf{q}, D)$ is an arbitrary function that maps $(\mathbf{q}, D)$ to a real value, i.e., $\boldsymbol{\Phi}(\mathbf{q}, D) \in \mathbb{R}$. $\boldsymbol{\theta}$ is the corresponding weight vector, we optimize this parameter for our evaluation metrics directly using the Powell Search algorithm (Paul et al., 1992) via cross-validation.

The features used in this paper are as follows:

- **Phrase translation features (PT)**: $\boldsymbol{\Phi}_{PT}(\mathbf{q}, D, A) = \log P(\mathbf{q}|D)$, where $P(\mathbf{q}|D)$ is computed using equations (12) to (15), and the phrase translation probability $P(\mathbf{w}|\mathbf{t})$ is estimated using equation (17).

- **Inverted Phrase translation features (IPT)**: $\boldsymbol{\Phi}_{IPT}(D, \mathbf{q}, A) = \log P(D|\mathbf{q})$, where $P(D|\mathbf{q})$ is computed using equations (12) to (15) except that we set $\mu_2 = 0$ in equation (15), and the phrase translation probability $P(\mathbf{w}|\mathbf{t})$ is estimated using equation (17).

- **Lexical weight feature (LW)**: $\boldsymbol{\Phi}_{LW}(\mathbf{q}, D, A) = \log P(\mathbf{q}|D)$, here $P(\mathbf{q}|D)$ is computed by equations (12) to (15), and the phrase translation probability is computed as lexical weight according to equation (18).

- **Inverted Lexical weight feature (ILW)**: $\boldsymbol{\Phi}_{ILW}(D, \mathbf{q}, A) = \log P(D|\mathbf{q})$, here $P(D|\mathbf{q})$ is computed by equations (12) to (15) except that we set $\mu_2 = 0$ in equation (15), and the phrase translation probability is computed as lexical weight according to equation (18).

- **Phrase alignment features (PA)**: $\boldsymbol{\Phi}_{PA}(\mathbf{q}, D, B) = \sum_2^K |a_k - b_{k-1} - 1|$, where B is a set of K bi-phrases, a_k is the start position of the phrase in D that was translated

into the kth phrase in queried question, and b_{k-1} is the end position of the phrase in D that was translated into the $(k-1)$th phrase in queried question. The feature, inspired by the distortion model in SMT (Koehn et al., 2003), models the degree to which the queried phrases are reordered. For all possible B, we only compute the feature value according to the Viterbi alignment, $\hat{B} = \arg\max_B P(\mathbf{q}, B|D)$. We find $\hat{B}$ using the Viterbi algorithm, which is almost identical to the dynamic programming recursion of equations (12) to (14), except that the *sum* operator in equation (13) is replaced with the *max* operator.

- **Unaligned word penalty features (UWP)**: $\mathbf{\Phi}_{UWP}(\mathbf{q}, D)$, which is defined as the ratio between the number of unaligned words and the total number of words in queried questions.

- **Language model features (LM)**: $\mathbf{\Phi}_{LM}(\mathbf{q}, D, A) = \log P_{LM}(\mathbf{q}|D)$, where $P_{LM}(\mathbf{q}|D)$ is the unigram language model with Jelinek-Mercer smoothing defined by equations (1) and (2).

- **Word translation features (WT)**: $\mathbf{\Phi}_{WT}(\mathbf{q}, D) = \log P(\mathbf{q}|D)$, where $P(\mathbf{q}|D)$ is the word-based translation model defined by equations (3) and (4).

4 Experiments

4.1 Data Set and Evaluation Metrics

We collect the questions from Yahoo! Answers and use the *getByCategory* function provided in Yahoo! Answers API[5] to obtain Q&A threads from the Yahoo! site. More specifically, we utilize the *resolved* questions under the top-level category at Yahoo! Answers, namely "Computers & Internet". The resulting question repository that we use for question retrieval contains 518,492 questions. To learn the translation probabilities, we use about one million question-answer pairs from another data set.[6]

In order to create the test set, we randomly select 300 questions for this category, denoted as "CI_TST". To obtain the ground-truth of question retrieval, we employ the Vector Space Model (VSM) (Salton et al., 1975) to retrieve the top 20 results and obtain manual judgements. The top 20 results don't include the queried question itself. Given a returned result by VSM, an annotator is asked to label it with "relevant" or "irrelevant". If a returned result is considered semantically equivalent to the queried question, the annotator will label it as "relevant"; otherwise, the annotator will label it as "irrelevant". Two annotators are involved in the annotation process. If a conflict happens, a third person will make judgement for the final result. In the process of manually judging questions, the annotators are presented only the questions. Table 3 provides the statistics on the final test set.

	#queries	#returned	#relevant
CI_TST	300	6,000	798

Table 3: Statistics on the Test Data

We evaluate the performance of our approach using **Mean Average Precision (MAP)**. We perform a significant test, i.e., a t-test with a default significant level of 0.05. Following the literature, we set the parameters $\lambda = 0.2$ (Cao et al., 2010) in equations (1), (3) and (5), and $\alpha = 0.8$ (Xue et al., 2008) in equation (6).

4.2 Question Retrieval Results

We randomly divide the test questions into five subsets and conduct 5-fold cross-validation experiments. In each trial, we tune the parameters μ_1 and μ_2 with four of the five subsets and then apply it to one remaining subset. The experiments reported below are those averaged over the five trials.

Table 4 presents the main retrieval performance. Row 1 to row 3 are baseline systems, all these methods use word-based translation models and obtain the state-of-the-art performance in previous work (Jeon et al., 2005; Xue et al., 2008). Row 3 is similar to row 2, the only difference is that TransLM only considers the question part, while Xue et al. (2008) incorporates the question part and answer part. Row 4 and row 5 are our proposed phrase-based translation model with maximum phrase length of five. Row 4 is phrase-based translation model purely based on question part, this model is equivalent to

[5] http://developer.yahoo.com/answers

[6] The Yahoo! Webscope dataset Yahoo answers comprehensive questions and answers version 1.0.2, available at http://reseach.yahoo.com/Academic_Relations.

#	Methods	Trans Prob	MAP
1	Jeon et al. (2005)	P_{pool}	0.289
2	TransLM	P_{pool}	0.324
3	Xue et al. (2008)	P_{pool}	0.352
4	P-Trans ($\mu_1 = 1, l = 5$)	P_{pool}	0.366
5	**P-Trans** ($l = 5$)	P_{pool}	**0.391**

Table 4: Comparison with different methods for question retrieval.

setting $\mu_1 = 1$ in equation (15). Row 5 is the phrase-based combination model which linearly combines the question part and answer part. As expected, different parts can play different roles: a phrase to be translated in queried questions may be translated from the question part or answer part. All these methods use pooling strategy to estimate the translation probabilities. There are some clear trends in the result of Table 4:

(1) Word-based translation language model (TransLM) significantly outperforms word-based translation model of Jeon et al. (2005) (row 1 vs. row 2). Similar observations have been made by Xue et al. (2008).

(2) Incorporating the answer part into the models, either word-based or phrase-based, can significantly improve the performance of question retrieval (row 2 vs. row 3; row 4 vs. row 5).

(3) Our proposed phrase-based translation model (P-Trans) significantly outperforms the state-of-the-art word-based translation models (row 2 vs. row 4 and row 3 vs. row 5, all these comparisons are statistically significant at $p < 0.05$).

4.3 Impact of Phrase Length

Our proposed phrase-based translation model, due to its capability of capturing contextual information, is more effective than the state-of-the-art word-based translation models. It is important to investigate the impact of the phrase length on the final retrieval performance. Table 5 shows the results, it is seen that using the longer phrases up to the maximum length of five can consistently improve the retrieval performance. However, using much longer phrases in the phrase-based translation model does not seem to produce significantly better performance (row 8 and row 9 vs. row 10 are not statistically significant).

#	Systems	MAP
6	P-Trans ($l = 1$)	0.352
7	P-Trans ($l = 2$)	0.373
8	P-Trans ($l = 3$)	0.386
9	P-Trans ($l = 4$)	0.390
10	P-Trans ($l = 5$)	0.391

Table 5: The impact of the phrase length on retrieval performance.

Model	#	Methods	Average	MAP
P-Trans ($l = 5$)	11	Initial	69	0.380
	12	TextRank	24	0.391

Table 6: Effectiveness of parallel corpus preprocessing.

4.4 Effectiveness of Parallel Corpus Preprocessing

Question-answer pairs collected from Yahoo! answers are very noisy, it is possible for translation models to contain "unnecessary" translations. In this paper, we attempt to identify and decrease the proportion of unnecessary translations in a translation model by using TextRank algorithm. This kind of "unnecessary" translation between words will eventually affect the bi-phrase translation.

Table 6 shows the effectiveness of parallel corpus preprocessing. Row 11 reports the average number of translations per word and the question retrieval performance when only stopwords [7] are removed. When using the TextRank algorithm for parallel corpus preprocessing, the average number of translations per word is reduced from 69 to 24, but the performance of question retrieval is significantly improved (row 11 vs. row 12). Similar results have been made by Lee et al. (2008).

4.5 Impact of Pooling Strategy

The correspondence of words or phrases in the question-answer pair is not as strong as in the bilingual sentence pair, thus noise will be inevitably introduced for both $P(\bar{\mathbf{a}}|\bar{\mathbf{q}})$ and $P(\bar{\mathbf{q}}|\bar{\mathbf{a}})$.

To see how much the pooling strategy benefit the question retrieval, we introduce two baseline methods for comparison. The first method (denoted as $P(\bar{\mathbf{a}}|\bar{\mathbf{q}})$) is used to denote the translation probability with the question as the source and the answer as

[7] http://truereader.com/manuals/onix/stopwords1.html

Model	#	Trans Prob	MAP
P-Trans ($l = 5$)	13	$P(\bar{\mathbf{a}}\|\bar{\mathbf{q}})$	0.387
	14	$P(\bar{\mathbf{q}}\|\bar{\mathbf{a}})$	0.381
	15	P_{pool}	0.391

Table 7: The impact of pooling strategy for question retrieval.

the target. The second (denoted as $P(\bar{\mathbf{a}}|\bar{\mathbf{q}})$) is used to denote the translation probability with the answer as the source and the question as the target. Table 7 provides the comparison. From this Table, we see that the pooling strategy significantly outperforms the two baseline methods for question retrieval (row 13 and row 14 vs. row 15).

5 Conclusions and Future Work

In this paper, we propose a novel phrase-based translation model for question retrieval. Compared to the traditional word-based translation models, the proposed approach is more effective in that it can capture contextual information instead of translating single words in isolation. Experiments conducted on real Q&A data demonstrate that the phrase-based translation model significantly outperforms the state-of-the-art word-based translation models.

There are some ways in which this research could be continued. First, question structure should be considered, so it is necessary to combine the proposed approach with other question retrieval methods (e.g., (Duan et al., 2008; Wang et al., 2009; Bunescu and Huang, 2010)) to further improve the performance. Second, we will try to investigate the use of the proposed approach for other kinds of data set, such as categorized questions from forum sites and FAQ sites.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 60875041 and No. 61070106). We thank the anonymous reviewers for their insightful comments. We also thank Maoxi Li and Jiajun Zhang for suggestion to use the alignment toolkits.

References

A. Berger and R. Caruana and D. Cohn and D. Freitag and V. Mittal. 2000. Bridging the lexical chasm: statistical approach to answer-finding. In *Proceedings of SIGIR*, pages 192-199.

A. Berger and J. Lafferty. 1999. Information retrieval as statistical translation. In *Proceedings of SIGIR*, pages 222-229.

D. Bernhard and I. Gurevych. 2009. Combining lexical semantic resources with question & answer archives for translation-based answer finding. In *Proceedings of ACL*, pages 728-736.

P. F. Brown and V. J. D. Pietra and S. A. D. Pietra and R. L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263-311.

R. Bunescu and Y. Huang. 2010. Learning the relative usefulness of questions in community QA. In *Proceedings of EMNLP*, pages 97-107.

X. Cao and G. Cong and B. Cui and C. S. Jensen. 2010. A generalized framework of exploring category information for question retrieval in community question answer archives. In *Proceedings of WWW*.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*.

H. Duan and Y. Cao and C. Y. Lin and Y. Yu. 2008. Searching questions by identifying questions topics and question focus. In *Proceedings of ACL*, pages 156-164.

J. Gao and X. He and J. Nie. 2010. Clickthrough-based translation models for web search: from word models to phrase models. In *Proceedings of CIKM*.

J. Jeon and W. Bruce Croft and J. H. Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of CIKM*, pages 84-90.

R. Mihalcea and P. Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of EMNLP*, pages 404-411.

P. Koehn and F. Och and D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*, pages 48-54.

J. -T. Lee and S. -B. Kim and Y. -I. Song and H. -C. Rim. 2008. Bridging lexical gaps between queries and questions on large online Q&A collections with compact translation models. In *Proceedings of EMNLP*, pages 410-418.

F. Och. 2002. Statistical mahcine translation: from single word models to alignment templates. Ph.D thesis, RWTH Aachen.

F. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417-449.

J. M. Ponte and W. B. Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of SIGIR*.

W. H. Press and S. A. Teukolsky and W. T. Vetterling and B. P. Flannery. 1992. *Numerical Recipes In C*. Cambridge Univ. Press.

S. Robertson and S. Walker and S. Jones and M. Hancock-Beaulieu and M. Gatford. 1994. Okapi at trec-3. In *Proceedings of TREC*, pages 109-126.

G. Salton and A. Wong and C. S. Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613-620.

X. Sun and J. Gao and D. Micol and C. Quirk. 2010. Learning phrase-based spelling error models from clickthrough data. In *Proceedings of ACL*.

K. Wang and Z. Ming and T-S. Chua. 2009. A syntactic tree matching approach to finding similar questions in community-based qa services. In *Proceedings of SIGIR*, pages 187-194.

X. Xue and J. Jeon and W. B. Croft. 2008. Retrieval models for question and answer archives. In *Proceedings of SIGIR*, pages 475-482.

C. Zhai and J. Lafferty. 2001. A study of smooth methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR*, pages 334-342.

Neutralizing Linguistically Problematic Annotations in Unsupervised Dependency Parsing Evaluation

Roy Schwartz[1] **Omri Abend**[1*] **Roi Reichart**[2] **Ari Rappoport**[1]

[1]Institute of Computer Science
Hebrew University of Jerusalem
{roys02|omria01|arir}@cs.huji.ac.il

[2]Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
roiri@csail.mit.edu

Abstract

Dependency parsing is a central NLP task. In this paper we show that the common evaluation for unsupervised dependency parsing is highly sensitive to problematic annotations. We show that for three leading unsupervised parsers (Klein and Manning, 2004; Cohen and Smith, 2009; Spitkovsky et al., 2010a), a small set of parameters can be found whose modification yields a significant improvement in standard evaluation measures. These parameters correspond to local cases where no linguistic consensus exists as to the proper gold annotation. Therefore, the standard evaluation does not provide a true indication of algorithm quality. We present a new measure, *Neutral Edge Direction* (NED), and show that it greatly reduces this undesired phenomenon.

1 Introduction

Unsupervised induction of dependency parsers is a major NLP task that attracts a substantial amount of research (Klein and Manning, 2004; Cohen et al., 2008; Headden et al., 2009; Spitkovsky et al., 2010a; Gillenwater et al., 2010; Berg-Kirkpatrick et al., 2010; Blunsom and Cohn, 2010, *inter alia*). Parser quality is usually evaluated by comparing its output to a gold standard whose annotations are linguistically motivated. However, there are cases in which there is no linguistic consensus as to what the correct annotation is (Kübler et al., 2009). Examples include which verb is the head in a verb group structure (e.g., "can" or "eat" in "can eat"), and which noun is the head in a sequence of proper nouns (e.g., "John" or "Doe" in "John Doe"). We refer to such annotations as (*linguistically*) *problematic*. For such cases, evaluation measures should not punish the algorithm for deviating from the gold standard.

In this paper we show that the evaluation measures reported in current works are highly sensitive to the annotation in problematic cases, and propose a simple new measure that greatly neutralizes the problem.

We start from the following observation: for three leading algorithms (Klein and Manning, 2004; Cohen and Smith, 2009; Spitkovsky et al., 2010a), a small set (at most 18 out of a few thousands) of parameters can be found whose modification dramatically improves the standard evaluation measures (the attachment score measure by 9.3-15.1%, and the undirected measure by a smaller but still significant 1.3-7.7%). The phenomenon is implementation independent, occurring with several algorithms based on a fundamental probabilistic dependency model[1].

We show that these parameter changes can be mapped to edge direction changes in local structures in the dependency graph, and that these correspond to problematic annotations. Thus, the standard evaluation measures do not reflect the true quality of the evaluated algorithm.

We explain why the standard undirected evaluation measure is in fact sensitive to such edge direc-

* Omri Abend is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship.

[1]It is also language-independent; we have produced it in five different languages: English, Czech, Japanese, Portuguese, and Turkish. Due to space considerations, in this paper we focus on English, because it is the most studied language for this task and the most practically useful one at present.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 663–672,
Portland, Oregon, June 19-24, 2011.

tion changes, and present a new evaluation measure, *Neutral Edge Direction* (NED), which greatly alleviates the problem by ignoring the edge direction in local structures. Using NED, manual modifications of model parameters always yields small performance differences. Moreover, NED sometimes punishes such manual parameter tweaking by yielding worse results. We explain this behavior using an experiment revealing that NED always prefers the structures that are more consistent with the modeling assumptions lying in the basis of the algorithm. When manual parameter modification is done against this preference, the NED results decrease.

The contributions of this paper are as follows. First, we show the impact of a small number of annotation decisions on the performance of unsupervised dependency parsers. Second, we observe that often these decisions are linguistically controversial and therefore this impact is misleading. This reveals a problem in the common evaluation of unsupervised dependency parsing. This is further demonstrated by noting that recent papers evaluate the task using three gold standards which differ in such decisions and which yield substantially different results. Third, we present the NED measure, which is agnostic to errors arising from choosing the non-gold direction in such cases.

Section 2 reviews related work. Section 3 describes the performed parameter modifications. Section 4 discusses the linguistic controversies in annotating problematic dependency structures. Section 5 presents NED. Section 6 describes experiments with it. A discussion is given in Section 7.

2 Related Work

Grammar induction received considerable attention over the years (see (Clark, 2001; Klein, 2005) for reviews). For unsupervised dependency parsing, the *Dependency Model with Valence (DMV)* (Klein and Manning, 2004) was the first to beat the simple right-branching baseline. A technical description of DMV is given at the end of this section.

The great majority of recent works, including those experimented with in this paper, are elaborations of DMV. Smith and Eisner (2005) improved the DMV results by generalizing the function maximized by DMV's EM training algorithm. Smith and Eisner (2006) used a structural locality bias, experimenting on five languages. Cohen et al. (2008) extended DMV by using a variational EM training algorithm and adding logistic normal priors. Cohen and Smith (2009, 2010) further extended it by using a *shared* logistic normal prior which provided a new way to encode the knowledge that some POS tags are more similar than others. A bilingual joint learning further improved their performance.

Headden et al. (2009) obtained the best reported results on WSJ10 by using a lexical extension of DMV. Gillenwater et al. (2010) used posterior regularization to bias the training towards a small number of parent-child combinations. Berg-Kirkpatrick et al. (2010) added new features to the M step of the DMV EM procedure. Berg-Kirkpatrick and Klein (2010) used a phylogenetic tree to model parameter drift between different languages. Spitkovsky et al. (2010a) explored several training protocols for DMV. Spitkovsky et al. (2010c) showed the benefits of Viterbi ("hard") EM to DMV training. Spitkovsky et al. (2010b) presented a novel *lightly-supervised* approach that used hyper-text mark-up annotation of web-pages to train DMV.

A few non-DMV-based works were recently presented. Daumé III (2009) used shift-reduce techniques. Blunsom and Cohn (2010) used tree substitution grammar to achieve best results on WSJ$^\infty$.

Druck et al. (2009) took a semi-supervised approach, using a set of rules such as "A noun is usually the parent of a determiner which is to its left", experimenting on several languages. Naseem et al. (2010) further extended this idea by using a single set of rules which globally applies to six different languages. The latter used a model similar to DMV.

The controversial nature of some dependency structures was discussed in (Nivre, 2006; Kübler et al., 2009). Klein (2005) discussed controversial constituency structures and the evaluation problems stemming from them, stressing the importance of a consistent standard of evaluation.

A few works explored the effects of annotation conventions on parsing performance. Nilsson et al. (2006) transformed the dependency annotations of coordinations and verb groups in the Prague TreeBank. They trained the supervised MaltParser (Nivre et al., 2006) on the transformed data, parsed the test data and re-transformed the resulting parse,

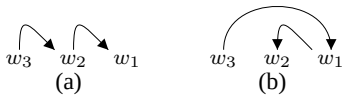


Figure 1: A dependency structure on the words w_1, w_2, w_3 before (Figure 1(a)) and after (Figure 1(b)) an *edge-flip* of $w_2 \rightarrow w_1$.

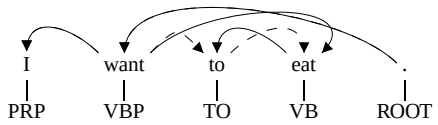


Figure 2: A parse of the sentence "I want to eat", before (straight line) and after (dashed line) an *edge-flip* of the edge "to"←"eat".

thus improving performance. Klein and Manning (2004) observed that a large portion of their errors is caused by predicting the wrong direction of the edge between a noun and its determiner. Kübler (2005) compared two different conversion schemes in German supervised constituency parsing and found one to have positive influence on parsing quality.

Dependency Model with Valence (DMV). DMV (Klein and Manning, 2004) defines a probabilistic grammar for unlabeled dependency structures. It is defined as follows: the root of the sentence is first generated, and then each head recursively generates its right and left dependents. The parameters of the model are of two types: P_{STOP} and P_{ATTACH}. $P_{STOP}(dir, h, adj)$ determines the probability to stop generating arguments, and is conditioned on 3 arguments: the head h, the direction dir ($(L)eft$ or $(R)ight$) and adjacency adj (whether the head already has dependents ($(Y)es$) in direction dir or not ($(N)o$)). $P_{ATTACH}(arg|h, dir)$ determines the probability to generate arg as head h's dependent in direction dir.

3 Significant Effects of Edge Flipping

In this section we present recurring error patterns in some of the leading unsupervised dependency parsers. These patterns are all local, confined to a sequence of up to three words (but mainly of just two consecutive words). They can often be mended by changing the directions of a few types of edges.

The modified parameters described in this section were handpicked to improve performance: we examined the local parser errors occurring the largest number of times, and found the corresponding parameters. Note that this is a valid methodology, since our goal is not to design a new algorithm but to demonstrate that modifying a small set of parameters can yield a major performance boost and eventually discover problems with evaluation methods or algorithms.

We start with a few definitions. Consider Figure 1(a) that shows a dependency structure on the words w_1, w_2, w_3. Edge flipping (henceforth, *edge-flip*) the edge $w_2 \rightarrow w_1$ is the following modification of a parse tree: (1) setting w_2's parent as w_1 (instead of the other way around), and (2) setting w_1's parent as w_3 (instead of the edge $w_3 \rightarrow w_2$). Figure 1(b) shows the dependency structure after the *edge-flip*.

Note that (1) imposes setting a new parent to w_2, as otherwise it would have had no parent. Setting this parent to be w_3 is the minimal modification of the original parse, since it does not change the attachment of the structure $[w_2, w_1]$ to the rest of the sentence, but only the direction of the internal edge.

Figure 2 presents a parse of the sentence "I want to eat", before and after an *edge-flip* of the edge "to"←"eat".

Since unsupervised dependency parsers are generally structure prediction models, the predictions of the parse edges are not independent. Therefore, there is no single parameter which completely controls the edge direction, and hence there is no direct way to perform an *edge-flip* by parameter modification. However, setting extreme values for the parameters controlling the direction of a certain edge type creates a strong preference towards one of the directions, and effectively determines the edge direction. This procedure is henceforth termed *parameter-flip*.

We show that by performing a few *parameter-flips*, a substantial improvement in the attachment score can be obtained. Results are reported for three algorithms.

Parameter Changes. All the works experimented with in this paper are not lexical and use sequences of POS tags as their input. In addition, they all use the DMV parameter set (P_{STOP} and P_{ATTACH}) for parsing. We will henceforth refer to this set, conditioned on POS tags, as the model parameter set.

We show how an edge in the dependency graph is encoded using the DMV parameters. Say the

model prefers setting "to" (POS tag: TO) as a dependent of the infinitive verb (POS tag: VB) to its right (e.g., "to eat"). This is reflected by a high value of $P_{ATTACH}(TO|VB, L)$, a low value of $P_{ATTACH}(VB|TO, R)$, since "to" tends to be a left dependent of the verb and not the other way around, and a low value of $P_{STOP}(VB, L, N)$, as the verb usually has at least one left argument (i.e., "to").

A *parameter-flip* of $w_1 \rightarrow w_2$ is hence performed by setting $P_{ATTACH}(w_2|w_1, R)$ to a very low value and $P_{ATTACH}(w_1|w_2, L)$ to a very high value. When the modifications to P_{ATTACH} are insufficient to modify the edge direction, $P_{STOP}(w_2, L, N)$ is set to a very low value and $P_{STOP}(w_1, R, N)$ to a very high value[2].

Table 1 describes the changes made for the three algorithms. The '+' signs in the table correspond to edges in which the algorithm disagreed with the gold standard, and were thus modified. Similarly, the '–' signs in the table correspond to edges in which the algorithm agreed with the gold standard, and were thus not modified. The number of modified parameters does not exceed 18 (out of a few thousands).

The *Freq*. column in the table shows the percentage of the tokens in sections 2-21 of PTB WSJ that participate in each structure. Equivalently, the percentage of edges in the corpus which are of either of the types appearing in the *Orig*. *Edge* column. As the table shows, the modified structures cover a significant portion of the tokens. Indeed, 42.9% of the tokens in the corpus participate in at least one of them[3].

Experimenting with Edge Flipping. We experimented with three DMV-based algorithms: a replication of (Klein and Manning, 2004), as appears in (Cohen et al., 2008) (henceforth, *km04*), Cohen and Smith (2009) (henceforth, *cs09*), and Spitkovsky et al. (2010a) (henceforth, *saj10a*). Decoding is done using the Viterbi algorithm[4]. For each of these algorithms we present the performance gain when compared to the original parameters.

The training set is sections 2-21 of the Wall Street Journal Penn TreeBank (Marcus et al., 1993). Testing is done on section 23. The constituency annotation was converted to dependencies using the rules of (Yamada and Matsumoto, 2003)[5].

Following standard practice, we present the attachment score (i.e., percentage of words that have a correct head) of each algorithm, with both the original parameters and the modified ones. We present results both on all sentences and on sentences of length ≤ 10, excluding punctuation.

Table 2 shows results for all algorithms[6]. The performance difference between the original and the modified parameter set is considerable for all data sets, where differences exceed 9.3%, and go up to 15.1%. These are enormous differences from the perspective of current algorithm evaluation results.

Structure	Freq.	Orig. Edge	*km04*	*cs09*	*saj10a*
Coordination ("John & Mary")	2.9%	$CC \rightarrow NNP$	–	+	–
Prepositional Phrase ("in the house")	32.7%	$DT \rightarrow NN$	+	+	+
		$DT \rightarrow NNP$	–	+	+
		$DT \rightarrow NNS$	–	–	+
		$IN \rightarrow DT$	+	+	–
		$IN \leftarrow NN$	+	+	–
		$IN \leftarrow NNP$	+	–	–
		$IN \leftarrow NNS$	–	+	–
		$PRP\$ \rightarrow NN$	–	–	+
Modal Verb ("can eat")	2.4%	$MD \leftarrow VB$	–	+	–
Infinitive Verb ("to eat")	4.5%	$TO \rightarrow VB$	–	+	+
Proper Name Sequence ("John Doe")	18.5%	$NNP \rightarrow NNP$	+	–	–

Table 1: Parameter changes for the three algorithms. The Freq. column shows what percentage of the tokens in sections 2-21 of PTB WSJ participate in each structure. The Orig. column indicates the original edge. The modified edge is of the opposite direction. The other columns show the different algorithms: *km04*: basic DMV model (replication of (Klein and Manning, 2004)); *cs09*; (Cohen and Smith, 2009); *saj10a*: (Spitkovsky et al., 2010a).

4 Linguistically Problematic Annotations

In this section, we discuss the controversial nature of the annotation in the modified structures (Kübler

[2]Note that this yields unnormalized models. Again, this is justified since the resulting model is only used as a basis for discussion and is not a fully fledged algorithm.

[3]Some tokens participate in more than one structure.

[4]http://www.cs.cmu.edu/∼scohen/parser.html.

[5]http://www.jaist.ac.jp/∼h-yamada/

[6]Results are slightly worse than the ones published in the original papers due to the different decoding algorithms (*cs09* use MBR while we used Viterbi) and a different conversion procedure (*saj10a* used (Collins, 1999) and not (Yamada and Matsumoto, 2003)) ; see Section 5.

Algo.	≤ 10			$\leq \infty$		
	$Orig.$	$Mod.$	Δ	$Orig.$	$Mod.$	Δ
km04	45.8	59.8	14	34.6	43.9	9.3
cs09	60.9	72.9	12	39.9	54.6	14.7
saj10a	54.7	69.8	15.1	41.6	54.3	12.7

Table 2: Results of the original ($Orig.$ columns), the modified ($Mod.$ columns) parameter sets and their difference (Δ columns) for the three algorithms.

et al., 2009). We remind the reader that structures for which no linguistic consensus exists as to their correct annotation are referred to as (linguistically) problematic.

We begin by showing that all the structures modified are indeed linguistically problematic. We then note that these controversies are reflected in the evaluation of this task, resulting in three, significantly different, gold standards currently in use.

Coordination Structures are composed of two proper nouns, separated by a conjunctor (e.g., "John and Mary"). It is not clear which token should be the head of this structure, if any (Nilsson et al., 2006).

Prepositional Phrases (e.g., "in the house" or "in Rome"), where every word is a reasonable candidate to head this structure. For example, in the annotation scheme used by (Collins, 1999) the preposition is the head, in the scheme used by (Johansson and Nugues, 2007) the noun is the head, while TUT annotation, presented in (Bosco and Lombardo, 2004), takes the determiner to be the noun's head.

Verb Groups are composed of a verb and an auxiliary or a modal verb (e.g., "can eat"). Some schemes choose the modal as the head (Collins, 1999), others choose the verb (Rambow et al., 2002).

Infinitive Verbs (e.g., "to eat") are also in controversy, as in (Yamada and Matsumoto, 2003) the verb is the head while in (Collins, 1999; Bosco and Lombardo, 2004) the "to" token is the head.

Sequences of Proper Nouns (e.g., "John Doe") are also subject to debate, as PTB's scheme takes the last proper noun as the head, and BIO's scheme defines a more complex scheme (Dredze et al., 2007).

Evaluation Inconsistency Across Papers. A fact that may not be recognized by some readers is that comparing the results of unsupervised dependency parsers across different papers is not directly possible, since different papers use different gold standard annotations *even when they are all derived from the Penn Treebank constituency annotation*. This happens because they use different rules for converting constituency annotation to dependency annotation. A probable explanation for this fact is that people have tried to correct linguistically problematic annotations in different ways, which is why we note this issue here[7].

There are three different annotation schemes in current use: (1) Collins head rules (Collins, 1999), used in e.g., (Berg-Kirkpatrick et al., 2010; Spitkovsky et al., 2010a); (2) Conversion rules of (Yamada and Matsumoto, 2003), used in e.g., (Cohen and Smith, 2009; Gillenwater et al., 2010); (3) Conversion rules of (Johansson and Nugues, 2007) used, e.g., in the CoNLL shared task 2007 (Nivre et al., 2007) and in (Blunsom and Cohn, 2010).

The differences between the schemes are substantial. For instance, 14.4% of section 23 is tagged differently by (1) and (2)[8].

5 The Neutral Edge Direction (NED) Measure

As shown in the previous sections, the annotation of problematic edges can substantially affect performance. This was briefly discussed in (Klein and Manning, 2004), which used undirected evaluation as a measure which is less sensitive to alternative annotations. Undirected accuracy was commonly used since to assess the performance of unsupervised parsers (e.g., (Smith and Eisner, 2006; Headden et al., 2008; Spitkovsky et al., 2010a)) but also of supervised ones (Wang et al., 2005; Wang et al., 2006). In this section we discuss why this measure is in fact not indifferent to *edge-flips* and propose a new measure, Neutral Edge Direction (NED).

[7] Indeed, half a dozen flags in the LTH Constituent-to-Dependency Conversion Tool (Johansson and Nugues, 2007) are used to control the conversion in problematic cases.

[8] In our experiments we used the scheme of (Yamada and Matsumoto, 2003), see Section 3. The significant effects of edge flipping were observed with the other two schemes as well.

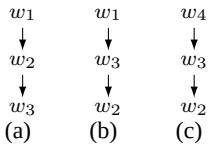


Figure 3: A dependency structure on the words w_1, w_2, w_3 before (Figure 3(a)) and after (Figure 3(b)) an *edge-flip* of $w_2 \rightarrow w_3$, and when the direction of the edge between w_2 and w_3 is switched and the new parent of w_3 is set to be some other word, w_4 (Figure 3(c)).

Undirected Evaluation. The measure is defined as follows: traverse over the tokens and mark a correct attachment if the token's induced parent is either (1) its gold parent or (2) its gold child. The score is the ratio of correct attachments and the number of tokens.

We show that this measure does not ignore *edge-flips*. Consider Figure 3 that shows a dependency structure on the words w_1, w_2, w_3 before (Figure 3(a)) and after (Figure 3(b)) an *edge-flip* of $w_2 \rightarrow w_3$. Assume that 3(a) is the gold standard and that 3(b) is the induced parse. Consider w_2. Its induced parent (w_3) is its gold child, and thus undirected evaluation does not consider it an error. On the other hand, w_3 is assigned w_2's gold parent, w_1. This is considered an error, since w_1 is neither w_3's gold parent (as it is w_2), nor its gold child[9]. Therefore, one of the two tokens involved in the *edge-flip* is penalized by the measure.

Recall the example "I want to eat" and the *edge-flip* of the edge "to"←"eat" (Figure 2). As "to"'s parent in the induced graph ("want") is neither its gold parent nor its gold child, the undirected evaluation measure marks it as an error. This is an example where an *edge-flip* in a problematic edge, which should not be considered an error, was in fact considered an error by undirected evaluation.

Neutral Edge Direction (NED). The NED measure is a simple extension of the undirected evaluation measure[10]. Unlike undirected evaluation, NED ignores all errors directly resulting from an *edge-flip*.

NED is defined as follows: traverse over the tokens and mark a correct attachment if the token's induced parent is either (1) its gold parent (2) its gold child or (3) its gold grandparent. The score is the ratio of correct attachments and the number of tokens.

NED, by its definition, ignores *edge-flips*. Consider again Figure 3, where we assume that 3(a) is the gold standard and that 3(b) is the induced parse. Much like undirected evaluation, NED will mark the attachment of w_2 as correct, since its induced parent is its gold child. However, unlike undirected evaluation, w_3's induced attachment will also be marked as correct, as its induced parent is its gold grandparent.

Now consider another induced parse in which the direction of the edge between w_2 and w_3 is switched and the w_3's parent is set to be some other word, w_4 (Figure 3(c)). This should be marked as an error, even if the direction of the edge between w_2 and w_3 is controversial, since the structure $[w_2, w_3]$ is no longer a dependent of w_1. It is indeed a NED error. Note that undirected evaluation gives the parses in Figure 3(b) and Figure 3(c) the same score, while if the structure $[w_2, w_3]$ is problematic, there is a major difference in their correctness.

Discussion. Problematic structures are ubiquitous, with more than 40% of the tokens in PTB WSJ appearing in at least one of them (see Section 3). Therefore, even a substantial difference in the attachment between two parsers is not necessarily indicative of a true quality difference. However, an attachment score difference that persists under NED is an indication of a true quality difference, since generally problematic structures are local (i.e., obtained by an *edge-flip*) and NED ignores such errors.

Reporting NED alone is insufficient, as obviously the edge direction does matter in some cases. For example, in adjective–noun structures (e.g., "big house"), the correct edge direction is widely agreed upon ("big"←"house") (Kübler et al., 2009), and thus choosing the wrong direction should be considered an error. Therefore, we suggest evaluating using both NED and attachment score in order to get a full picture of the parser's performance.

A possible criticism on NED is that it is only indifferent to alternative annotations in structures of size 2 (e.g., "to eat") and does not necessarily handle larger problematic structures, such as coordinations

[9] Otherwise, the gold parse would have contained a $w_1 \rightarrow w_2 \rightarrow w_3 \rightarrow w_1$ cycle.

[10] An implementation of NED is available at http://www.cs.huji.ac.il/∼roys02/software/ned.html

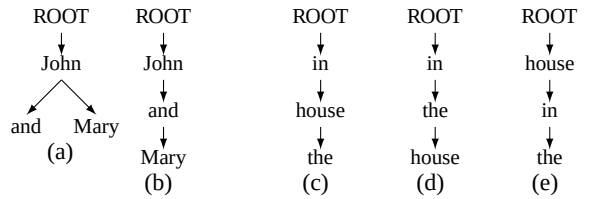


Figure 4: Alternative parses of "John and Mary" and "in the house". Figure 4(a) follows (Collins, 1999), Figure 4(b) follows (Johansson and Nugues, 2007). Figure 4(c) follows (Collins, 1999; Yamada and Matsumoto, 2003). Figure 4(d) and Figure 4(e) show induced parses made by (*km04*,*saj10a*) and *cs09*, respectively.

(see Section 4). For example, Figure 4(a) and Figure 4(b) present two alternative annotations of the sentence "John and Mary". Assume the parse in Figure 4(a) is the gold parse and that in Figure 4(b) is the induced parse. The word "Mary" is a NED error, since its induced parent ("and") is neither its gold child nor its gold grandparent. Thus, NED does not accept all possible annotations of structures of size 3. On the other hand, using a method which accepts all possible annotations of structures of size 3 seems too permissive. A better solution may be to modify the gold standard annotation, so to explicitly annotate problematic structures as such. We defer this line of research to future work.

NED is therefore an evaluation measure which is indifferent to *edge-flips*, and is consequently less sensitive to alternative annotations. We now show that NED is indifferent to the differences between the structures originally learned by the algorithms mentioned in Section 3 and the gold standard annotation in all the problematic cases we consider.

Most of the modifications made are *edge-flips*, and are therefore ignored by NED. The exceptions are coordinations and prepositional phrases which are structures of size 3. In the former, the alternative annotations differ only in a single *edge-flip* (i.e., $CC \rightarrow NNP$), and are thus not NED errors. Regarding prepositional phrases, Figure 4(c) presents the gold standard of "in the house", Figure 4(d) the parse induced by *km04* and *saj10a* and Figure 4(e) the parse induced by *cs09*. As the reader can verify, both induced parses receive a perfect NED score.

In order to further demonstrate NED's insensitivity to alternative annotations, we took two of the three common gold standard annotations (see Section 4) and evaluated them one against the other. We considered section 23 of WSJ following the scheme of (Yamada and Matsumoto, 2003) as the gold standard and of (Collins, 1999) as the evaluated set. Results show that the attachment score is only 85.6%, the undirected accuracy is improved to 90.3%, while the NED score is 95.3%. This shows that NED is significantly less sensitive to the differences between the different annotation schemes, compared to the other evaluation measures.

6 Experimenting with NED

In this section we show that NED indeed reduces the performance difference between the original and the modified parameter sets, thus providing empirical evidence for its validity. For brevity, we present results only for the entire WSJ corpus. Results on WSJ10 are similar. The datasets and decoding algorithms are the same as those used in Section 3.

Table 3 shows the score differences between the parameter sets using attachment score, undirected evaluation and NED. A substantial difference persists under undirected evaluation: a gap of 7.7% in *cs09*, of 3.5% in *saj10a* and of 1.3% in *km04*.

The differences are further reduced using NED. This is consistent with our discussion in Section 5, and shows that undirected evaluation only ignores some of the errors inflicted by *edge-flips*.

For *cs09*, the difference is substantially reduced, but a 4.2% performance gap remains. For *km04* and *saj10a*, the original parameters outperform the new ones by 3.6% and 1% respectively.

We can see that even when ignoring *edge-flips*, some difference remains, albeit not necessarily in the favor of the modified models. This is because we did not directly perform *edge-flips*, but rather *parameter-flips*. The difference is thus a result of second-order effects stemming from the *parameter-flips*. In the next section, we explain why the remaining difference is positive for some algorithms (*cs09*) and negative for others (*km04, saj10a*).

For completeness, Table 4 shows a comparison of some of the current state-of-the-art algorithms, using attachment score, undirected evaluation and NED. The training and test sets are those used in Section 3. The table shows that the relative orderings of the algorithms under NED is different than under the other

Algo.	$Mod. - Orig.$		
	Attach.	Undir.	NED
km04	9.3 (43.9–34.6)	1.3 (54.2–52.9)	–3.6 (63–66.6)
cs09	14.7 (54.6–39.9)	7.7 (56.9–49.2)	4.2 (66.8–62.6)
saj10a	12.7 (54.3–41.6)	3.5 (59.4–55.9)	–1 (66.8–67.8)

Table 3: Differences between the modified and original parameter sets when evaluated using attachment score (*Attach.*), undirected evaluation (*Undir.*), and NED.

measures. This is an indication that NED provides a different perspective on algorithm quality[11].

Algo.	Att_{10}	Att_{∞}	Un_{10}	Un_{∞}	NED_{10}	NED_{∞}
bbdk10	66.1	49.6	70.1	56.0	75.5	61.8
bc10	**67.2**	**53.6**	**73**	**61.7**	81.6	70.2
cs09	61.5	42	66.9	50.4	81.5	62.9
gggtp10	57.1	45	62.5	53.2	80.4	65.1
km04	45.8	34.6	60.3	52.9	78.4	66.6
saj10a	54.7	41.6	66.5	55.9	78.9	67.8
saj10c	63.8	46.1	72.6	58.8	**84.2**	**70.8**
saj10b*	67.9	48.2	74.0	57.7	86.0	70.7

Table 4: A comparison of recent works, using Att (*attachment score*) Un (*undirected evaluation*) and NED, on sentences of length $\leq$ 10 (excluding punctuation) and on all sentences. The gold standard is obtained using the rules of (Yamada and Matsumoto, 2003). *bbdk10*: (Berg-Kirkpatrick et al., 2010), *bc10*: (Blunsom and Cohn, 2010), *cs09*: (Cohen and Smith, 2009), *gggtp10*: (Gillenwater et al., 2010), *km04*: A replication of (Klein and Manning, 2004), *saj10a*: (Spitkovsky et al., 2010a), *saj10c*: (Spitkovsky et al., 2010c), *saj10b**: A lightly-supervised algorithm (Spitkovsky et al., 2010b).

7 Discussion

In this paper we explored two ways of dealing with cases in which there is no clear theoretical justification to prefer one dependency structure over another. Our experiments suggest that it is crucial to deal with such structures if we would like to have a proper evaluation of unsupervised parsing algorithms against a gold standard.

The first way was to modify the parameters of the parsing algorithms so that in cases where such problematic decisions are to be made they follow the gold standard annotation. Indeed, this modification leads to a substantial improvement in the attachment score of the algorithms.

[11]Results may be different than the ones published in the original papers due to the different conversion procedures used in each work. See Section 4 for discussion.

The second way was to change the evaluation. The NED measure we proposed does not punish for differences between gold and induced structures in the problematic cases. Indeed, in Section 6 (Table 3) we show that the differences between the original and modified models are much smaller when evaluating with NED compared to when evaluating with the traditional attachment score.

As Table 3 reveals, however, even when evaluating with NED, there is still some difference between the original and the modified model, for each of the algorithms we consider. Moreover, for two of the algorithms (*km04* and *saj10a*) NED prefers the original model while for one (*cs09*) it prefers the modified version. In this section we explain these patterns and show that they are both consistent and predictable.

Our hypothesis, for which we provide empirical justification, is that in cases where there is no theoretically preferred annotation, NED prefers the structures that are more learnable by DMV. That is, NED gives higher scores to the annotations that better fit the assumptions and modeling decisions of DMV, the model that lies in the basis of the parsing algorithms.

To support our hypothesis we perform an experiment requiring two preparatory steps for each algorithm. First, we construct a supervised version of the algorithm. This supervised version consists of the same statistical model as the original unsupervised algorithm, but the parameters are estimated to maximize the likelihood of a *syntactically annotated* training corpus, rather than of a plain text corpus.

Second, we construct two corpora for the algorithm, both consist of the same text and differ only in their syntactic annotation. The first is annotated with the gold standard annotation. The second is similarly annotated except in the linguistically problematic structures. We replace these structures with the ones that would have been created with the unsupervised version of the algorithm (see Table 1 for the relevant structures for each algorithm)[12]. Each

[12]In cases the structures are comprised of a single edge, the second corpus is obtained from the gold standard by an *edge-flip*. The only exceptions are the cases of the prepositional phrases. Their gold standard and the learned structures for each of the algorithms are shown in Figure 4. In this case, the second corpus is obtained from the gold standard by replacing each prepositional phrase in the gold standard with the corresponding

corpus is divided into a training and a test set.

We then train the supervised version of the algorithms on each of the training sets. We parse the test data twice, once with each of the resulting models. We evaluate both parsed corpora against the corpus annotation from which they originated.

The training set of each corpus consists of sections 2–21 of WSJ20 (i.e., WSJ sentences of length $\leq$20, excluding punctuation)[13] and the test set is section 23 of WSJ$^{\infty}$. Evaluation is performed using both NED and attachment score. The patterns we observed are very similar for both. For brevity, we report only attachment score results.

	km04		*cs09*		*saj10a*	
	Orig.	*Gold*	*Orig.*	*Gold*	*Orig.*	*Gold*
NED, Unsup.	**66.6**	63	62.6	**66.8**	**67.8**	66.8
Sup.	**71.3**	69.9	63.3	**69.9**	**71.8**	69.9

Table 5: The first line shows the NED results from Section 6, when using the original parameters (*Orig.* columns) and the modified parameters (*Gold* columns). The second line shows the results of the supervised versions of the algorithms using the corpus which agrees with the unsupervised model in the problematic cases (*Orig.*) and the gold standard (*Gold*).

The results of our experiment are presented in Table 5 along with a comparison to the NED scores from Section 6. The table clearly demonstrates that a set of parameters (original or modified) is preferred by NED in the unsupervised experiments reported in Section 6 (top line) if and only if the structures produced by this set are better learned by the supervised version of the algorithm (bottom line).

This observation supports our hypothesis that in cases where there is no theoretical preference for one structure over the other, NED (unlike the other measures) prefers the structures that are more consistent with the modeling assumptions lying in the basis of the algorithm. We consider this to be a desired property of a measure since a more consistent model should be preferred where no theoretical preference exists.

learned structure.

[13] In using WSJ20, we follow (Spitkovsky et al., 2010a), which showed that training the DMV on sentences of bounded length yields a higher score than using the entire corpus. We use it as we aim to use an optimal setting.

8 Conclusion

In this paper we showed that the standard evaluation of unsupervised dependency parsers is highly sensitive to problematic annotations. We modified a small set of parameters that controls the annotation in such problematic cases in three leading parsers. This resulted in a major performance boost, which is unindicative of a true difference in quality.

We presented *Neutral Edge Direction* (NED), a measure that is less sensitive to the annotation of local structures. As the problematic structures are generally local, NED is less sensitive to their alternative annotations. In the future, we suggest reporting NED along with the current measures.

Acknowledgements. We would like to thank Shay Cohen for his assistance with his implementation of the DMV parser and Taylor Berg-Kirkpatrick, Phil Blunsom and Jennifer Gillenwater for providing us with their data sets. We would also like to thank Valentin I. Spitkovsky for his comments and for providing us with his data sets.

References

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero and Dan Klein, 2010. *Painless unsupervised learning with features*. In *Proc. of NAACL*.

Taylor Berg-Kirkpatrick and Dan Klein, 2010. *Phylogenetic Grammar Induction*. In *Proc. of ACL*.

Cristina Bosco and Vincenzo Lombardo, 2004. *Dependency and relational structure in treebank annotation*. In *Proc. of the Workshop on Recent Advances in Dependency Grammar at COLING'04*.

Phil Blunsom and Trevor Cohn, 2010. *Unsupervised Induction of Tree Substitution Grammars for Dependency Parsing*. In *Proc. of EMNLP*.

Shay B. Cohen, Kevin Gimpel and Noah A. Smith, 2008. *Logistic Normal Priors for Unsupervised Probabilistic Grammar Induction*. In *Proc. of NIPS*.

Shay B. Cohen and Noah A. Smith, 2009. *Shared Logistic Normal Distributions for Soft Parameter Tying*. In *Proc. of HLT-NAACL*.

Michael J. Collins, 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia.

Alexander Clark, 2001. *Unsupervised language acquisition: theory and practice*. Ph.D. thesis, University of Sussex.

Hal Daumé III, 2009. *Unsupervised search-based structured prediction*. In *Proc. of ICML*.

Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, João V. Graça and Fernando Pereira, 2007. *Frustratingly Hard Domain Adaptation for Dependency Parsing*. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

Gregory Druck, Gideon Mann and Andrew McCallum, 2009. *Semi-supervised learning of dependency parsers using generalized expectation criteria*. In *Proc. of ACL*.

Jennifer Gillenwater, Kuzman Ganchev, João V. Graça, Ben Taskar and Fernando Preira, 2010. *Sparsity in dependency grammar induction*. In *Proc. of ACL*.

William P. Headden III, David McClosky, and Eugene Charniak, 2008. *Evaluating Unsupervised Part-of-Speech Tagging for Grammar Induction*. In *Proc. of COLING*.

William P. Headden III, Mark Johnson and David McClosky, 2009. *Improving unsupervised dependency parsing with richer contexts and smoothing*. In *Proc. of HLT-NAACL*.

Richard Johansson and Pierre Nugues, 2007. *Extended Constituent-to-Dependency Conversion for English*. In *Proc. of NODALIDA*.

Dan Klein, 2005. *The unsupervised learning of natural language structure*. Ph.D. thesis, Stanford University.

Dan Klein and Christopher Manning, 2004. *Corpus-based induction of syntactic structure: Models of dependency and constituency*. In *Proc. of ACL*.

Sandra Kübler, 2005. *How Do Treebank Annotation Schemes Influence Parsing Results? Or How Not to Compare Apples And Oranges*. In *Proc. of RANLP*.

Sandra Kübler, R. McDonald and Joakim Nivre, 2009. *Dependency Parsing*. Morgan And Claypool Publishers.

Mitchell Marcus, Beatrice Santorini and Mary Ann Marcinkiewicz, 1993. *Building a large annotated corpus of English: The Penn treebank. Computational Linguistics* 19:313-330.

Tahira Naseem, Harr Chen, Regina Barzilay and Mark Johnson, 2010. *Using universal linguistic knowledge to guide grammar induction*. In *Proc. of EMNLP*.

Joakim Nivre, 2006. *Inductive Dependency Parsing*. Springer.

Joakim Nivre, Johan Hall and Jens Nilsson, 2006. *MaltParser: A data-driven parser-generator for dependency parsing*. In *Proc. of LREC-2006*.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel and Deniz Yuret, 2007. *The CoNLL 2007 shared task on dependency parsing*. In *Proc. of the CoNLL Shared Task, EMNLP-CoNLL, 2007*.

Jens Nilsson, Joakim Nivre and Johan Hall, 2006. *Graph transformations in data-driven dependency parsing*. In *Proc. of ACL*.

Owen Rambow, Cassandre Creswell, Rachel Szekely, Harriet Tauber and Marilyn Walker, 2002. *A dependency treebank for English*. In *Proc. of LREC*.

Noah A. Smith and Jason Eisner, 2005. *Guiding unsupervised grammar induction using contrastive estimation*. In *Proc. of IJCAI*.

Noah A. Smith and Jason Eisner, 2006. *Annealing structural bias in multilingual weighted grammar induction*. In *Proc. of ACL*.

Valentin I. Spitkovsky, Hiyan Alshawi and Daniel Jurafsky, 2010a. *From Baby Steps to Leapfrog: How "Less is More" in Unsupervised Dependency Parsing*. In *Proc. of NAACL-HLT*.

Valentin I. Spitkovsky, Hiyan Alshawi and Daniel Jurafsky, 2010b. *Profiting from Mark-Up: Hyper-Text Annotations for Guided Parsing*. In *Proc. of ACL*.

Valentin I. Spitkovsky, Hiyan Alshawi, Daniel Jurafsky and Christopher D. Manning, 2010c. *Viterbi training improves unsupervised dependency parsing*. In *Proc. of CoNLL*.

Qin Iris Wang, Dale Schuurmans and Dekang Lin, 2005. *Strictly Lexical Dependency Parsing*. In *IWPT*.

Qin Iris Wang, Colin Cherry, Dan Lizotte and Dale Schuurmans, 2006. *Improved Large Margin Dependency Parsing via Local Constraints and Laplacian Regularization*. In *Proc. of CoNLL*.

Hiroyasu Yamada and Yuji Matsumoto, 2003. *Statistical dependency analysis with support vector machines*. In *Proc. of the International Workshop on Parsing Technologies*.

Dynamic Programming Algorithms for Transition-Based Dependency Parsers

Marco Kuhlmann
Dept. of Linguistics and Philology
Uppsala University, Sweden
marco.kuhlmann@lingfil.uu.se

Carlos Gómez-Rodríguez
Departamento de Computación
Universidade da Coruña, Spain
cgomezr@udc.es

Giorgio Satta
Dept. of Information Engineering
University of Padua, Italy
satta@dei.unipd.it

Abstract

We develop a general dynamic programming technique for the tabulation of transition-based dependency parsers, and apply it to obtain novel, polynomial-time algorithms for parsing with the arc-standard and arc-eager models. We also show how to reverse our technique to obtain new transition-based dependency parsers from existing tabular methods. Additionally, we provide a detailed discussion of the conditions under which the feature models commonly used in transition-based parsing can be integrated into our algorithms.

1 Introduction

Dynamic programming algorithms, also known as tabular or chart-based algorithms, are at the core of many applications in natural language processing. When applied to formalisms such as context-free grammar, they provide polynomial-time parsing algorithms and polynomial-space representations of the resulting parse forests, even in cases where the size of the search space is exponential in the length of the input string. In combination with appropriate semirings, these packed representations can be exploited to compute many values of interest for machine learning, such as best parses and feature expectations (Goodman, 1999; Li and Eisner, 2009).

In this paper, we follow the line of investigation started by Huang and Sagae (2010) and apply dynamic programming to (projective) transition-based dependency parsing (Nivre, 2008). The basic idea, originally developed in the context of push-down automata (Lang, 1974; Tomita, 1986; Billot and Lang, 1989), is that while the number of computations of a transition-based parser may be exponential in the length of the input string, several portions of these computations, when appropriately represented, can be shared. This can be effectively implemented through dynamic programming, resulting in a packed representation of the set of all computations.

The contributions of this paper can be summarized as follows. We provide (declarative specifications of) novel, polynomial-time algorithms for two widely-used transition-based parsing models: arc-standard (Nivre, 2004; Huang and Sagae, 2010) and arc-eager (Nivre, 2003; Zhang and Clark, 2008). Our algorithm for the arc-eager model is the first tabular algorithm for this model that runs in polynomial time. Both algorithms are derived using the same general technique; in fact, we show that this technique is applicable to all transition-parsing models whose transitions can be classified into "shift" and "reduce" transitions. We also show how to reverse the tabulation to derive a new transition system from an existing tabular algorithm for dependency parsing, originally developed by Gómez-Rodríguez et al. (2008). Finally, we discuss in detail the role of feature information in our algorithms, and in particular the conditions under which the feature models traditionally used in transition-based dependency parsing can be integrated into our framework.

While our general approach is the same as the one of Huang and Sagae (2010), we depart from their framework by not representing the computations of a parser as a graph-structured stack in the sense of Tomita (1986). We instead simulate computations as in Lang (1974), which results in simpler algorithm specifications, and also reveals deep similarities between transition-based systems for dependency parsing and existing tabular methods for lexicalized context-free grammars.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 673–682,
Portland, Oregon, June 19-24, 2011.

2 Transition-Based Dependency Parsing

We start by briefly introducing the framework of transition-based dependency parsing; for details, we refer to Nivre (2008).

2.1 Dependency Graphs

Let $w = w_0 \cdots w_{n-1}$ be a string over some fixed alphabet, where $n \geq 1$ and w_0 is the special token ROOT. A *dependency graph* for w is a directed graph $G = (V_w, A)$, where $V_w = \{0, \ldots, n-1\}$ is the set of nodes, and $A \subseteq V_w \times V_w$ is the set of arcs. Each node in V_w encodes the position of a token in w, and each arc in A encodes a dependency relation between two tokens. To denote an arc $(i, j) \in A$, we write $i \rightarrow j$; here, the node i is the head, and the node j is the dependent. A sample dependency graph is given in the left part of Figure 2.

2.2 Transition Systems

A *transition system* is a structure $S = (C, T, I, C_t)$, where C is a set of *configurations*, T is a finite set of *transitions*, which are partial functions $t : C \rightharpoonup C$, I is a total *initialization function* mapping each input string to a unique initial configuration, and $C_t \subseteq C$ is a set of *terminal configurations*.

The transition systems that we investigate in this paper differ from each other only with respect to their sets of transitions, and are identical in all other aspects. In each of them, a configuration is defined relative to a string w as above, and is a triple $c = (\sigma, \beta, A)$, where σ and β are disjoint lists of nodes from V_w, called *stack* and *buffer*, respectively, and $A \subseteq V_w \times V_w$ is a set of arcs. We denote the stack, buffer and arc set associated with c by $\sigma(c)$, $\beta(c)$, and $A(c)$, respectively. We follow a standard convention and write the stack with its topmost element to the right, and the buffer with its first element to the left; furthermore, we indicate concatenation in the stack and in the buffer by a vertical bar. The initialization function maps each string w to the initial configuration $([], [0, \ldots, |w|-1], \emptyset)$. The set of terminal configurations contains all configurations of the form $([0], [], A)$, where A is some set of arcs.

Given an input string w, a parser based on S processes w from left to right, starting in the initial configuration $I(w)$. At each point, it applies one of the transitions, until at the end it reaches a terminal configuration; the dependency graph defined by the arc set associated with that configuration is then returned as the analysis for w. Formally, a *computation* of S on w is a sequence $\gamma = c_0, \ldots, c_m$, $m \geq 0$, of configurations (defined relative to w) in which each configuration is obtained as the value of the preceding one under some transition. It is called *complete* whenever $c_0 = I(w)$, and $c_m \in C_t$. We note that a computation can be uniquely specified by its initial configuration c_0 and the sequence of its transitions, understood as a string over T. Complete computations, where c_0 is fixed, can be specified by their transition sequences alone.

$$
\begin{aligned}
(\sigma, i|\beta, A) &\vdash (\sigma|i, \beta, A) && \text{(sh)} \\
(\sigma|i|j, \beta, A) &\vdash (\sigma|j, \beta, A \cup \{j \rightarrow i\}) && \text{(la)} \\
(\sigma|i|j, \beta, A) &\vdash (\sigma|i, \beta, A \cup \{i \rightarrow j\}) && \text{(ra)}
\end{aligned}
$$

Figure 1: Transitions in the arc-standard model.

3 Arc-Standard Model

To introduce the core concepts of the paper, we first look at a particularly simple model for transition-based dependency parsing, known as the *arc-standard model*. This model has been used, in slightly different variants, by a number of parsers (Nivre, 2004; Attardi, 2006; Huang and Sagae, 2010).

3.1 Transition System

The arc-standard model uses three types of transitions: SHIFT (sh) removes the first node in the buffer and pushes it to the stack. LEFT-ARC (la) creates a new arc with the topmost node on the stack as the head and the second-topmost node as the dependent, and removes the second-topmost node from the stack. RIGHT-ARC (ra) is symmetric to LEFT-ARC in that it creates an arc with the second-topmost node as the head and the topmost node as the dependent, and removes the topmost node.

The three transitions can be formally specified as in Figure 1. The right half of Figure 2 shows a complete computation of the arc-standard transition system, specified by its transition sequence. The picture also shows the contents of the stack over the course of the computation; more specifically, column i shows the stack $\sigma(c_i)$ associated with the configuration c_i.

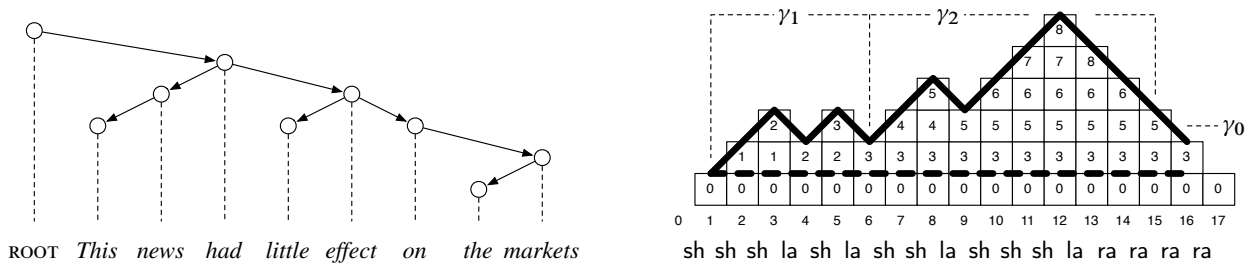


Figure 2: A dependency tree (left) and a computation generating this tree in the arc-standard system (right).

3.2 Push Computations

The key to the tabulation of transition-based dependency parsers is to find a way to decompose computations into smaller, shareable parts. For the arc-standard model, as well as for the other transition systems that we consider in this paper, we base our decomposition on the concept of *push computations*. By this, we mean computations

$$\gamma = c_0, \ldots, c_m, \quad m \geq 1,$$

on some input string w with the following properties:

(P1) The initial stack $\sigma(c_0)$ is not modified during the computation, and is not even exposed after the first transition: For every $1 \leq i \leq m$, there exists a non-empty stack σ_i such that $\sigma(c_i) = \sigma(c_0)|\sigma_i$.

(P2) The overall effect of the computation is to push a single node to the stack: The stack $\sigma(c_m)$ can be written as $\sigma(c_m) = \sigma(c_0)|h$, for some $h \in V_w$.

We can verify that the computation in Figure 2 is a push computation. We can also see that it contains shorter computations that are push computations; one example is the computation $\gamma_0 = c_1, \ldots, c_{16}$, whose overall effect is to push the node 3. In Figure 2, this computation is marked by the zig-zag path traced in bold. The dashed line delineates the stack $\sigma(c_1)$, which is not modified during γ_0.

Every computation that consists of a single sh transition is a push computation. Starting from these atoms, we can build larger push computations by means of two (partial) binary operations f_{la} and f_{ra}, defined as follows. Let $\gamma_1 = c_{10}, \ldots, c_{1m_1}$ and $\gamma_2 = c_{20}, \ldots, c_{2m_2}$ be push computations on the same input string w such that $c_{1m_1} = c_{20}$. Then

$$f_{\mathsf{ra}}(\gamma_1, \gamma_2) = c_{10}, \ldots, c_{1m_1}, c_{21}, \ldots, c_{2m_2}, c,$$

where c is obtained from c_{2m_2} by applying the ra transition. (The operation f_{la} is defined analogously.) We can verify that $f_{\mathsf{ra}}(\gamma_1, \gamma_2)$ is another push computation. For instance, with respect to Figure 2, $f_{\mathsf{ra}}(\gamma_1, \gamma_2) = \gamma_0$. Conversely, we say that the push computation γ_0 can be *decomposed* into the subcomputations γ_1 and γ_2, and the operation f_{ra}.

3.3 Deduction System

Building on the compositional structure of push computations, we now construct a deduction system (in the sense of Shieber et al. (1995)) that tabulates the computations of the arc-standard model for a given input string $w = w_0 \cdots w_{n-1}$. For $0 \leq i \leq n$, we shall write β_i to denote the buffer $[i, \ldots, n-1]$. Thus, β_0 denotes the full buffer, associated with the initial configuration $I(w)$, and β_n denotes the empty buffer, associated with a terminal configuration $c \in C_t$.

Item form. The items of our deduction system take the form $[i, h, j]$, where $0 \leq i \leq h < j \leq n$. The intended interpretation of an item $[i, h, j]$ is: For every configuration c_0 with $\beta(c_0) = \beta_i$, there exists a push computation $\gamma = c_0, \ldots, c_m$ such that $\beta(c_m) = \beta_j$, and $\sigma(c_m) = \sigma(c_0)|h$.

Goal. The only goal item is $[0, 0, n]$, asserting that there exists a complete computation for w.

Axioms. For every stack σ, position $i < n$ and arc set A, by a single sh transition we obtain the push computation $(\sigma, \beta_i, A), (\sigma|i, \beta_{i+1}, A)$. Therefore we can take the set of all items of the form $[i, i, i+1]$ as the axioms of our system.

Inference rules. The inference rules parallel the composition operations f_{la} and f_{ra}. Suppose that we have deduced the items $[i, h_1, k]$ and $[k, h_2, j]$, where $0 \leq i \leq h_1 < k \leq h_2 < j \leq n$. The item $[i, h_1, k]$ asserts that for every configuration c_{10}

$$\textbf{Item form: } [i,h,j]\,,\ 0 \le i \le h < j \le |w| \qquad \textbf{Goal: } [0,0,|w|] \qquad \textbf{Axioms: } [i,i,i+1]$$

$$\textbf{Inference rules: } \frac{[i,h_1,k] \quad [k,h_2,j]}{[i,h_2,j]}\ (\mathsf{la}; h_2 \to h_1) \qquad \frac{[i,h_1,k] \quad [k,h_2,j]}{[i,h_1,j]}\ (\mathsf{ra}; h_1 \to h_2)$$

Figure 3: Deduction system for the arc-standard model.

with $\beta(c_{10}) = \beta_i$, there exists a push computation $\gamma_1 = c_{10}, \ldots, c_{1m_1}$ such that $\beta(c_{1m_1}) = \beta_k$, and $\sigma(c_{1m_1}) = \sigma(c_{10})|h_1$. Using the item $[k, h_2, j]$, we deduce the existence of a second push computation $\gamma_2 = c_{20}, \ldots, c_{2m_2}$ such that $c_{20} = c_{1m_1}$, $\beta(c_{2m_2}) = \beta_j$, and $\sigma(c_{2m_2}) = \sigma(c_{10})|h_1|h_2$. By means of f_{ra}, we can then compose γ_1 and γ_2 into a new push computation

$$f_{\mathsf{ra}}(\gamma_1, \gamma_2) \;=\; c_{10}, \ldots, c_{1m_1}, c_{21}, \ldots, c_{2m_2}, c\,.$$

Here, $\beta(c) = \beta_j$, and $\sigma(c) = \sigma(c_{10})|h_1$. Therefore, we may generate the item $[i, h_1, j]$. The inference rule for la can be derived analogously.

Figure 3 shows the complete deduction system.

3.4 Completeness and Non-Ambiguity

We have informally argued that our deduction system is sound. To show completeness, we prove the following lemma: For all $0 \le i \le h < j \le |w|$ and every push computation $\gamma = c_0, \ldots, c_m$ on w with $\beta(c_0) = \beta_i$, $\beta(c_m) = \beta_j$ and $\sigma(c_m) = \sigma(c_0)|h$, the item $[i, h, j]$ is generated. The proof is by induction on m, and there are two cases:

$m = 1$. In this case, γ consists of a single sh transition, $h = i$, $j = i + 1$, and we need to show that the item $[i, i, i + 1]$ is generated. This holds because this item is an axiom.

$m \ge 2$. In this case, γ ends with either a la or a ra transition. Let c be the rightmost configuration in γ that is different from c_m and whose stack size is one larger than the size of $\sigma(c_0)$. The computations

$$\gamma_1 = c_0, \ldots, c \quad \text{and} \quad \gamma_2 = c, \ldots, c_{m-1}$$

are both push computations with strictly fewer transitions than γ. Suppose that the last transition in γ is ra. In this case, $\beta(c) = \beta_k$ for some $i < k < j$, $\sigma(c) = \sigma(c_0)|h$ with $h < k$, $\beta(c_{m-1}) = \beta_j$, and $\sigma(c_{m-1}) = \sigma(c_0)|h|h'$ for some $k \le h' < j$. By induction, we may assume that we have generated items $[i, h, k]$ and $[k, h', j]$. Applying the inference rule for ra, we deduce the item $[i, h, j]$. An analogous argument can be made for f_{la}.

Apart from being sound and complete, our deduction system also has the property that it assigns at most one derivation to a given item. To see this, note that in the proof of the lemma, the choice of c is uniquely determined: If we take any other configuration c' that meets the selection criteria, then the computation $\gamma_2' = c', \ldots, c_{m-1}$ is not a push computation, as it contains c as an intermediate configuration, and thereby violates property P1.

3.5 Discussion

Let us briefly take stock of what we have achieved so far. We have provided a deduction system capable of tabulating the set of all computations of an arc-standard parser on a given input string, and proved the correctness of this system relative to an interpretation based on push computations. Inspecting the system, we can see that its generic implementation takes space in $\mathcal{O}(|w|^3)$ and time in $\mathcal{O}(|w|^5)$.

Our deduction system is essentially the same as the one for the CKY algorithm for bilexicalized context-free grammar (Collins, 1996; Gómez-Rodríguez et al., 2008). This equivalence reveals a deep correspondence between the arc-standard model and bilexicalized context-free grammar, and, via results by Eisner and Satta (1999), to head automata. In particular, Eisner's and Satta's "hook trick" can be applied to our tabulation to reduce its runtime to $\mathcal{O}(|w|^4)$.

4 Adding Features

The main goal with the tabulation of transition-based dependency parsers is to obtain a representation based on which semiring values such as the highest-scoring computation for a given input (and with it, a dependency tree) can be calculated. Such computations involve the use of feature information. In this section, we discuss how our tabulation of the arc-standard system can be extended for this purpose.

$$\frac{[i, h_1, k; \langle x_2, x_1\rangle, \langle x_1, x_3\rangle] : v_1 \quad [k, h_2, j; \langle x_1, x_3\rangle, \langle x_3, x_4\rangle] : v_2}{[i, h_1, j; \langle x_2, x_1\rangle, \langle x_1, x_3\rangle] : v_1 + v_2 + \langle x_3, x_4\rangle \cdot \vec{\alpha}_{\mathsf{ra}}} \ (\mathsf{ra})$$

$$\frac{[i, h, j; \langle x_2, x_1\rangle, \langle x_1, x_3\rangle] : v}{[j, j, j+1; \langle x_1, x_3\rangle, \langle x_3, w_j\rangle] : \langle x_1, x_3\rangle \cdot \vec{\alpha}_{\mathsf{sh}}} \ (\mathsf{sh})$$

Figure 4: Extended inference rules under the feature model $\Phi = \langle \mathsf{s}_1.\mathsf{w}, \mathsf{s}_0.\mathsf{w}\rangle$. The annotations indicate how to calculate a candidate for an update of the Viterbi score of the conclusion using the Viterbi scores of the premises.

4.1 Scoring Computations

For the sake of concreteness, suppose that we want to score computations based on the following model, taken from Zhang and Clark (2008). The score of a computation γ is broken down into a sum of scores $score(t, c_t)$ for combinations of a transition t in the transition sequence associated with γ and the configuration c_t in which t was taken:

$$score(\gamma) \ = \ \sum_{t \in \gamma} score(t, c_t) \qquad (1)$$

The score $score(t, c_t)$ is defined as the dot product of the feature representation of c_t relative to a *feature model* Φ and a transition-specific weight vector $\vec{\alpha}_t$:

$$score(t, c_t) \ = \ \Phi(c_t) \cdot \vec{\alpha}_t$$

The feature model Φ is a vector $\langle \phi_1, \ldots, \phi_n\rangle$ of elementary *feature functions*, and the feature representation $\Phi(c)$ of a configuration c is a vector $\vec{x} = \langle \phi_1(c), \ldots, \phi_n(c)\rangle$ of atomic values. Two examples of feature functions are the word form associated with the topmost and second-topmost node on the stack; adopting the notation of Huang and Sagae (2010), we will write these functions as $\mathsf{s}_0.\mathsf{w}$ and $\mathsf{s}_1.\mathsf{w}$, respectively. Feature functions like these have been used in several parsers (Nivre, 2006; Zhang and Clark, 2008; Huang et al., 2009).

4.2 Integration of Feature Models

To integrate feature models into our tabulation of the arc-standard system, we can use extended items of the form $[i, h, j; \vec{x}_L, \vec{x}_R]$ with the same intended interpretation as the old items $[i, h, j]$, except that the initial configuration of the asserted computations $\gamma = c_0, \ldots, c_m$ now is required to have the feature representation $\vec{x}_L$, and the final configuration is required to have the representation $\vec{x}_R$:

$$\Phi(c_0) = \vec{x}_L \quad \text{and} \quad \Phi(c_m) = \vec{x}_R$$

We shall refer to the vectors $\vec{x}_L$ and $\vec{x}_R$ as the *left-context vector* and the *right-context vector* of the computation γ, respectively.

We now need to change the deduction rules so that they become faithful to the extended interpretation. Intuitively speaking, we must ensure that the feature values can be computed along the inference rules. As a concrete example, consider the feature model $\Phi = \langle \mathsf{s}_1.\mathsf{w}, \mathsf{s}_0.\mathsf{w}\rangle$. In order to integrate this model into our tabulation, we change the rule for ra as in Figure 4, where $x_1, \ldots, x_4$ range over possible word forms. The shared variable occurrences in this rule capture the constraints that hold between the feature values of the subcomputations γ_1 and γ_2 asserted by the premises, and the computations $f_{\mathsf{ra}}(\gamma_1, \gamma_2)$ asserted by the conclusion. To illustrate this, suppose that γ_1 and γ_2 are as in Figure 2. Then the three occurrences of x_3 for instance encode that

$$[\mathsf{s}_0.\mathsf{w}](c_6) = [\mathsf{s}_1.\mathsf{w}](c_{15}) = [\mathsf{s}_0.\mathsf{w}](c_{16}) = w_3 \,.$$

We also need to extend the axioms, which correspond to computations consisting of a single sh transition. The most conservative way to do this is to use a generate-and-test technique: Extend the existing axioms by all valid choices of left-context and right-context vectors, that is, by all pairs $\vec{x}_L, \vec{x}_R$ such that there exists a configuration c with $\Phi(c) = \vec{x}_L$ and $\Phi(\mathsf{sh}(c)) = \vec{x}_R$. The task of filtering out useless guesses can then be delegated to the deduction system.

A more efficient way is to only have one axiom, for the case where $c = I(w)$, and to add to the deduction system a new, unary inference rule for sh as in Figure 4. This rule only creates items whose left-context vector is the right-context vector of some other item, which prevents the generation of useless items. In the following, we take this second approach, which is also the approach of Huang and Sagae (2010).

$$\frac{[i, h, j; \langle x_2, x_1\rangle, \langle x_1, x_3\rangle] : (p, v)}{[j, j, j+1; \langle x_1, x_3\rangle, \langle x_3, w_j\rangle] : (p+\sigma, \sigma)}\ (\mathsf{sh})\,,\ \text{where } \sigma = \langle x_1, x_3\rangle \cdot \vec{\alpha}_{\mathsf{sh}}$$

$$\frac{[i, h_1, k; \langle x_2, x_1\rangle, \langle x_1, x_3\rangle] : (p_1, v_1) \quad [k, h_2, j; \langle x_1, x_3\rangle, \langle x_3, x_4\rangle] : (p_2, v_2)}{[i, h_1, j; \langle x_2, x_1\rangle, \langle x_1, x_3\rangle] : (p_1 + v_2 + \rho, v_1 + v_2 + \rho)}\ (\mathsf{ra})\,,\ \text{where } \rho = \langle x_3, x_4\rangle \cdot \vec{\alpha}_{\mathsf{ra}}$$

Figure 5: Extended inference rules under the feature model $\Phi = \langle \mathsf{s}_0.\mathsf{w}, \mathsf{s}_1.\mathsf{w}\rangle$. The annotations indicate how to calculate a candidate for an update of the prefix score and Viterbi score of the conclusion.

4.3 Computing Viterbi Scores

Once we have extended our deduction system with feature information, many values of interest can be computed. One simple example is the *Viterbi score* for an input w, defined as

$$\arg\max_{\gamma \in \Gamma(w)} \mathit{score}(\gamma)\,, \tag{2}$$

where $\Gamma(w)$ denotes the set of all complete computations for w. The score of a complex computation $f_t(\gamma_1, \gamma_2)$ is the sum of the scores of its subcomputations γ_1, γ_2, plus the transition-specific dot product. Since this dot product only depends on the feature representation of the final configuration of γ_2, the Viterbi score can be computed on top of the inference rules using standard techniques. The crucial calculation is indicated in Figure 4.

4.4 Computing Prefix Scores

Another interesting value is the *prefix score* of an item, which, apart from the Viterbi score, also includes the cost of the best search path leading to the item. Huang and Sagae (2010) use this quantity to order the items in a beam search on top of their dynamic programming method. In our framework, prefix scores can be computed as indicated in Figure 5. Alternatively, we can also use the more involved calculation employed by Huang and Sagae (2010), which allows them to get rid of the left-context vector from their items.[1]

4.5 Compatibility

So far we have restricted our attention to a concrete and extremely simplistic feature model. The feature models that are used in practical systems are considerably more complex, and not all of them are compatible with our framework in the sense that they can be integrated into our deduction system in the way described in Section 4.2.

For a simple example of a feature model that is incompatible with our tabulation, consider the model $\Phi' = \langle \mathsf{s}_0.\mathsf{rc}.\mathsf{w}\rangle$, whose single feature function extracts the word form of the right child (rc) of the topmost node on the stack. Even if we know the values of this feature for two computations γ_1, γ_2, we have no way to compute its value for the composed computation $f_{\mathsf{ra}}(\gamma_1, \gamma_2)$: This value coincides with the word form of the topmost node on the stack associated with γ_2, but in order to have access to it in the context of the ra rule, our feature model would need to also include the feature function $\mathsf{s}_0.\mathsf{w}$.

The example just given raises the question whether there is a general criterion based on which we can decide if a given feature model is compatible with our tabulation. An attempt to provide such a criterion has been made by Huang and Sagae (2010), who define a constraint on feature models called "monotonicity" and claim that this constraint guarantees that feature values can be computed using their dynamic programming approach. Unfortunately, this claim is wrong. In particular, the feature model Φ' given above is "monotonic", but cannot be tabulated, neither in our nor in their framework. In general, it seems clear that the question of compatibility is a question about the *relation* between the tabulation and the feature model, and not about the feature model alone. To find practically useful characterizations of compatibility is an interesting avenue for future research.

5 Arc-Eager Model

Up to now, we have only discussed the arc-standard model. In this section, we show that the framework of push computations also provides a tabulation of another widely-used model for dependency parsing, the *arc-eager model* (Nivre, 2003).

[1] The essential idea in the calculation by Huang and Sagae (2010) is to delegate (in the computation of the Viterbi score) the scoring of sh transitions to the inference rules for la/ra.

$$(\sigma, i|\beta, A) \vdash (\sigma|i, \beta, A) \quad (\mathsf{sh})$$
$$(\sigma|i, j|\beta, A) \vdash (\sigma, j|\beta, A \cup \{j \rightarrow i\}) \quad (\mathsf{la_e})$$
only if i does not have an incoming arc
$$(\sigma|i, j|\beta, A) \vdash (\sigma|i|j, \beta, A \cup \{i \rightarrow j\}) \quad (\mathsf{ra_e})$$
$$(\sigma|i, \beta, A) \vdash (\sigma, \beta, A) \quad (\mathsf{re})$$
only if i has an incoming arc

Figure 6: Transitions in the arc-eager model.

5.1 Transition System

The arc-eager model has three types of transitions, shown in Figure 6: SHIFT (sh) works just like in arc-standard, moving the first node in the buffer to the stack. LEFT-ARC ($\mathsf{la_e}$) creates a new arc with the first node in the buffer as the head and the topmost node on the stack as the dependent, and pops the stack. It can only be applied if the topmost node on the stack has not already been assigned a head, so as to preserve the single-head constraint. RIGHT-ARC ($\mathsf{ra_e}$) creates an arc in the opposite direction as LEFT-ARC, and moves the first node in the buffer to the stack. Finally, REDUCE (re) simply pops the stack; it can only be applied if the topmost node on the stack has already been assigned a head.

Note that, unlike in the case of arc-standard, the parsing process in the arc-eager model is not bottom-up: the right dependents of a node are attached before they have been assigned their own right dependents.

5.2 Shift-Reduce Parsing

If we look at the specification of the transitions of the arc-standard and the arc-eager model and restrict our attention to the effect that they have on the stack and the buffer, then we can see that all seven transitions fall into one of three types:

$$(\sigma, i|\beta) \vdash (\sigma|i, \beta) \quad \mathsf{sh}, \mathsf{ra_e} \quad \text{(T1)}$$
$$(\sigma|i|j, \beta) \vdash (\sigma|j, \beta) \quad \mathsf{la} \quad \text{(T2)}$$
$$(\sigma|i, \beta) \vdash (\sigma, \beta) \quad \mathsf{ra}, \mathsf{la_e}, \mathsf{re} \quad \text{(T3)}$$

We refer to transitions of type T1 as *shift* and to transitions of type T2 and T3 as *reduce transitions*.

The crucial observation now is that the concept of push computations and the approach to their tabulation that we have taken for the arc-standard system can easily be generalized to other transition systems whose transitions are of the type *shift* or *reduce*. In particular, the proof of the correctness of our deduction system that we gave in Section 3 still goes through if instead of sh we write "shift" and instead of la and ra we write "reduce".

5.3 Deduction System

Generalizing our construction for the arc-standard model along these lines, we obtain a tabulation of the arc-eager model. Just like in the case of arc-standard, each single shift transition in that model (be it sh or $\mathsf{ra_e}$) constitutes a push computation, while the reduce transitions induce operations $f_{\mathsf{la_e}}$ and f_{re}. The only difference is that the preconditions of $\mathsf{la_e}$ and re must be met. Therefore, $f_{\mathsf{la_e}}(\gamma_1, \gamma_2)$ is only defined if the topmost node on the stack in the final configuration of γ_2 has not yet been assigned a head, and $f_{\mathsf{re}}(\gamma_1, \gamma_2)$ is only defined in the opposite case.

Item form. In our deduction system for the arc-eager model we use items of the form $[i, h^b, j]$, where $0 \leq i \leq h < j \leq |w|$, and $b \in \{0, 1\}$. An item $[i, h^b, j]$ has the same meaning as the corresponding item in our deduction system for arc-standard, but also keeps record of whether the node h has been assigned a head ($b = 1$) or not ($b = 0$).

Goal. The only goal item is $[0, 0^0, |w|]$. (The item $[0, 0^1, |w|]$ asserts that the node 0 has a head, which never happens in a complete computation.)

Axioms. Reasoning as in arc-standard, the axioms of the deduction system for the arc-eager model are the items of the form $[i, i^0, i + 1]$ and $[j, j^1, j + 1]$, where $j > 0$: the former correspond to the push computations obtained from a single sh, the latter to those obtained from a single $\mathsf{ra_e}$, which apart from shifting a node also assigns it a head.

Inference rules. Also analogously to arc-standard, if we know that there exists a push computation γ_1 of the form asserted by the item $[i, h^b, k]$, and a push computation γ_2 of the form asserted by $[k, g^0, j]$, where $j < |w|$, then we can build the push computation $f_{\mathsf{la_e}}(\gamma_1, \gamma_2)$ of the form asserted by the item $[i, h^b, j]$. Similarly, if γ_2 is of the form asserted by $[k, g^1, j]$, then we can build $f_{\mathsf{re}}(\gamma_1, \gamma_2)$, which again is of the form by asserted $[i, h^b, j]$. Thus:

$$\frac{[i, i^b, k] \quad [k, k^0, j]}{[i, i^b, j]} \ (\mathsf{la_e}), \qquad \frac{[i, i^b, k] \quad [k, k^1, j]}{[i, i^b, j]} \ (\mathsf{re}).$$

Item form: $[i^b, j]\,, 0 \leq i < j \leq |w|\,, b \in \{0,1\}$ **Goal:** $[0^0, |w|]$ **Axioms:** $[0^0, 1]$

$$\frac{[i^b, j]}{[j^0, j+1]}\ (\mathsf{sh}) \qquad \frac{[i^b, k]\ \ [k^0, j]}{[i^b, j]}\ (\mathsf{la_e}; j \rightarrow k)\,, j < |w| \qquad \frac{[i^b, j]}{[j^1, j+1]}\ (\mathsf{ra_e}; i \rightarrow j) \qquad \frac{[i^b, k]\ \ [k^1, j]}{[i^b, j]}\ (\mathsf{re})$$

Figure 7: Deduction system for the arc-eager model.

As mentioned above, the correctness and non-ambiguity of the system can be proved as in Section 3. Features can be added in the same way as discussed in Section 4.

5.4 Computational Complexity

Looking at the inference rules, it is clear that an implementation of the deduction system for arc-eager takes space in $\mathcal{O}(|w|^3)$ and time in $\mathcal{O}(|w|^5)$, just like in the case of arc-standard. However, a closer inspection reveals that we can give even tighter bounds.

In all derivable items $[i, h^b, j]$, it holds that $i = h$. This can easily be shown by induction: The property holds for the axioms, and the first two indexes of a consequent of a deduction rule coincide with the first two indexes of the left antecedent. Thus, if we use the notation $[i^b, k]$ as a shorthand for $[i, i^b, k]$, then we can rewrite the inference rules for the arc-eager system as in Figure 7, where, additionally, we have added unary rules for sh and ra and restricted the set of axioms along the lines set out in Section 4.2. With this formulation, it is apparent that the space complexity of the generic implementation of the deduction system is in fact even in $\mathcal{O}(|w|^2)$, and its time complexity is in $\mathcal{O}(|w|^3)$.

6 Hybrid Model

We now reverse the approach that we have taken in the previous sections: Instead of tabulating a transition system in order to get a dynamic-programming parser that simulates its computations, we start with a tabular parser and derive a transition system from it. In the new model, dependency trees are built bottom-up as in the arc-standard model, but the set of all computations in the system can be tabulated in space $\mathcal{O}(|w|^2)$ and time $\mathcal{O}(|w|^3)$, as in arc-eager.

6.1 Deduction System

Gómez-Rodríguez et al. (2008) present a deductive version of the dependency parser of Yamada and Matsumoto (2003); their deduction system is given in Figure 8. The generic implementation of the deduction system takes space $\mathcal{O}(|w|^2)$ and time $\mathcal{O}(|w|^3)$.

In the original interpretation of the deduction system, an item $[i, j]$ asserts the existence of a pair of (projective) dependency trees: the first tree rooted at token w_i, having all nodes in the substring $w_i \cdots w_{k-1}$ as descendants, where $i < k \leq j$; and the second tree rooted at token w_j, having all nodes in the substring $w_k \cdots w_j$ as descendants. (Note that we use fencepost indexes, while Gómez-Rodríguez et al. (2008) indexes positions.)

6.2 Transition System

In the context of our tabulation framework, we adopt a new interpretation of items: An item $[i, j]$ has the same meaning as an item $[i, i, j]$ in the tabulation of the arc-standard model; for every configuration c with $\beta(c) = \beta_i$, it asserts the existence of a push computation that starts with c and ends with a configuration c' for which $\beta(c') = \beta_j$ and $\sigma(c') = \sigma(c)|i$.

If we interpret the inference rules of the system in terms of composition operations on push computations as usual, and also take the intended direction of the dependency arcs into account, then this induces a transition system with three transitions:

$$\begin{aligned}
(\sigma, i\,|\beta, A) &\vdash (\sigma|i, \beta, A) && (\mathsf{sh})\\
(\sigma|i, j\,|\beta, A) &\vdash (\sigma, j\,|\beta, A \cup \{j \rightarrow i\}) && (\mathsf{la_h})\\
(\sigma|i\,|j, \beta, A) &\vdash (\sigma|i, \beta, A \cup \{i \rightarrow j\}) && (\mathsf{ra})
\end{aligned}$$

We call this transition system the *hybrid model*, as sh and ra are just like in arc-standard, while $\mathsf{la_h}$ is like the LEFT-ARC transition in the arc-eager model ($\mathsf{la_e}$), except that it does not have the precondition. Like the arc-standard but unlike the arc-eager model, the hybrid model builds dependencies bottom-up.

7 Conclusion

In this paper, we have provided a general technique for the tabulation of transition-based dependency parsers, and applied it to obtain dynamic programming algorithms for two widely-used parsing models,

Item form: $[i, j]\,, 0 \leq i < j \leq |w|$ **Goal:** $[0, |w|]$ **Axioms:** $[0, 1]$

Inference rules: $\dfrac{[i, j]}{[j, j+1]}$ (sh) $\quad \dfrac{[i, k] \quad [k, j]}{[i, j]}$ $(\mathsf{la_h}; j \rightarrow k)\,, j < |w|$ $\quad \dfrac{[i, k] \quad [k, j]}{[i, j]}$ $(\mathsf{ra}; i \rightarrow k)$

Figure 8: Deduction system for the hybrid model.

arc-standard and (for the first time) arc-eager. The basic idea behind our technique is the same as the one implemented by Huang and Sagae (2010) for the special case of the arc-standard model, but instead of their graph-structured stack representation we use a tabulation akin to Lang's approach to the simulation of pushdown automata (Lang, 1974). This considerably simplifies both the presentation and the implementation of parsing algorithms. It has also enabled us to give simple proofs of correctness and establish relations between transition-based parsers and existing parsers based on dynamic programming.

While this paper has focused on the theoretical aspects and the analysis of dynamic programming versions of transition-based parsers, an obvious avenue for future work is the evaluation of the empirical performance and efficiency of these algorithms in connection with specific feature models. The feature models used in transition-based dependency parsing are typically very expressive, and exhaustive search with them quickly becomes impractical even for our cubic-time algorithms of the arc-eager and hybrid model. However, Huang and Sagae (2010) have provided evidence that the use of dynamic programming on top of a transition-based dependency parser can improve accuracy even without exhaustive search. The tradeoff between expressivity of the feature models on the one hand and the efficiency of the search on the other is a topic that we find worth investigating. Another interesting observation is that dynamic programming makes it possible to use predictive features, which cannot easily be integrated into a non-tabular transition-based parser. This could lead to the development of parsing models that cross the border between transition-based and tabular parsing.

Acknowledgments

All authors contributed equally to the work presented in this paper. M. K. wrote most of the manuscript. C. G.-R. has been partially supported by Ministerio de Educación y Ciencia and FEDER (HUM2007-66607-C04) and Xunta de Galicia (PGIDIT07SIN005206PR, Rede Galega de Procesamento da Linguaxe e Recuperación de Información, Rede Galega de Lingüística de Corpus, Bolsas Estadías INCITE/FSE cofinanced).

References

Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL)*, pages 166–170, New York, USA.

Sylvie Billot and Bernard Lang. 1989. The structure of shared forests in ambiguous parsing. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 143–151, Vancouver, Canada.

Michael Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 184–191, Santa Cruz, CA, USA.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and Head Automaton Grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 457–464, College Park, MD, USA.

Carlos Gómez-Rodríguez, John Carroll, and David J. Weir. 2008. A deductive approach to dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL): Human Language Technologies*, pages 968–976, Columbus, OH, USA.

Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–605.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1077–1086, Uppsala, Sweden.

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

Bernard Lang. 1974. Deterministic techniques for efficient non-deterministic parsers. In Jacques Loecx,

editor, *Automata, Languages and Programming, 2nd Colloquium, University of Saarbrücken, July 29–August 2, 1974*, number 14 in Lecture Notes in Computer Science, pages 255–269. Springer.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 40–51, Singapore.

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

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain.

Joakim Nivre. 2006. *Inductive Dependency Parsing*, volume 34 of *Text, Speech and Language Technology*. Springer.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Stuart M. Shieber, Yves Schabes, and Fernando Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1–2):3–36.

Masaru Tomita. 1986. *Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems*. Springer.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the Eighth International Workshop on Parsing Technologies (IWPT)*, pages 195–206, Nancy, France.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 562—571, Honolulu, HI, USA.

Shift-Reduce CCG Parsing

Yue Zhang
University of Cambridge
Computer Laboratory
yue.zhang@cl.cam.ac.uk

Stephen Clark
University of Cambridge
Computer Laboratory
stephen.clark@cl.cam.ac.uk

Abstract

CCGs are directly compatible with binary-branching bottom-up parsing algorithms, in particular CKY and shift-reduce algorithms. While the chart-based approach has been the dominant approach for CCG, the shift-reduce method has been little explored. In this paper, we develop a shift-reduce CCG parser using a discriminative model and beam search, and compare its strengths and weaknesses with the chart-based C&C parser. We study different errors made by the two parsers, and show that the shift-reduce parser gives competitive accuracies compared to C&C. Considering our use of a small beam, and given the high ambiguity levels in an automatically-extracted grammar and the amount of information in the CCG lexical categories which form the shift actions, this is a surprising result.

1 Introduction

Combinatory Categorial Grammar (CCG; Steedman (2000)) is a lexicalised theory of grammar which has been successfully applied to a range of problems in NLP, including treebank creation (Hockenmaier and Steedman, 2007), syntactic parsing (Hockenmaier, 2003; Clark and Curran, 2007), logical form construction (Bos et al., 2004) and surface realization (White and Rajkumar, 2009). From a parsing perspective, the C&C parser (Clark and Curran, 2007) has been shown to be competitive with state-of-the-art statistical parsers on a variety of test suites, including those consisting of grammatical relations (Clark and Curran, 2007), Penn Treebank phrase-structure trees (Clark and Curran, 2009), and unbounded dependencies (Rimell et al., 2009).

The binary branching nature of CCG means that it is naturally compatible with bottom-up parsing algorithms such as shift-reduce and CKY (Ades and Steedman, 1982; Steedman, 2000). However, the parsing work by Clark and Curran (2007), and also Hockenmaier (2003) and Fowler and Penn (2010), has only considered chart-parsing. In this paper we fill a gap in the CCG literature by developing a shift-reduce parser for CCG.

Shift-reduce parsers have become popular for dependency parsing, building on the initial work of Yamada and Matsumoto (2003) and Nivre and Scholz (2004). One advantage of shift-reduce parsers is that the scoring model can be defined over actions, allowing highly efficient parsing by using a greedy algorithm in which the highest scoring action (or a small number of possible actions) is taken at each step. In addition, high accuracy can be maintained by using a model which utilises a rich set of features for making each local decision (Nivre et al., 2006).

Following recent work applying global discriminative models to large-scale structured prediction problems (Collins and Roark, 2004; Miyao and Tsujii, 2005; Clark and Curran, 2007; Finkel et al., 2008), we build our shift-reduce parser using a global linear model, and compare it with the chart-based C&C parser. Using standard development and test sets from CCGbank, our shift-reduce parser gives a labeled F-measure of 85.53%, which is competitive with the 85.45% F-measure of the C&C parser on recovery of predicate-argument dependencies from CCGbank. Hence our work shows that

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 683–692,
Portland, Oregon, June 19-24, 2011.

transition-based parsing can be successfully applied to CCG, improving on earlier attempts such as Hassan et al. (2008). Detailed analysis shows that our shift-reduce parser yields a higher precision, lower recall and higher F-score on most of the common CCG dependency types compared to C&C.

One advantage of the shift-reduce parser is that it easily handles sentences for which it is difficult to find a spanning analysis, which can happen with CCG because the lexical categories at the leaves of a derivation place strong contraints on the set of possible derivations, and the supertagger which provides the lexical categories sometimes makes mistakes. Unlike the C&C parser, the shift-reduce parser naturally produces fragmentary analyses when appropriate (Nivre et al., 2006), and can produce sensible local structures even when a full spanning analysis cannot be found.[1]

Finally, considering this work in the wider parsing context, it provides an interesting comparison between heuristic beam search using a rich set of features, and optimal dynamic programming search where the feature range is restricted. We are able to perform this comparison because the use of the CCG supertagger means that the C&C parser is able to build the complete chart, from which it can find the optimal derivation, with no pruning whatsoever at the parsing stage. In contrast, the shift-reduce parser uses a simple beam search with a relatively small beam. Perhaps surprisingly, given the ambiguity levels in an automatically-extracted grammar, and the amount of information in the CCG lexical categories which form the shift actions, the shift-reduce parser using heuristic beam search is able to outperform the chart-based parser.

2 CCG Parsing

CCG, and the application of CCG to wide-coverage parsing, is described in detail elsewhere (Steedman, 2000; Hockenmaier, 2003; Clark and Curran, 2007). Here we provide only a short description.

During CCG parsing, adjacent categories are combined using CCG's combinatory rules. For example, a verb phrase in English ($S\backslash NP$) can combine with an NP to its left using function application:

$$NP \;\; S\backslash NP \Rightarrow S$$

Categories can also combine using function composition, allowing the combination of "may" $((S\backslash NP)/(S\backslash NP))$ and "like" $((S\backslash NP)/NP)$ in coordination examples such as "John may like but may detest Mary":

$$(S\backslash NP)/(S\backslash NP) \;\; (S\backslash NP)/NP \Rightarrow (S\backslash NP)/NP$$

In addition to binary rules, such as function application and composition, there are also unary rules which operate on a single category in order to change its type. For example, forward type-raising can change a subject NP into a complex category looking to the right for a verb phrase:

$$NP \Rightarrow S/(S\backslash NP)$$

An example CCG derivation is given in Section 3.

The resource used for building wide-coverage CCG parsers of English is CCGbank (Hockenmaier and Steedman, 2007), a version of the Penn Treebank in which each phrase-structure tree has been transformed into a normal-form CCG derivation. There are two ways to extract a grammar from this resource. One approach is to extract a lexicon, i.e. a mapping from words to sets of lexical categories, and then manually define the combinatory rule schemas, such as functional application and composition, which combine the categories together. The derivations in the treebank are then used to provide training data for the statistical disambiguation model. This is the method used in the C&C parser.[2]

The second approach is to read the complete grammar from the derivations, by extracting combinatory rule *instances* from the local trees consisting of a parent category and one or two child categories, and applying only those instances during parsing. (These rule instances also include rules to deal with punctuation and unary type-changing rules, in addition to instances of the combinatory rule schemas.) This is the method used by Hockenmaier (2003) and is the method we adopt in this paper.

Fowler and Penn (2010) demonstrate that the second extraction method results in a context-free approximation to the grammar resulting from the first

[1] See e.g. Riezler et al. (2002) and Zhang et al. (2007) for chart-based parsers which can produce fragmentary analyses.

[2] Although the C&C default mode applies a restriction for efficiency reasons in which only rule instances seen in CCGbank can be applied, making the grammar of the second type.

method, which has the potential to produce a mildly-context sensitive grammar (given the existence of certain combinatory rules) (Weir, 1988). However, it is important to note that the advantages of CCG, in particular the tight relationship between syntax and semantic interpretation, are still maintained with the second approach, as Fowler and Penn (2010) argue.

3 The Shift-reduce CCG Parser

Given an input sentence, our parser uses a stack of partial derivations, a queue of incoming words, and a series of actions—derived from the rule instances in CCGbank—to build a derivation tree. Following Clark and Curran (2007), we assume that each input word has been assigned a POS-tag (from the Penn Treebank tagset) and a set of CCG lexical categories. We use the same maximum entropy POS-tagger and supertagger as the C&C parser. The derivation tree can be transformed into CCG dependencies or grammatical relations by a post-processing step, which essentially runs the C&C parser deterministically over the derivation, interpreting the derivation and generating the required output.

The configuration of the parser, at each step of the parsing process, is shown in part (a) of Figure 1, where the stack holds the partial derivation trees that have been built, and the queue contains the incoming words that have not been processed. In the figure, $\mathtt{S}_{(\mathtt{H})}$ represents a category S on the stack with head word H, while $\mathtt{Q}_i$ represents a word in the incoming queue.

The set of action types used by the parser is as follows: {SHIFT, COMBINE, UNARY, FINISH}. Each action type represents a set of possible actions available to the parser at each step in the process.

The SHIFT-X action pushes the next incoming word onto the stack, and assigns the lexical category X to the word (Figure 1(b)). The label X can be any lexical category from the set assigned to the word being shifted by the supertagger. Hence the shift action performs lexical category disambiguation. This is in contrast to a shift-reduce dependency parser in which a shift action typically just pushes a word onto the stack.

The COMBINE-X action pops the top two nodes off the stack, and combines them into a new node, which is pushed back on the stack. The category of the new node is X. A COMBINE action corresponds to a combinatory rule in the CCG grammar (or one of the additional punctuation or type-changing rules), which is applied to the categories of the top two nodes on the stack.

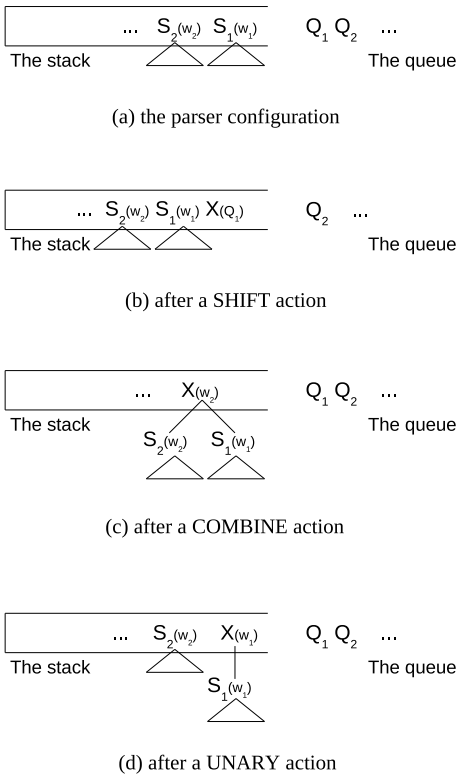


Figure 1: The parser configuration and set of actions.

The UNARY-X action pops the top of the stack, transforms it into a new node with category X, and pushes the new node onto the stack. A UNARY action corresponds to a unary type-changing or type-raising rule in the CCG grammar, which is applied to the category on top of the stack.

The FINISH action terminates the parsing process; it can be applied when all input words have been shifted onto the stack. Note that the FINISH action can be applied when the stack contains more than one node, in which case the parser produces a set of partial derivation trees, each corresponding to a node on the stack. This sometimes happens when a full derivation tree cannot be built due to supertagging errors, and provides a graceful solution to the problem of producing high-quality fragmentary parses when necessary.

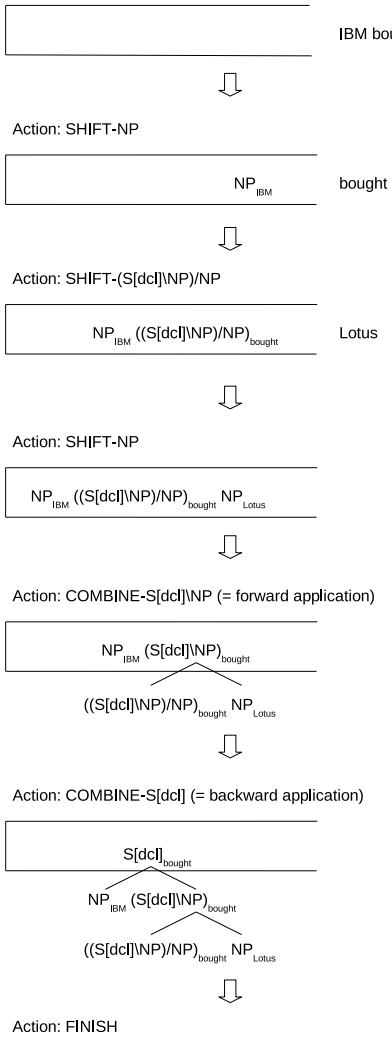


Figure 2: An example parsing process.

Figure 2 shows the shift-reduce parsing process for the example sentence "IBM bought Lotus". First the word "IBM" is shifted onto the stack as an NP; then "bought" is shifted as a transitive verb looking for its object NP on the right and subject NP on the left ((S[dcl]\NP)/NP); and then "Lotus" is shifted as an NP. Then "bought" is combined with its object "Lotus" resulting in a verb phrase looking for its subject on the left (S[dcl]\NP). Finally, the resulting verb phrase is combined with its subject, resulting in a declarative sentence (S[dcl]).

A key difference with previous work on shift-reduce dependency (Nivre et al., 2006) and CFG (Sagae and Lavie, 2006b) parsing is that, for CCG, there are many more shift actions – a shift action for each word-lexical category pair. Given the amount of syntactic information in the lexical categories, the choice of correct category, from those supplied by the supertagger, is often a difficult one, and often a choice best left to the parsing model. The C&C parser solves this problem by building the complete packed chart consistent with the lexical categories supplied by the supertagger, leaving the selection of the lexical categories to the Viterbi algorithm. For the shift-reduce parser the choice is also left to the parsing model, but in contrast to C&C the correct lexical category could be lost at any point in the heuristic search process. Hence it is perhaps surprising that we are able to achieve a high parsing accuracy of 85.5%, given a relatively small beam size.

4 Decoding

Greedy local search (Yamada and Matsumoto, 2003; Sagae and Lavie, 2005; Nivre and Scholz, 2004) has typically been used for decoding in shift-reduce parsers, while beam-search has recently been applied as an alternative to reduce error-propagation (Johansson and Nugues, 2007; Zhang and Clark, 2008; Zhang and Clark, 2009; Huang et al., 2009). Both greedy local search and beam-search have linear time complexity. We use beam-search in our CCG parser.

To formulate the decoding algorithm, we define a *candidate item* as a tuple $\langle S, Q, F \rangle$, where S represents the stack with partial derivations that have been built, Q represents the queue of incoming words that have not been processed, and F is a boolean value that represents whether the candidate item has been finished. A candidate item is *finished* if and only if the `FINISH` action has been applied to it, and no more actions can be applied to a candidate item after it reaches the finished status. Given an input sentence, we define the *start item* as the unfinished item with an empty stack and the whole input sentence as the incoming words. A derivation is built from the start item by repeated applications of actions until the item is finished.

To apply beam-search, an agenda is used to hold the N-best partial (unfinished) candidate items at each parsing step. A separate *candidate output* is

```
function DECODE(input, agenda, list, N,
                grammar, candidate_output):
  agenda.clear()
  agenda.insert(GETSTARTITEM(input))
  candidate_output = NONE
  while not agenda.empty():
    list.clear()
    for item in agenda:
      for action in grammar.getActions(item):
        item' = item.apply(action)
        if item'.F == TRUE:
          if candidate_output == NONE or
            item'.score > candidate_output.score:
            candidate_output = item'
          else:
            list.append(item')
    agenda.clear()
    agenda.insert(list.best(N))
```

Figure 3: The decoding algorithm; N is the agenda size

	feature templates
1	S_0wp, S_0c, S_0pc, S_0wc, S_1wp, S_1c, S_1pc, S_1wc, S_2pc, S_2wc, S_3pc, S_3wc,
2	Q_0wp, Q_1wp, Q_2wp, Q_3wp,
3	S_0Lpc, S_0Lwc, S_0Rpc, S_0Rwc, S_0Upc, S_0Uwc, S_1Lpc, S_1Lwc, S_1Rpc, S_1Rwc, S_1Upc, S_1Uwc,
4	S_0wcS_1wc, S_0cS_1w, S_0wS_1c, S_0cS_1c, S_0wcQ_0wp, S_0cQ_0wp, S_0wcQ_0p, S_0cQ_0p, S_1wcQ_0wp, S_1cQ_0wp, S_1wcQ_0p, S_1cQ_0p,
5	S_0wcS_1cQ_0p, S_0cS_1wcQ_0p, S_0cS_1cQ_0wp, S_0cS_1cQ_0p, S_0pS_1pQ_0p, S_0wcQ_0pQ_1p, S_0cQ_0wpQ_1p, S_0cQ_0pQ_1wp, S_0cQ_0pQ_1p, S_0pQ_0pQ_1p, S_0wcS_1cS_2c, S_0cS_1wcS_2c, S_0cS_1cS_2wc, S_0cS_1cS_2c, S_0pS_1pS_2p,
6	S_0cS_0HcS_0Lc, S_0cS_0HcS_0Rc, S_1cS_1HcS_1Rc, S_0cS_0RcQ_0p, S_0cS_0RcQ_0w, S_0cS_0LcS_1c, S_0cS_0LcS_1w, S_0cS_1cS_1Rc, S_0wS_1cS_1Rc.

Table 1: Feature templates.

used to record the current best finished item that has been found, since candidate items can be finished at different steps. Initially the agenda contains only the start item, and the *candidate output* is set to none. At each step during parsing, each candidate item from the agenda is extended in all possible ways by applying one action according to the grammar, and a number of new candidate items are generated. If a newly generated candidate is finished, it is compared with the current *candidate output*. If the candidate output is none or the score of the newly generated candidate is higher than the score of the candidate output, the candidate output is replaced with the newly generated item; otherwise the newly generated item is discarded. If the newly generated candidate is unfinished, it is appended to a *list* of newly generated partial candidates. After all candidate items from the agenda have been processed, the agenda is cleared and the N-best items from the list are put on the agenda. Then the list is cleared and the parser moves on to the next step. This process repeats until the agenda is empty (which means that no new items have been generated in the previous step), and the candidate output is the final derivation. Pseudocode for the algorithm is shown in Figure 3.

5 Model and Training

We use a global linear model to score candidate items, trained discriminatively with the averaged perceptron (Collins, 2002). Features for a (finished or partial) candidate are extracted from each action that have been applied to build the candidate. Following Collins and Roark (2004), we apply the "early update" strategy to perceptron training: at any step during decoding, if neither the candidate output nor any item in the agenda is correct, decoding is stopped and the parameters are updated using the current highest scored item in the agenda or the candidate output, whichever has the higher score.

Table 1 shows the feature templates used by the parser. The symbols S_0, S_1, S_2 and S_3 in the table represent the top four nodes on the stack (if existent), and Q_0, Q_1, Q_2 and Q_3 represent the front four words in the incoming queue (if existent). S_0H and S_1H represent the subnodes of S_0 and S_1 that have the lexical head of S_0 and S_1, respectively. S_0L represents the left subnode of S_0, when the lexical head is from the right subnode. S_0R and S_1R represent the right subnode of S_0 and S_1, respectively,

when the lexical head is from the left subnode. If S_0 is built by a UNARY action, S_0U represents the only subnode of S_0. The symbols w, p and c represent the word, the POS, and the CCG category, respectively.

These rich feature templates produce a large number of features: 36 million after the first training iteration, compared to around 0.5 million in the C&C parser.

6 Experiments

Our experiments were performed using CCGBank (Hockenmaier and Steedman, 2007), which was split into three subsets for training (Sections 02–21), development testing (Section 00) and the final test (Section 23). Extracted from the training data, the CCG grammar used by our parser consists of 3070 binary rule instances and 191 unary rule instances.

We compute F-scores over labeled CCG dependencies and also lexical category accuracy. CCG dependencies are defined in terms of lexical categories, by numbering each argument slot in a complex category. For example, the first NP in a transitive verb category is a CCG dependency relation, corresponding to the subject of the verb. Clark and Curran (2007) gives a more precise definition. We use the `generate` script from the C&C tools[3] to transform derivations into CCG dependencies.

There is a mismatch between the grammar that `generate` uses, which is the same grammar as the C&C parser, and the grammar we extract from CCGbank, which contains more rule instances. Hence `generate` is unable to produce dependencies for some of the derivations our shift-reduce parser produces. In order to allow `generate` to process all derivations from the shift-reduce parser, we repeatedly removed rules that the `generate` script cannot handle from our grammar, until all derivations in the development data could be dealt with. In fact, this procedure potentially reduces the accuracy of the shift-reduce parser, but the effect is comparatively small because only about 4% of the development and test sentences contain rules that are not handled by the `generate` script.

All experiments were performed using automatically assigned POS-tags, with 10-fold cross validation used to assign POS-tags and lexical categories to the training data. At the supertagging stage, multiple lexical categories are assigned to each word in the input. For each word, the supertagger assigns all lexical categories whose forward-backward probability is above $\beta \cdot max$, where max is the highest lexical category probability for the word, and β is a threshold parameter. To give the parser a reasonable freedom in lexical category disambiguation, we used a small β value of 0.0001, which results in 3.6 lexical categories being assigned to each word on average in the training data. For training, but not testing, we also added the correct lexical category to the list of lexical categories for a word in cases when it was not provided by the supertagger.

Increasing the size of the beam in the parser beam search leads to higher accuracies but slower running time. In our development experiments, the accuracy improvement became small when the beam size reached 16, and so we set the size of the beam to 16 for the remainder of the experiments.

6.1 Development test accuracies

Table 2 shows the labeled precision (lp), recall (lr), F-score (lf), sentence-level accuracy (lsent) and lexical category accuracy (cats) of our parser and the C&C parser on the development data. We ran the C&C parser using the normal-form model (we reproduced the numbers reported in Clark and Curran (2007)), and copied the results of the hybrid model from Clark and Curran (2007), since the hybrid model is not part of the public release.

The accuracy of our parser is much better when evaluated on all sentences, partly because C&C failed on 0.94% of the data due to the failure to produce a spanning analysis. Our shift-reduce parser does not suffer from this problem because it produces fragmentary analyses for those cases. When evaluated on only those sentences that C&C could analyze, our parser gave 0.29% higher F-score. Our shift-reduce parser also gave higher accuracies on lexical category assignment. The sentence accuracy of our shift-reduce parser is also higher than C&C, which confirms that our shift-reduce parser produces reasonable sentence-level analyses, despite the possibility for fragmentary analysis.

[3] Available at http://svn.ask.it.usyd.edu.au/trac/candc/wiki; we used the `generate` and `evaluate` scripts, as well as the C&C parser, for evaluation and comparison.

	lp.	lr.	lf.	lsent.	cats.	evaluated on
shift-reduce	87.15%	82.95%	85.00%	33.82%	92.77%	all sentences
C&C (normal-form)	85.22%	82.52%	83.85%	31.63%	92.40%	all sentences
shift-reduce	87.55%	83.63%	85.54%	34.14%	93.11%	99.06% (C&C coverage)
C&C (hybrid)	–	–	85.25%	–	–	99.06% (C&C coverage)
C&C (normal-form)	85.22%	84.29%	84.76%	31.93%	92.83%	99.06% (C&C coverage)

Table 2: Accuracies on the development test data.

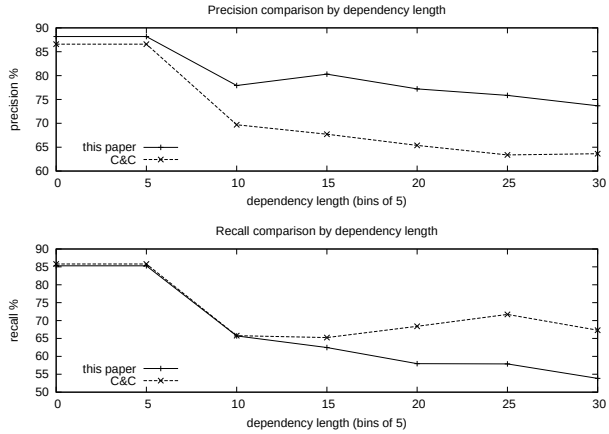


Figure 4: P & R scores relative to dependency length.

6.2 Error comparison with C&C parser

Our shift-reduce parser and the chart-based C&C parser offer two different solutions to the CCG parsing problem. The comparison reported in this section is similar to the comparison between the chart-based MSTParser (McDonald et al., 2005) and shift-reduce MaltParser (Nivre et al., 2006) for dependency parsing. We follow McDonald and Nivre (2007) and characterize the errors of the two parsers by sentence and dependency length and dependency type.

We measured precision, recall and F-score relative to different sentence lengths. Both parsers performed better on shorter sentences, as expected. Our shift-reduce parser performed consistently better than C&C on all sentence lengths, and there was no significant difference in the rate of performance degradation between the parsers as the sentence length increased.

Figure 4 shows the comparison of labeled precision and recall relative to the dependency length (i.e. the number of words between the head and dependent), in bins of size 5 (e.g. the point at x=5 shows the precision or recall for dependency lengths 1 – 5). This experiment was performed using the normal-form version of the C&C parser, and the evaluation was on the sentences for which C&C gave an analysis. The number of dependencies drops when the dependency length increases; there are 141, 180 and 124 dependencies from the gold-standard, C&C output and our shift-reduce parser output, respectively, when the dependency length is between 21 and 25, inclusive. The numbers drop to 47, 56 and 36 when the dependency length is between 26 and 30. The recall of our parser drops more quickly as the dependency length grows beyond 15. A likely reason is that the recovery of longer-range dependencies requires more processing steps, increasing the chance of the correct structure being thrown off the beam. In contrast, the precision did not drop more quickly than C&C, and in fact is consistently higher than C&C across all dependency lengths, which reflects the fact that the long range dependencies our parser managed to recover are comparatively reliable.

Table 3 shows the comparison of labeled precision (lp), recall (lr) and F-score (lf) for the most common CCG dependency types. The numbers for C&C are for the hybrid model, copied from Clark and Curran (2007). While our shift-reduce parser gave higher precision for almost all categories, it gave higher recall on only half of them, but higher F-scores for all but one dependency type.

6.3 Final results

Table 4 shows the accuracies on the test data. The numbers for the normal-form model are evaluated by running the publicly available parser, while those for the hybrid dependency model are from Clark and Curran (2007). Evaluated on all sentences, the accuracies of our parser are much higher than the C&C parser, since the C&C parser failed to produce any output for 10 sentences. When evaluating both

category	arg	lp. (o)	lp. (C)	lr. (o)	lr. (C)	lf. (o)	lf. (C)	freq.
N/N	1	**95.77%**	95.28%	**95.79%**	95.62%	**95.78%**	95.45%	7288
NP/N	1	**96.70%**	96.57%	**96.59%**	96.03%	**96.65%**	96.30%	4101
(NP\NP)/NP	2	**83.19%**	82.17%	**89.24%**	88.90%	**86.11%**	85.40%	2379
(NP\NP)/NP	1	**82.53%**	81.58%	**87.99%**	85.74%	**85.17%**	83.61%	2174
((S\NP)\(S\NP))/NP	3	**77.60%**	71.94%	71.58%	**73.32%**	**74.47%**	72.63%	1147
((S\NP)\(S\NP))/NP	2	**76.30%**	70.92%	70.60%	**71.93%**	**73.34%**	71.42%	1058
((S[dcl]\NP)/NP	2	**85.60%**	81.57%	84.30%	**86.37%**	**84.95%**	83.90%	917
PP/NP	1	73.76%	**75.06%**	**72.83%**	70.09%	**73.29%**	72.49%	876
((S[dcl]\NP)/NP	1	**85.32%**	81.62%	82.00%	**85.55%**	**83.63%**	83.54%	872
((S\NP)\(S\NP))	2	84.44%	**86.85%**	86.60%	**86.73%**	85.51%	**86.79%**	746

Table 3: Accuracy comparison on the most common CCG dependency types. (o) – our parser; (C) – C&C (hybrid)

	lp.	lr.	lf.	lsent.	cats.	evaluated
shift-reduce	87.43%	83.61%	85.48%	35.19%	93.12%	all sentences
C&C (normal-form)	85.58%	82.85%	84.20%	32.90%	92.84%	all sentences
shift-reduce	87.43%	83.71%	85.53%	35.34%	93.15%	99.58% (C&C coverage)
C&C (hybrid)	86.17%	84.74%	85.45%	32.92%	92.98%	99.58% (C&C coverage)
C&C (normal-form)	85.48%	84.60%	85.04%	33.08%	92.86%	99.58% (C&C coverage)
F&P (Petrov I-5)*	86.29%	85.73%	86.01%	–	–	– (F&P ∩ C&C coverage; 96.65% on dev. test)
C&C hybrid*	86.46%	85.11%	85.78%	–	–	– (F&P ∩ C&C coverage; 96.65% on dev. test)

Table 4: Comparison with C&C; final test. * – not directly comparable.

parsers on the sentences for which C&C produces an analysis, our parser still gave the highest accuracies. The shift-reduce parser gave higher precision, and lower recall, than C&C; it also gave higher sentence-level and lexical category accuracy.

The last two rows in the table show the accuracies of Fowler and Penn (2010) (F&P), who applied the CFG parser of Petrov and Klein (2007) to CCG, and the corresponding accuracies for the C&C parser on the same test sentences. F&P can be treated as another chart-based parser; their evaluation is based on the sentences for which both their parser and C&C produced dependencies (or more specifically those sentences for which `generate` could produce dependencies), and is not directly comparable with ours, especially considering that their test set is smaller and potentially slightly easier.

The final comparison is parser speed. The shift-reduce parser is linear-time (in both sentence length and beam size), and can analyse over 10 sentences per second on a 2GHz CPU, with a beam of 16, which compares very well with other constituency parsers. However, this is no faster than the chart-based C&C parser, although speed comparisons are difficult because of implementation differences (C&C uses heavily engineered C++ with a focus on efficiency).

7 Related Work

Sagae and Lavie (2006a) describes a shift-reduce parser for the Penn Treebank parsing task which uses best-first search to allow some ambiguity into the parsing process. Differences with our approach are that we use a beam, rather than best-first, search; we use a global model rather than local models chained together; and finally, our results surpass the best published results on the CCG parsing task, whereas Sagae and Lavie (2006a) matched the best PTB results only by using a parser combination.

Matsuzaki et al. (2007) describes similar work to ours but using an automatically-extracted HPSG, rather than CCG, grammar. They also use the generalised perceptron to train a disambiguation model. One difference is that Matsuzaki et al. (2007) use an approximating CFG, in addition to the supertagger, to improve the efficiency of the parser.

Ninomiya et al. (2009) (and Ninomiya et al. (2010)) describe a greedy shift-reduce parser for HPSG, in which a single action is chosen at each parsing step, allowing the possibility of highly efficient parsing. Since the HPSG grammar has relatively tight constraints, similar to CCG, the possibility arises that a spanning analysis cannot be found for some sentences. Our approach to this problem was to allow the parser to return a fragmentary analysis; Ninomiya et al. (2009) adopt a different approach based on default unification.

Finally, our work is similar to the comparison of the chart-based MSTParser (McDonald et al., 2005) and shift-reduce MaltParser (Nivre et al., 2006) for dependency parsing. MSTParser can perform exhaustive search, given certain feature restrictions, because the complexity of the parsing task is lower than for constituent parsing. C&C can perform exhaustive search because the supertagger has already reduced the search space. We also found that approximate heuristic search for shift-reduce parsing, utilising a rich feature space, can match the performance of the optimal chart-based parser, as well as similar error profiles for the two CCG parsers compared to the two dependency parsers.

8 Conclusion

This is the first work to present competitive results for CCG using a transition-based parser, filling a gap in the CCG parsing literature. Considered in terms of the wider parsing problem, we have shown that state-of-the-art parsing results can be obtained using a global discriminative model, one of the few papers to do so without using a generative baseline as a feature. The comparison with C&C also allowed us to compare a shift-reduce parser based on heuristic beam search utilising a rich feature set with an optimal chart-based parser whose features are restricted by dynamic programming, with favourable results for the shift-reduce parser.

The complementary errors made by the chart-based and shift-reduce parsers opens the possibility of effective parser combination, following similar work for dependency parsing.

The parser code can be downloaded at `http://www.sourceforge.net/projects/zpar`, version 0.5.

Acknowledgements

We thank the anonymous reviewers for their suggestions. Yue Zhang and Stephen Clark are supported by the European Union Seventh Framework Programme (FP7-ICT-2009-4) under grant agreement no. 247762.

References

A. E. Ades and M. Steedman. 1982. On the order of words. *Linguistics and Philosophy*, pages 517 – 558.

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of COLING-04*, pages 1240–1246, Geneva, Switzerland.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Stephen Clark and James R. Curran. 2009. Comparing the accuracy of CCG and Penn Treebank parsers. In *Proceedings of ACL-2009 (short papers)*, pages 53–56, Singapore.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*, pages 111–118, Barcelona, Spain.

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

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Feature-based, conditional random field parsing. In *Proceedings of the 46th Meeting of the ACL*, pages 959–967, Columbus, Ohio.

Timothy A. D. Fowler and Gerald Penn. 2010. Accurate context-free parsing with Combinatory Categorial Grammar. In *Proceedings of ACL-2010*, Uppsala, Sweden.

H. Hassan, K. Sima'an, and A. Way. 2008. A syntactic language model based on incremental CCG parsing. In *Proceedings of the Second IEEE Spoken Language Technology Workshop*, Goa, India.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Liang Huang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce

parsing. In *Proceedings of the 2009 EMNLP Conference*, pages 1222–1231, Singapore.

Richard Johansson and Pierre Nugues. 2007. Incremental dependency parsing using online learning. In *Proceedings of the CoNLL/EMNLP Conference*, pages 1134–1138, Prague, Czech Republic.

Takuya Matsuzaki, Yusuke Miyao, and Jun ichi Tsujii. 2007. Efficient HPSG parsing with supertagging and CFG-filtering. In *Proceedings of IJCAI-07*, pages 1671–1676, Hyderabad, India.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP/CoNLL*, pages 122–131, Prague, Czech Republic.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Meeting of the ACL*, pages 91–98, Michigan, Ann Arbor.

Yusuke Miyao and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of the 43rd meeting of the ACL*, pages 83–90, University of Michigan, Ann Arbor.

Takashi Ninomiya, Takuya Matsuzaki, Nobuyuki Shimizu, and Hiroshi Nakagawa. 2009. Deterministic shift-reduce parsing for unification-based grammars by using default unification. In *Proceedings of EACL-09*, pages 603–611, Athens, Greece.

Takashi Ninomiya, Takuya Matsuzaki, Nobuyuki Shimizu, and Hiroshi Nakagawa. 2010. Deterministic shift-reduce parsing for unification-based grammars. *Journal of Natural Language Engineering*, DOI:10.1017/S1351324910000240.

J. Nivre and M. Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of COLING-04*, pages 64–70, Geneva, Switzerland.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of CoNLL*, pages 221–225, New York, USA.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of HLT/NAACL*, pages 404–411, Rochester, New York, April.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Meeting of the ACL*, pages 271–278, Philadelphia, PA.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of EMNLP-09*, pages 813–821, Singapore.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of IWPT*, pages 125–132, Vancouver, Canada.

Kenji Sagae and Alon Lavie. 2006a. A best-first probabilistic shift-reduce parser. In *Proceedings of COLING/ACL poster session*, pages 691–698, Sydney, Australia, July.

Kenji Sagae and Alon Lavie. 2006b. Parser combination by reparsing. In *Proceedings of HLT/NAACL, Companion Volume: Short Papers*, pages 129–132, New York, USA.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, Mass.

David Weir. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania.

Michael White and Rajakrishnan Rajkumar. 2009. Perceptron reranking for CCG realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 410–419, Singapore.

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

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of EMNLP-08*, Hawaii, USA.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the Chinese Treebank using a global discriminative model. In *Proceedings of IWPT*, Paris, France, October.

Yi Zhang, Valia Kordoni, and Erin Fitzgerald. 2007. Partial parse selection for robust deep processing. In *Proceedings of the ACL 2007 Workshop on Deep Linguistic Processing*, Prague, Czech Republic.

Web-Scale Features for Full-Scale Parsing

Mohit Bansal and **Dan Klein**
Computer Science Division
University of California, Berkeley
{mbansal, klein}@cs.berkeley.edu

Abstract

Counts from large corpora (like the web) can be powerful syntactic cues. Past work has used web counts to help resolve isolated ambiguities, such as binary noun-verb PP attachments and noun compound bracketings. In this work, we first present a method for generating web count features that address the full range of syntactic attachments. These features encode both surface evidence of lexical affinities as well as paraphrase-based cues to syntactic structure. We then integrate our features into full-scale dependency and constituent parsers. We show relative error reductions of 7.0% over the second-order dependency parser of McDonald and Pereira (2006), 9.2% over the constituent parser of Petrov et al. (2006), and 3.4% over a non-local constituent reranker.

1 Introduction

Current state-of-the art syntactic parsers have achieved accuracies in the range of 90% F1 on the Penn Treebank, but a range of errors remain. From a dependency viewpoint, structural errors can be cast as incorrect attachments, even for constituent (phrase-structure) parsers. For example, in the Berkeley parser (Petrov et al., 2006), about 20% of the errors are prepositional phrase attachment errors as in Figure 1, where a preposition-headed (IN) phrase was assigned an incorrect parent in the implied dependency tree. Here, the Berkeley parser (solid blue edges) incorrectly attaches *from debt* to the noun phrase *$ 30 billion* whereas the correct attachment (dashed gold edges) is to the verb *raising*. However, there are a range of error types, as shown in Figure 2. Here, (a) is a non-canonical PP

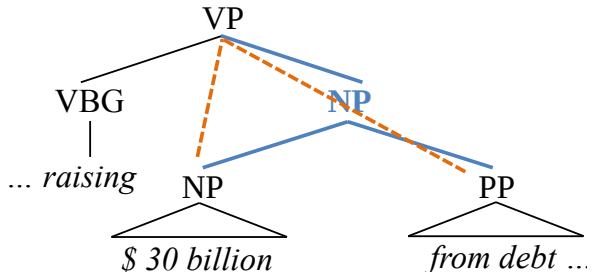


Figure 1: A PP attachment error in the parse output of the Berkeley parser (on Penn Treebank). Guess edges are in solid blue, gold edges are in dashed gold and edges common in guess and gold parses are in black.

attachment ambiguity where *by yesterday afternoon* should attach to *had already*, (b) is an NP-internal ambiguity where *half a* should attach to *dozen* and not to *newspapers*, and (c) is an adverb attachment ambiguity, where *just* should modify *fine* and not the verb *'s*.

Resolving many of these errors requires information that is simply not present in the approximately 1M words on which the parser was trained. One way to access more information is to exploit surface counts from large corpora like the web (Volk, 2001; Lapata and Keller, 2004). For example, the phrase *raising from* is much more frequent on the Web than *$ x billion from*. While this 'affinity' is only a surface correlation, Volk (2001) showed that comparing such counts can often correctly resolve tricky PP attachments. This basic idea has led to a good deal of successful work on disambiguating isolated, binary PP attachments. For example, Nakov and Hearst (2005b) showed that looking for *paraphrase* counts can further improve PP resolution. In this case, the existence of reworded phrases like *raising it from* on the Web also imply a verbal at-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 693–702,
Portland, Oregon, June 19-24, 2011.

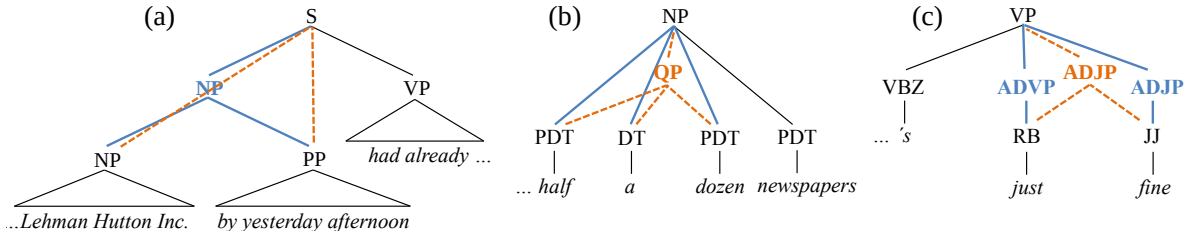


Figure 2: Different kinds of attachment errors in the parse output of the Berkeley parser (on Penn Treebank). Guess edges are in solid blue, gold edges are in dashed gold and edges common in guess and gold parses are in black.

tachment. Still other work has exploited Web counts for other isolated ambiguities, such as NP coordination (Nakov and Hearst, 2005b) and noun-sequence bracketing (Nakov and Hearst, 2005a; Pitler et al., 2010). For example, in (b), *half dozen* is more frequent than *half newspapers*.

In this paper, we show how to apply these ideas to all attachments in full-scale parsing. Doing so requires three main issues to be addressed. First, we show how features can be generated for arbitrary head-argument configurations. Affinity features are relatively straightforward, but paraphrase features, which have been hand-developed in the past, are more complex. Second, we integrate our features into full-scale parsing systems. For dependency parsing, we augment the features in the second-order parser of McDonald and Pereira (2006). For constituent parsing, we rerank the output of the Berkeley parser (Petrov et al., 2006). Third, past systems have usually gotten their counts from web search APIs, which does not scale to quadratically-many attachments in each sentence. Instead, we consider how to efficiently mine the Google n-grams corpus.

Given the success of Web counts for isolated ambiguities, there is relatively little previous research in this direction. The most similar work is Pitler et al. (2010), which use Web-scale n-gram counts for multi-way noun bracketing decisions, though that work considers only sequences of nouns and uses only affinity-based web features. Yates et al. (2006) use Web counts to filter out certain 'semantically bad' parses from extraction candidate sets but are not concerned with distinguishing amongst top parses. In an important contrast, Koo et al. (2008) smooth the sparseness of lexical features in a discriminative dependency parser by using cluster-based word-senses as intermediate abstractions in addition to POS tags (also see Finkel et al. (2008)). Their work also gives a way to tap into corpora beyond the training data, through cluster membership rather than explicit corpus counts and paraphrases.

This work uses a large web-scale corpus (Google n-grams) to compute features for the full parsing task. To show end-to-end effectiveness, we incorporate our features into state-of-the-art dependency and constituent parsers. For the dependency case, we can integrate them into the dynamic programming of a base parser; we use the discriminatively-trained MST dependency parser (McDonald et al., 2005; McDonald and Pereira, 2006). Our first-order web-features give 7.0% relative error reduction over the second-order dependency baseline of McDonald and Pereira (2006). For constituent parsing, we use a reranking framework (Charniak and Johnson, 2005; Collins and Koo, 2005; Collins, 2000) and show 9.2% relative error reduction over the Berkeley parser baseline. In the same framework, we also achieve 3.4% error reduction over the non-local syntactic features used in Huang (2008). Our web-scale features reduce errors for a range of attachment types. Finally, we present an analysis of influential features. We not only reproduce features suggested in previous work but also discover a range of new ones.

2 Web-count Features

Structural errors in the output of state-of-the-art parsers, constituent or dependency, can be viewed as attachment errors, examples of which are Figure 1 and Figure 2.[1] One way to address attachment errors is through features which factor over head-argument

[1]For constituent parsers, there can be minor tree variations which can result in the same set of induced dependencies, but these are rare in comparison.

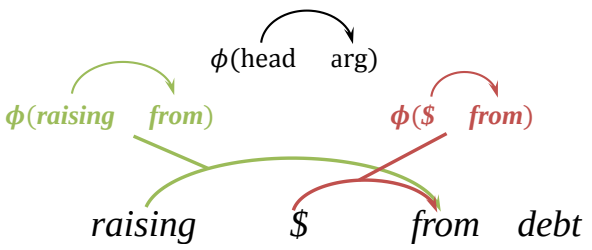


Figure 3: Features factored over head-argument pairs.

pairs, as is standard in the dependency parsing literature (see Figure 3). Here, we discuss which web-count based features $\phi(h, a)$ should fire over a given head-argument pair (we consider the words h and a to be indexed, and so features can be sensitive to their order and distance, as is also standard).

2.1 Affinity Features

Affinity statistics, such as lexical co-occurrence counts from large corpora, have been used previously for resolving individual attachments at least as far back as Lauer (1995) for noun-compound bracketing, and later for PP attachment (Volk, 2001; Lapata and Keller, 2004) and coordination ambiguity (Nakov and Hearst, 2005b). The approach of Lauer (1995), for example, would be to take an ambiguous noun sequence like *hydrogen ion exchange* and compare the various counts (or associated conditional probabilities) of n-grams like *hydrogen ion* and *hydrogen exchange*. The attachment with the greater score is chosen. More recently, Pitler et al. (2010) use web-scale n-grams to compute similar association statistics for longer sequences of nouns.

Our *affinity features* closely follow this basic idea of association statistics. However, because a real parser will not have access to gold-standard knowledge of the competing attachment sites (see Atterer and Schutze (2007)'s criticism of previous work), we must instead compute features for all possible head-argument pairs from our web corpus. Moreover, when there are only two competing attachment options, one can do things like directly compare two count-based heuristics and choose the larger. Integration into a parser requires features to be functions of single attachments, not pairwise comparisons between alternatives. A learning algorithm can then weight features so that they compare appropriately across parses.

We employ a collection of affinity features of varying specificity. The basic feature is the core adjacency count feature ADJ, which fires for all (h, a) pairs. What is specific to a particular (h, a) is the value of the feature, not its identity. For example, in a naive approach, the value of the ADJ feature might be the count of the query issued to the web-corpus – the 2-gram $q = ha$ or $q = ah$ depending on the order of h and a in the sentence. However, it turns out that there are several problems with this approach. First, rather than a single all-purpose feature like ADJ, the utility of such query counts will vary according to aspects like the parts-of-speech of h and a (because a high adjacency count is not equally informative for all kinds of attachments). Hence, we add more refined affinity features that are specific to each pair of POS tags, i.e. ADJ $\wedge$ POS(h) $\wedge$ POS(a). The values of these POS-specific features, however, are still derived from the same queries as before. Second, using real-valued features did not work as well as binning the query-counts (we used $b = floor(\log_r(\text{count})/5) * 5)$ and then firing indicator features ADJ $\wedge$ POS(h) $\wedge$ POS(a) $\wedge$ b for values of b defined by the query count. Adding still more complex features, we conjoin to the preceding features the order of the words h and a as they occur in the sentence, and the (binned) distance between them. For features which mark distances, wildcards ($\star$) are used in the query $q = h \star a$, where the number of wildcards allowed in the query is proportional to the binned distance between h and a in the sentence. Finally, we also include unigram variants of the above features, which are sensitive to only one of the head or argument. For all features used, we add cumulative variants where indicators are fired for all count bins b' up to query count bin b.

2.2 Paraphrase Features

In addition to measuring counts of the words present in the sentence, there exist clever ways in which paraphrases and other accidental indicators can help resolve specific ambiguities, some of which are discussed in Nakov and Hearst (2005a), Nakov and Hearst (2005b). For example, finding attestations of *eat : spaghetti with sauce* suggests a nominal attachment in *Jean ate spaghetti with sauce*. As another example, one clue that the example in Figure 1 is

a verbal attachment is that the proform paraphrase *raising it from* is commonly attested. Similarly, the attestation of *be noun prep* suggests nominal attachment.

These paraphrase features hint at the correct attachment decision by looking for web n-grams with special contexts that reveal syntax superficially. Again, while effective in their isolated disambiguation tasks, past work has been limited by both the range of attachments considered and the need to intuit these special contexts. For instance, frequency of the pattern *The noun prep* suggests noun attachment and of the pattern *verb adverb prep* suggests verb attachment for the preposition in the phrase *verb noun prep*, but these features were not in the manually brainstormed list.

In this work, we automatically generate a large number of paraphrase-style features for arbitrary attachment ambiguities. To induce our list of features, we first mine useful context words. We take each (correct) training dependency relation (h, a) and consider web n-grams of the form cha, hca, and hac. Aggregating over all h and a (of a given POS pair), we determine which context words c are most frequent in each position. For example, for h = *raising* and a = *from* (see Figure 1), we look at web n-grams of the form *raising c from* and see that one of the most frequent values of c on the web turns out to be the word *it*.

Once we have collected context words (for each position p in {BEFORE, MIDDLE, AFTER}), we turn each context word c into a collection of features of the form PARA $\wedge$ POS(h) $\wedge$ POS(a) $\wedge$ c $\wedge$ p $\wedge$ dir, where dir is the linear order of the attachment in the sentence. Note that h and a are head and argument words and so actually occur in the sentence, but c is a context word that generally does not. For such features, the queries that determine their values are then of the form cha, hca, and so on. Continuing the previous example, if the test set has a possible attachment of two words like h = *lowering* and a = *with*, we will fire a feature PARA $\wedge$ VBG $\wedge$ IN $\wedge$ *it* $\wedge$ MIDDLE $\wedge$ $\rightarrow$ with value (indicator bins) set according to the results of the query *lowering it with*. The idea is that if frequent occurrences of *raising it from* indicated a correct attachment between *raising* and *from*, frequent occurrences of *lowering it with* will indicate the correctness of an attachment between *lowering* and *with*. Finally, to handle the cases where no induced context word is helpful, we also construct abstracted versions of these paraphrase features where the context words c are collapsed to their parts-of-speech POS(c), obtained using a unigram-tagger trained on the parser training set. As discussed in Section 5, the top features learned by our learning algorithm duplicate the hand-crafted configurations used in previous work (Nakov and Hearst, 2005b) but also add numerous others, and, of course, apply to many more attachment types.

3 Working with Web n-Grams

Previous approaches have generally used search engines to collect count statistics (Lapata and Keller, 2004; Nakov and Hearst, 2005b; Nakov and Hearst, 2008). Lapata and Keller (2004) uses the number of page hits as the web-count of the queried n-gram (which is problematic according to Kilgarriff (2007)). Nakov and Hearst (2008) post-processes the first 1000 result snippets. One challenge with this approach is that an external search API is now embedded into the parser, raising issues of both speed and daily query limits, especially if all possible attachments trigger queries. Such methods also create a dependence on the quality and post-processing of the search results, limitations of the query process (for instance, search engines can ignore punctuation (Nakov and Hearst, 2005b)).

Rather than working through a search API (or scraper), we use an offline web corpus – the Google n-gram corpus (Brants and Franz, 2006) – which contains English n-grams (n = 1 to 5) and their observed frequency counts, generated from nearly 1 trillion word tokens and 95 billion sentences. This corpus allows us to efficiently access huge amounts of web-derived information in a compressed way, though in the process it limits us to local queries. In particular, we only use counts of n-grams of the form $x \star y$ where the gap length is ≤ 3.

Our system requires the counts from a large collection of these n-gram queries (around 4.5 million). The most basic queries are counts of head-argument pairs in contiguous *h a* and gapped *h* $\star$ *a* configurations.[2] Here, we describe how we process queries

[2]Paraphrase features give situations where we query $\star$ *h a*

of the form (q_1, q_2) with some number of wildcards in between. We first collect all such queries over all trees in preprocessing (so a new test set requires a new query-extraction phase). Next, we exploit a simple but efficient trie-based hashing algorithm to efficiently answer all of them in one pass over the n-grams corpus.

Consider Figure 4, which illustrates the data structure which holds our queries. We first create a trie of the queries in the form of a nested hashmap. The key of the outer hashmap is the first word q_1 of the query. The entry for q_1 points to an inner hashmap whose key is the final word q_2 of the query bigram. The values of the inner map is an array of 4 counts, to accumulate each of $(q_1 q_2)$, $(q_1 \star q_2)$, $(q_1 \star\star q_2)$, and $(q_1 \star\star\star q_2)$, respectively. We use k-grams to collect counts of $(q_1 ... q_2)$ with gap length = $k-2$, i.e. 2-grams to get $count(q_1 q_2)$, 3-grams to get $count(q_1 \star q_2)$ and so on.

With this representation of our collection of queries, we go through the web n-grams (n = 2 to 5) one by one. For an n-gram $w_1 ... w_n$, if the first n-gram word w_1 doesn't occur in the outer hashmap, we move on. If it does match (say $\bar{q}_1 = w_1$), then we look into the inner map for $\bar{q}_1$ and check for the final word w_n. If we have a match, we increment the appropriate query's result value.

In similar ways, we also mine the most frequent words that occur before, in between and after the head and argument query pairs. For example, to collect mid words, we go through the 3-grams $w_1 w_2 w_3$; if w_1 matches $\bar{q}_1$ in the outer hashmap and w_3 occurs in the inner hashmap for $\bar{q}_1$, then we store w_2 and the count of the 3-gram. After the sweep, we sort the context words in decreasing order of count. We also collect unigram counts of the head and argument words by sweeping over the unigrams once.

In this way, our work is linear in the size of the n-gram corpus, but essentially constant in the number of queries. Of course, if the number of queries is expected to be small, such as for a one-off parse of a single sentence, other solutions might be more appropriate; in our case, a large-batch setting, the number of queries was such that this formulation was chosen. Our main experiments (with no parallelization) took 115 minutes to sweep over the 3.8 billion

and *h a* $\star$; these are handled similarly.

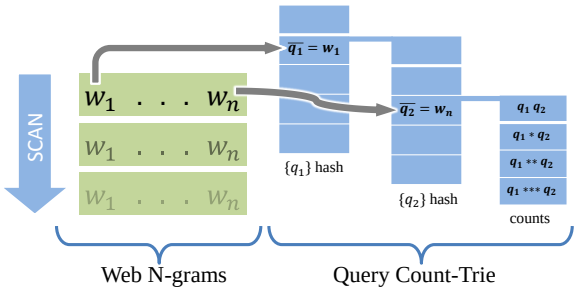


Figure 4: Trie-based nested hashmap for collecting ngram web-counts of queries.

n-grams (n = 1 to 5) to compute the answers to 4.5 million queries, much less than the time required to train the baseline parsers.

4 Parsing Experiments

Our features are designed to be used in full-sentence parsing rather than for limited decisions about isolated ambiguities. We first integrate our features into a dependency parser, where the integration is more natural and pushes all the way into the underlying dynamic program. We then add them to a constituent parser in a reranking approach. We also verify that our features contribute on top of standard reranking features.[3]

4.1 Dependency Parsing

For dependency parsing, we use the discriminatively-trained MSTParser[4], an implementation of first and second order MST parsing models of McDonald et al. (2005) and McDonald and Pereira (2006). We use the standard splits of Penn Treebank into training (sections 2-21), development (section 22) and test (section 23). We used the 'pennconverter'[5] tool to convert Penn trees from constituent format to dependency format. Following Koo et al. (2008), we used the MXPOST tagger (Ratnaparkhi, 1996) trained on the full training data to provide part-of-speech tags for the development

[3] All reported experiments are run on *all* sentences, i.e. without any length limit.

[4] http://sourceforge.net/projects/mstparser

[5] This supersedes 'Penn2Malt' and is available at http://nlp.cs.lth.se/software/treebank_converter. We follow its recommendation to patch WSJ data with NP bracketing by Vadas and Curran (2007).

	Order 2	+ Web features	% Error Redn.
Dev (sec 22)	92.1	**92.7**	7.6%
Test (sec 23)	91.4	**92.0**	7.0%

Table 1: UAS results for English WSJ dependency parsing. Dev is WSJ section 22 (all sentences) and Test is WSJ section 23 (all sentences). The order 2 baseline represents McDonald and Pereira (2006).

	$k = 1$	$k = 2$	$k = 10$	$k = 25$	$k = 50$	$k = 100$
Dev	90.6	92.3	95.1	95.8	96.2	96.5
Test	90.2	91.8	94.7	95.6	96.1	96.4

Table 2: Oracle F1-scores for k-best lists output by Berkeley parser for English WSJ parsing (Dev is section 22 and Test is section 23, all lengths).

and the test set, and we used 10-way jackknifing to generate tags for the training set.

We added our first-order Web-scale features to the MSTParser system to evaluate improvement over the results of McDonald and Pereira (2006).[6] Table 1 shows unlabeled attachments scores (UAS) for their second-order projective parser and the improved numbers resulting from the addition of our Web-scale features. Our first-order web-scale features show significant improvement even over their non-local *second*-order features.[7] Additionally, our web-scale features are at least an order of magnitude fewer in number than even their first-order base features.

4.2 Constituent Parsing

We also evaluate the utility of web-scale features on top of a state-of-the-art constituent parser – the Berkeley parser (Petrov et al., 2006), an unlexicalized phrase-structure parser. Because the underlying parser does not factor along lexical attachments, we instead adopt the discriminative reranking framework, where we generate the top-k candidates from the baseline system and then rerank this k-best list using (generally non-local) features.

Our baseline system is the Berkeley parser, from which we obtain k-best lists for the development set (WSJ section 22) and test set (WSJ section 23) using a grammar trained on all the training data (WSJ sections 2-21).[8] To get k-best lists for the training set, we use 3-fold jackknifing where we train a grammar on 2 folds to get parses for the third fold.[9] The oracle scores of the k-best lists (for different values of k) for the development and test sets are shown in Table 2. Based on these results, we used 50-best lists in our experiments. For discriminative learning, we used the averaged perceptron (Collins, 2002; Huang, 2008).

Our core feature is the log conditional likelihood of the underlying parser.[10] All other features are indicator features. First, we add all the Web-scale features as defined above. These features alone achieve a 9.2% relative error reduction. The affinity and paraphrase features contribute about two-fifths and three-fifths of this improvement, respectively. Next, we rerank with only the features (both local and non-local) from Huang (2008), a simplified merge of Charniak and Johnson (2005) and Collins (2000) (here *configurational*). These features alone achieve around the same improvements over the baseline as our web-scale features, even though they are highly non-local and extensive. Finally, we rerank with both our Web-scale features and the configurational features. When combined, our web-scale features give a further error reduction of 3.4% over the configurational reranker (and a combined error reduction of 12.2%). All results are shown in Table 3.[11]

5 Analysis

Table 4 shows error counts and relative reductions that our web features provide over the 2nd-order dependency baseline. While we do see substantial gains for classic PP (IN) attachment cases, we see equal or greater error reductions for a range of attachment types. Further, Table 5 shows how the to-

[6] Their README specifies 'training-k:5 iters:10 loss-type:nopunc decode-type:proj', which we used for all final experiments; we used the faster 'training-k:1 iters:5' setting for most development experiments.

[7] Work such as Smith and Eisner (2008), Martins et al. (2009), Koo and Collins (2010) has been exploring more non-local features for dependency parsing. It will be interesting to see how these features interact with our web features.

[8] Settings: 6 iterations of split and merge with smoothing.

[9] Default: we ran the Berkeley parser in its default 'fast' mode; the output k-best lists are ordered by max-rule-score.

[10] This is output by the flag -confidence. Note that baseline results with just this feature are slightly worse than 1-best results because the k-best lists are generated by max-rule-score. We report both numbers in Table 3.

[11] We follow Collins (1999) for head rules.

Parsing Model	Dev (sec 22) F1	Dev (sec 22) EX	Test (sec 23) F1	Test (sec 23) EX
Baseline (1-best)	90.6	39.4	90.2	37.3
log $p(t\|w)$	90.4	38.9	89.9	37.3
+ Web features	91.6	42.5	91.1	40.6
+ Configurational features	91.8	43.8	91.1	40.6
+ Web + Configurational	**92.1**	**44.0**	**91.4**	**41.4**

Table 3: Parsing results for reranking 50-best lists of Berkeley parser (Dev is WSJ section 22 and Test is WSJ section 23, all lengths).

Arg Tag	# Attach	Baseline	This Work	% ER
NN	5725	5387	5429	12.4
NNP	4043	3780	3804	9.1
IN	4026	3416	3490	12.1
DT	3511	3424	3429	5.8
NNS	2504	2319	2348	15.7
JJ	2472	2310	2329	11.7
CD	1845	1739	1738	-0.9
VBD	1705	1571	1580	6.7
RB	1308	1097	1100	1.4
CC	1000	855	854	-0.7
VB	983	940	945	11.6
TO	868	761	776	14.0
VBN	850	776	786	13.5
VBZ	705	633	629	-5.6
PRP	612	603	606	33.3

Table 4: Error reduction for attachments of various child (argument) categories. The columns depict the tag, its total attachments as argument, number of correct ones in baseline (McDonald and Pereira, 2006) and this work, and the relative error reduction. Results are for dependency parsing on the dev set for *iters:5,training-k:1*.

Arg Tag	% Error Redn for Various Parent Tags
NN	IN: 18, NN: 23, VB: 30, NNP:20, VBN: 33
IN	NN: 11, VBD: 11, NNS: 20, VB:18, VBG: 23
NNS	IN: 9, VBD: 29, VBP: 21, VB:15, CC: 33

Table 5: Error reduction for each type of parent attachment for a given child in Table 4.

POS_{head}	POS_{arg}	Example (*head, arg*)
RB	IN	*back → into*
NN	IN	*review → of*
NN	DT	*The ← rate*
NNP	IN	*Regulation → of*
VB	NN	*limit → access*
VBD	NN	*government ← cleared*
NNP	NNP	*Dean ← Inc*
NN	TO	*ability → to*
JJ	IN	*active → for*
NNS	TO	*reasons → to*
IN	NN	*under → pressure*
NNS	IN	*reports → on*
NN	NNP	*Warner ← studio*
NNS	JJ	*few ← plants*

Table 6: The highest-weight features (thresholded at a count of 400) of the affinity schema. We list only the head and argument POS and the direction (arrow from head to arg). We omit features involving punctuation.

tal errors break down by gold head. For example, the 12.1% total error reduction for attachments of an IN argument (which includes PPs as well as complementized SBARs) includes many errors where the gold attachments are to both noun and verb heads. Similarly, for an NN-headed argument, the major corrections are for attachments to noun and verb heads, which includes both object-attachment ambiguities and coordination ambiguities.

We next investigate the features that were given high weight by our learning algorithm (in the constituent parsing case). We first threshold features by a minimum training count of 400 to focus on frequently-firing ones (recall that our features are not bilexical indicators and so are quite a bit more frequent). We then sort them by descending (signed) weight.

Table 6 shows which affinity features received the highest weights, as well as examples of training set attachments for which the feature fired (for concreteness), suppressing both features involving punctuation and the features' count and distance bins. With the standard caveats that interpreting feature weights in isolation is always to be taken for what it is, the first feature (RB→IN) indicates that high counts for an adverb occurring adjacent to a preposition (like *back into the spotlight*) is a useful indicator that the adverb actually modifies that preposition. The second row (NN→IN) indicates that whether a preposition is appropriate to attach to a noun is well captured by how often that preposition follows that noun. The fifth row (VB→NN) indicates that when considering an NP as the object of a verb, it is a good sign if that NP's head frequently occurs immediately following that verb. All of these features essentially state cases where local surface counts are good indi-

POS_{head}	mid-word	POS_{arg}	Example (*head, arg*)
VBN	*this*	IN	*leaned, from*
VB	*this*	IN	*publish, in*
VBG	*him*	IN	*using, as*
VBG	*them*	IN	*joining, in*
VBD	*directly*	IN	*converted, into*
VBD	*held*	IN	*was, in*
VBN	*jointly*	IN	*offered, by*
VBZ	*it*	IN	*passes, in*
VBG	*only*	IN	*consisting, of*
VBN	*primarily*	IN	*developed, for*
VB	*us*	IN	*exempt, from*
VBG	*this*	IN	*using, as*
VBD	*more*	IN	*looked, like*
VB	*here*	IN	*stay, for*
VBN	*themselves*	IN	*launched, into*
VBG	*down*	IN	*lying, on*

Table 7: The highest-weight features (thresholded at a count of 400) of the mid-word schema for a verb head and preposition argument (with head on left of argument).

bfr-word	POS_{head}	POS_{arg}	Example (*head, arg*)
second	NN	IN	*season, in*
The	NN	IN	*role, of*
strong	NN	IN	*background, in*
our	NNS	IN	*representatives, in*
any	NNS	IN	*rights, against*
A	NN	IN	*review, of*
:	NNS	IN	*Results, in*
three	NNS	IN	*years, in*
In	NN	IN	*return, for*
no	NN	IN	*argument, about*
current	NN	IN	*head, of*
no	NNS	IN	*plans, for*
public	NN	IN	*appearance, at*
from	NNS	IN	*sales, of*
net	NN	IN	*revenue, of*
,	NNS	IN	*names, of*
you	NN	IN	*leave, in*
have	NN	IN	*time, for*
some	NN	IN	*money, for*
annual	NNS	IN	*reports, on*

Table 8: The highest-weight features (thresholded at a count of 400) of the before-word schema for a noun head and preposition argument (with head on left of argument).

cators of (possibly non-adjacent) attachments.

A subset of paraphrase features, which in the automatically-extracted case don't really correspond to paraphrases at all, are shown in Table 7. Here we show features for verbal heads and IN arguments. The mid-words m which rank highly are those where the occurrence of hma as an n-gram is a good indicator that a attaches to h (m of course does not have to actually occur in the sentence). Interestingly, the top such features capture exactly the intuition from Nakov and Hearst (2005b), namely that if the verb h and the preposition a occur with a pronoun in between, we have evidence that a attaches to h (it certainly can't attach to the pronoun). However, we also see other indicators that the preposition is selected for by the verb, such as adverbs like *directly*.

As another example of known useful features being learned automatically, Table 8 shows the previous-context-word paraphrase features for a noun head and preposition argument (N $\rightarrow$ IN). Nakov and Hearst (2005b) suggested that the attestation of *be* N IN is a good indicator of attachment to the noun (the IN cannot generally attach to forms of auxiliaries). One such feature occurs on this top list – for the context word *have* – and others occur farther down. We also find their surface marker / punctuation cues of *:* and *,* preceding the noun. However, we additionally find other cues, most notably that if the N IN sequence occurs following a capitalized determiner, it tends to indicate a nominal attachment (in the n-gram, the preposition cannot attach leftward to anything else because of the beginning of the sentence).

In Table 9, we see the top-weight paraphrase features that had a conjunction as a middle-word cue. These features essentially say that if two heads w_1 and w_2 occur in the direct coordination n-gram w_1 *and* w_2, then they are good heads to coordinate (coordination unfortunately looks the same as complementation or modification to a basic dependency model). These features are relevant to a range of coordination ambiguities.

Finally, Table 10 depicts the high-weight, high-count general paraphrase-cue features for arbitrary head and argument categories, with those shown in previous tables suppressed. Again, many interpretable features appear. For example, the top entry (*the* JJ NNS) shows that when considering attaching an adjective a to a noun h, it is a good sign if the

POS_{head}	mid-CC	POS_{arg}	Example (*head, arg*)
NNS	*and*	NNS	*purchases, sales*
VB	*and*	VB	*buy, sell*
NN	*and*	NN	*president, officer*
NN	*and*	NNS	*public, media*
VBD	*and*	VBD	*said, added*
VBZ	*and*	VBZ	*makes, distributes*
JJ	*and*	JJ	*deep, lasting*
IN	*and*	IN	*before, during*
VBD	*and*	RB	*named, now*
VBP	*and*	VBP	*offer, need*

Table 9: The highest-weight features (thresholded at a count of 400) of the mid-word schema where the mid-word was a conjunction. For variety, for a given head-argument POS pair, we only list features corresponding to the *and* conjunction and $h \rightarrow a$ direction.

POS_h	POS_a	mid/bfr-word	Example (*h, a*)
NNS	JJ	b = *the*	*other* ← *things*
NN	NN	m = -	*auto* ← *maker*
VBD	NN	m = *this*	*adopted* → *plan*
NNS	NN	b = *of*	*computer* ← *products*
NN	DT	m = *current*	*the* ← *proposal*
VBG	IN	b = *of*	*going* → *into*
NNS	IN	m = ”	*clusters* → *of*
IN	NN	m = *your*	*In* → *review*
TO	VB	b = *used*	*to* → *ease*
VBZ	NN	m = *that*	*issue* ← *has*
IN	NNS	m = *two*	*than* → *minutes*
IN	NN	b = *used*	*as* → *tool*
IN	VBD	m = *they*	*since* → *were*
VB	TO	b = *will*	*fail* → *to*

Table 10: The high-weight high-count (thresholded at a count of 2000) general features of the mid and before paraphrase schema (examples show head and arg in linear order with arrow from head to arg).

trigram *the a h* is frequent – in that trigram, the adjective attaches to the noun. The second entry (NN - NN) shows that one noun is a good modifier of another if they frequently appear together hyphenated (another punctuation-based cue mentioned in previous work on noun bracketing, see Nakov and Hearst (2005a)). While they were motivated on separate grounds, these features can also compensate for inapplicability of the affinity features. For example, the third entry (VBD *this* NN) is a case where even if the head (a VBD like *adopted*) actually selects strongly for the argument (a NN like *plan*), the bigram *adopted plan* may not be as frequent as expected, because it requires a determiner in its minimal analogous form *adopted the plan*.

6 Conclusion

Web features are a way to bring evidence from a large unlabeled corpus to bear on hard disambiguation decisions that are not easily resolvable based on limited parser training data. Our approach allows revealing features to be mined for the entire range of attachment types and then aggregated and balanced in a full parsing setting. Our results show that these web features resolve ambiguities not correctly handled by current state-of-the-art systems.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful suggestions. This research is supported by BBN under DARPA contract HR0011-06-C-0022.

References

M. Atterer and H. Schutze. 2007. Prepositional phrase attachment without oracles. *Computational Linguistics*, 33(4):469476.

Thorsten Brants and Alex Franz. 2006. The Google Web 1T 5-gram corpus version 1.1. *LDC2006T13*.

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

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

Michael Collins. 1999. Head-Driven Statistical Models for Natural Language Parsing. *Ph.D. thesis, University of Pennsylvania, Philadelphia*.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of ICML*.

Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL*.

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

Adam Kilgarriff. 2007. Googleology is bad science. *Computational Linguistics*, 33(1).

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*.

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

Mirella Lapata and Frank Keller. 2004. The Web as a baseline: Evaluating the performance of unsupervised Web-based models for a range of NLP tasks. In *Proceedings of HLT-NAACL*.

M. Lauer. 1995. Corpus statistics meet the noun compound: some empirical results. In *Proceedings of ACL*.

André F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of ACL-IJCNLP*.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*.

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

Preslav Nakov and Marti Hearst. 2005a. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *Proceedings of CoNLL*.

Preslav Nakov and Marti Hearst. 2005b. Using the web as an implicit training set: Application to structural ambiguity resolution. In *Proceedings of EMNLP*.

Preslav Nakov and Marti Hearst. 2008. Solving relational similarity problems using the web as a corpus. In *Proceedings of ACL*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of COLING-ACL*.

Emily Pitler, Shane Bergsma, Dekang Lin, , and Kenneth Church. 2010. Using web-scale n-grams to improve base NP parsing performance. In *Proceedings of COLING*.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP*.

David A. Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of EMNLP*.

David Vadas and James R. Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *Proceedings of ACL*.

Martin Volk. 2001. Exploiting the WWW as a corpus to resolve PP attachment ambiguities. In *Proceedings of Corpus Linguistics*.

Alexander Yates, Stefan Schoenmackers, and Oren Etzioni. 2006. Detecting parser errors using web-based semantic filters. In *Proceedings of EMNLP*.

The impact of language models and loss functions on repair disfluency detection

Simon Zwarts and Mark Johnson
Centre for Language Technology
Macquarie University
{simon.zwarts|mark.johnson|}@mq.edu.au

Abstract

Unrehearsed spoken language often contains disfluencies. In order to correctly interpret a spoken utterance, any such disfluencies must be identified and removed or otherwise dealt with. Operating on transcripts of speech which contain disfluencies, we study the effect of language model and loss function on the performance of a linear reranker that rescores the 25-best output of a noisy-channel model. We show that language models trained on large amounts of non-speech data improve performance more than a language model trained on a more modest amount of speech data, and that optimising f-score rather than log loss improves disfluency detection performance.

Our approach uses a log-linear reranker, operating on the top n analyses of a noisy channel model. We use large language models, introduce new features into this reranker and examine different optimisation strategies. We obtain a disfluency detection f-scores of 0.838 which improves upon the current state-of-the-art.

1 Introduction

Most spontaneous speech contains disfluencies such as partial words, filled pauses (e.g., "uh", "um", "huh"), explicit editing terms (e.g., "I mean"), parenthetical asides and repairs. Of these, repairs pose particularly difficult problems for parsing and related Natural Language Processing (NLP) tasks. This paper presents a model of disfluency detection based on the noisy channel framework, which specifically targets the repair disfluencies. By combining language models and using an appropriate loss function in a log-linear reranker we are able to achieve f-scores which are higher than previously reported.

Often in natural language processing algorithms, more data is more important than better algorithms (Brill and Banko, 2001). It is this insight that drives the first part of the work described in this paper. This paper investigates how we can use language models trained on large corpora to increase repair detection accuracy performance.

There are three main innovations in this paper. First, we investigate the use of a variety of language models trained from text or speech corpora of various genres and sizes. The largest available language models are based on written text: we investigate the effect of written text language models as opposed to language models based on speech transcripts. Second, we develop a new set of reranker features explicitly designed to capture important properties of speech repairs. Many of these features are lexically grounded and provide a large performance increase. Third, we utilise a loss function, approximate expected f-score, that explicitly targets the asymmetric evaluation metrics used in the disfluency detection task. We explain how to optimise this loss function, and show that this leads to a marked improvement in disfluency detection. This is consistent with Jansche (2005) and Smith and Eisner (2006), who observed similar improvements when using approximate f-score loss for other problems. Similarly we introduce a loss function based on the edit-f-score in our domain.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 703–711,
Portland, Oregon, June 19-24, 2011.

Together, these three improvements are enough to boost detection performance to a higher f-score than previously reported in literature. Zhang et al. (2006) investigate the use of 'ultra large feature spaces' as an aid for disfluency detection. Using over 19 million features, they report a final f-score in this task of 0.820. Operating on the same body of text (Switchboard), our work leads to an f-score of 0.838, this is a 9% relative improvement in residual f-score.

The remainder of this paper is structured as follows. First in Section 2 we describe related work. Then in Section 3 we present some background on disfluencies and their structure. Section 4 describes appropriate evaluation techniques. In Section 5 we describe the noisy channel model we are using. The next three sections describe the new additions: Section 6 describe the corpora used for language models, Section 7 describes features used in the log-linear model employed by the reranker and Section 8 describes appropriate loss functions which are critical for our approach. We evaluate the new model in Section 9. Section 10 draws up a conclusion.

2 Related work

A number of different techniques have been proposed for automatic disfluency detection. Schuler et al. (2010) propose a Hierarchical Hidden Markov Model approach; this is a statistical approach which builds up a syntactic analysis of the sentence and marks those subtrees which it considers to be made up of disfluent material. Although they are interested not only in disfluency but also a syntactic analysis of the utterance, including the disfluencies being analysed, their model's final f-score for disfluency detection is lower than that of other models.

Snover et al. (2004) investigate the use of purely lexical features combined with part-of-speech tags to detect disfluencies. This approach is compared to approaches which use primarily prosodic cues, and appears to perform equally well. However, the authors note that this model finds it difficult to identify disfluencies which by themselves are very fluent. As we will see later, the individual components of a disfluency do not have to be disfluent by themselves. This can occur when a speaker edits her speech for meaning-related reasons, rather than errors that arise from performance. The edit repairs which are the focus of our work typically have this characteristic.

Noisy channel models have done well on the disfluency detection task in the past; the work of Johnson and Charniak (2004) first explores such an approach. Johnson et al. (2004) adds some handwritten rules to the noisy channel model and use a maximum entropy approach, providing results comparable to Zhang et al. (2006), which are state-of-the art results.

Kahn et al. (2005) investigated the role of prosodic cues in disfluency detection, although the main focus of their work was accurately recovering and parsing a fluent version of the sentence. They report a 0.782 f-score for disfluency detection.

3 Speech Disfluencies

We follow the definitions of Shriberg (1994) regarding speech disfluencies. She identifies and defines three distinct parts of a speech disfluency, referred to as the *reparandum*, the *interregnum* and the *repair*. Consider the following utterance:

$$\overbrace{\underbrace{\textit{I want a flight}}_{}\ \textit{to Boston}}^{\text{reparandum}},\ \underbrace{\textit{uh, I mean}}_{\text{interregnum}}\ \underbrace{\textit{to Denver}}_{\text{repair}}\ \textit{on Friday} \tag{1}$$

The reparandum *to Boston* is the part of the utterance that is 'edited out'; the interregnum *uh, I mean* is a filled pause, which need not always be present; and the repair *to Denver* replaces the reparandum.

Shriberg and Stolcke (1998) studied the location and distribution of repairs in the Switchboard corpus (Godfrey and Holliman, 1997), the primary corpus for speech disfluency research, but did not propose an actual model of repairs. They found that the overall distribution of speech disfluencies in a large corpus can be fit well by a model that uses only information on a very local level. Our model, as explained in section 5, follows from this observation.

As our domain of interest we use the Switchboard corpus. This is a large corpus consisting of transcribed telephone conversations between two partners. In the Treebank III (Marcus et al., 1999) corpus there is annotation available for the Switchboard corpus, which annotates which parts of utterances

are in a reparandum, interregnum or repair.

4 Evaluation metrics for disfluency detection systems

Disfluency detection systems like the one described here identify a subset of the word tokens in each transcribed utterance as "edited" or disfluent. Perhaps the simplest way to evaluate such systems is to calculate the accuracy of labelling they produce, i.e., the fraction of words that are correctly labelled (i.e., either "edited" or "not edited"). However, as Charniak and Johnson (2001) observe, because only 5.9% of words in the Switchboard corpus are "edited", the trivial baseline classifier which assigns all words the "not edited" label achieves a labelling accuracy of 94.1%.

Because the labelling accuracy of the trivial baseline classifier is so high, it is standard to use a different evaluation metric that focuses more on the detection of "edited" words. We follow Charniak and Johnson (2001) and report the f-score of our disfluency detection system. The f-score f is:

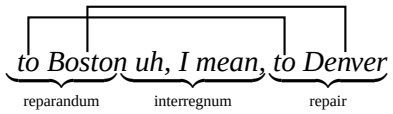
$$f = \frac{2c}{g+e} \quad (2)$$

where g is the number of "edited" words in the gold test corpus, e is the number of "edited" words proposed by the system on that corpus, and c is the number of the "edited" words proposed by the system that are in fact correct. A perfect classifier which correctly labels every word achieves an f-score of 1, while the trivial baseline classifiers which label every word as "edited" or "not edited" respectively achieve a very low f-score.

Informally, the f-score metric focuses more on the "edited" words than it does on the "not edited" words. As we will see in section 8, this has implications for the choice of loss function used to train the classifier.

5 Noisy Channel Model

Following Johnson and Charniak (2004), we use a noisy channel model to propose a 25-best list of possible speech disfluency analyses. The choice of this model is driven by the observation that the repairs frequently seem to be a "rough copy" of the reparandum, often incorporating the same or very similar words in roughly the same word order. That is, they seem to involve "crossed" dependencies between the reparandum and the repair. Example (3) shows the crossing dependencies. As this example also shows, the repair often contains many of the same words that appear in the reparandum. In fact, in our Switchboard training corpus we found that 62reparandum also appeared in the associated repair,



(3)

5.1 Informal Description

Given an observed sentence Y we wish to find the most likely source sentence $\hat{X}$, where

$$\hat{X} = \operatorname*{argmax}_X P(Y|X)P(X) \quad (4)$$

In our model the unobserved X is a substring of the complete utterance Y.

Noisy-channel models are used in a similar way in statistical speech recognition and machine translation. The language model assigns a probability $P(X)$ to the string X, which is a substring of the observed utterance Y. The channel model $P(Y|X)$ generates the utterance Y, which is a potentially disfluent version of the source sentence X. A repair can potentially begin before any word of X. When a repair has begun, the channel model incrementally processes the succeeding words from the start of the repair. Before each succeeding word either the repair can end or else a sequence of words can be inserted in the reparandum. At the end of each repair, a (possibly null) interregnum is appended to the reparandum.

We will look at these two components in the next two Sections in more detail.

5.2 Language Model

Informally, the task of language model component of the noisy channel model is to assess fluency of the sentence with disfluency removed. Ideally we would like to have a model which assigns a very high probability to disfluency-free utterances and a lower probability to utterances still containing disfluencies. For computational complexity reasons, as described in the next section, inside the noisy channel model we use a bigram language model. This

bigram language model is trained on the fluent version of the Switchboard corpus (training section).

We realise that a bigram model might not be able to capture more complex language behaviour. This motivates our investigation of a range of additional language models, which are used to define features used in the log-linear reranker as described below.

5.3 Channel Model

The intuition motivating the channel model design is that the words inserted into the reparandum are very closely related to those in the repair. Indeed, in our training data we find that 62% of the words in the reparandum are exact copies of words in the repair; this identity is strong evidence of a repair. The channel model is designed so that exact copy reparandum words will have high probability.

Because these repair structures can involve an unbounded number of crossed dependencies, they cannot be described by a context-free or finite-state grammar. This motivates the use of a more expressive formalism to describe these repair structures.

We assume that X is a substring of Y, i.e., that the source sentence can be obtained by deleting words from Y, so for a fixed observed utterance Y there are only a finite number of possible source sentences. However, the number of possible source sentences, X, grows exponentially with the length of Y, so exhaustive search is infeasible. Tree Adjoining Grammars (TAG) provide a systematic way of formalising the channel model, and their polynomial-time dynamic programming parsing algorithms can be used to search for likely repairs, at least when used with simple language models like a bigram language model. In this paper we first identify the 25 most likely analyses of each sentence using the TAG channel model together with a bigram language model.

Further details of the noisy channel model can be found in Johnson and Charniak (2004).

5.4 Reranker

To improve performance over the standard noisy channel model we use a reranker, as previously suggest by Johnson and Charniak (2004). We rerank a 25-best list of analyses. This choice is motivated by an oracle experiment we performed, probing for the location of the best analysis in a 100-best list. This experiment shows that in 99.5% of the cases the best analysis is located within the first 25, and indicates that an f-score of 0.958 should be achievable as the upper bound on a model using the first 25 best analyses. We therefore use the top 25 analyses from the noisy channel model in the remainder of this paper and use a reranker to choose the most suitable candidate among these.

6 Corpora for language modelling

We would like to use additional data to model the fluent part of spoken language. However, the Switchboard corpus is one of the largest widely-available disfluency-annotated speech corpora. It is reasonable to believe that for effective disfluency detection Switchboard is not large enough and more text can provide better analyses. Schwartz et al. (1994), although not focusing on disfluency detection, show that using written language data for modelling spoken language can improve performance. We turn to three other bodies of text and investigate the use of these corpora for our task, disfluency detection. We will describe these corpora in detail here.

The predictions made by several language models are likely to be strongly correlated, even if the language models are trained on different corpora. This motivates the choice for log-linear learners, which are built to handle features which are not necessarily independent. We incorporate information from the external language models by defining a reranker feature for each external language model. The value of this feature is the log probability assigned by the language model to the candidate underlying fluent substring X

For each of our corpora (including Switchboard) we built a 4-gram language model with Kneser-Ney smoothing (Kneser and Ney, 1995). For each analysis we calculate the probability under that language model for the candidate underlying fluent substring X. We use this log probability as a feature in the reranker. We use the SRILM toolkit (Stolcke, 2002) both for estimating the model from the training corpus as well as for computing the probabilities of the underlying fluent sentences X of the different analysis.

As previously described, **Switchboard** is our pri-

mary corpus for our model. The language model part of the noisy channel model already uses a bigram language model based on Switchboard, but in the reranker we would like to also use 4-grams for reranking. Directly using Switchboard to build a 4-gram language model is slightly problematic. When we use the training data of Switchboard both for language fluency prediction and the same training data also for the loss function, the reranker will overestimate the weight associated with the feature derived from the Switchboard language model, since the fluent sentence itself is part of the language model training data. We solve this by dividing the Switchboard training data into 20 folds. For each fold we use the 19 other folds to construct a language model and then score the utterance in this fold with that language model.

The largest widely-available corpus for language modelling is the **Web 1T 5-gram** corpus (Brants and Franz, 2006). This data set, collected by Google Inc., contains English word n-grams and their observed frequency counts. Frequency counts are produced from this billion-token corpus of web text. Because of the noise[1] present in this corpus there is an ongoing debate in the scientific community of the use of this corpus for serious language modelling.

The **Gigaword** Corpus (Graff and Cieri, 2003) is a large body of newswire text. The corpus contains $1.6 \cdot 10^9$ tokens, however fluent newswire text is not necessarily of the same domain as disfluency removed speech.

The **Fisher** corpora Part I (David et al., 2004) and Part II (David et al., 2005) are large bodies of transcribed text. Unlike Switchboard there is no disfluency annotation available for Fisher. Together the two Fisher corpora consist of $2.2 \cdot 10^7$ tokens.

7 Features

The log-linear reranker, which rescores the 25-best lists produced by the noisy-channel model, can also include additional features besides the noisy-channel log probabilities. As we show below, these additional features can make a substantial improvement to disfluency detection performance. Our reranker incorporates two kinds of features. The first are log-probabilities of various scores computed by the noisy-channel model and the external language models. We only include features which occur at least 5 times in our training data.

The noisy channel and language model features consist of:

1. LMP: 4 features indicating the probabilities of the underlying fluent sentences under the language models, as discussed in the previous section.

2. NCLogP: The Log Probability of the entire noisy channel model. Since by itself the noisy channel model is already doing a very good job, we do not want this information to be lost.

3. LogFom: This feature is the log of the "figure of merit" used to guide search in the noisy channel model when it is producing the 25-best list for the reranker. The log figure of merit is the sum of the log language model probability and the log channel model probability plus 1.5 times the number of edits in the sentence. This feature is redundant, i.e., it is a linear combination of other features available to the reranker model: we include it here so the reranker has direct access to all of the features used by the noisy channel model.

4. NCTransOdd: We include as a feature parts of the noisy channel model itself, i.e. the channel model probability. We do this so that the task to choosing appropriate weights of the channel model and language model can be moved from the noisy channel model to the log-linear optimisation algorithm.

The boolean indicator features consist of the following 3 groups of features operating on words and their edit status; the latter indicated by one of three possible flags: _ when the word is not part of a disfluency or E when it is part of the reparandum or I when it is part of the interregnum.

1. CopyFlags_X_Y: When there is an exact copy in the input text of length X ($1 \leq X \leq 3$) and the gap between the copies is Y ($0 \leq Y \leq 3$) this feature is the sequence of flags covering the two copies. Example: `CopyFlags_1_0 (E`

[1] We do not mean speech disfluencies here, but noise in web-text; web-text is often poorly written and unedited text.

_) records a feature when two identical words are present, directly consecutive and the first one is part of a disfluency (**E**dited) while the second one is not. There are 745 different instances of these features.

2. WordsFlags_L_n_R: This feature records the immediate area around an n-gram ($n \leq 3$). L denotes how many flags to the left and R ($0 \leq R \leq 1$) how many to the right are includes in this feature (Both L and R range over 0 and 1). Example: `WordsFlags_1_1_0 (need _)` is a feature that fires when a fluent word is followed by the word 'need' (one flag to the left, none to the right). There are 256808 of these features present.

3. SentenceEdgeFlags_B_L: This feature indicates the location of a disfluency in an utterance. The Boolean B indicates whether this features records sentence initial or sentence final behaviour, L ($1 \leq L \leq 3$) records the length of the flags. Example `SentenceEdgeFlags_1_1 (I)` is a feature recording whether a sentence ends on an interregnum. There are 22 of these features present.

We give the following analysis as an example:

but E *but _ that _ does _ n't _ work _*

The language model features are the probability calculated over the fluent part. NCLogP, LogFom and NCTransOdd are present with their associated value. The following binary flags are present:
`CopyFlags_1_0 (E _)`
`WordsFlags:0:1:0 (but E)`
`WordsFlags:0:1:0 (but _)`
`WordsFlags:1:1:0 (E but _)`
`WordsFlags:1:1:0 (_ that _)`
`WordsFlags:0:2:0 (but E but _)` etc.[2]
`SentenceEdgeFlags:0:1 (E)`
`SentenceEdgeFlags:0:2 (E _)`
`SentenceEdgeFlags:0:3 (E _ _)`

These three kinds of boolean indicator features together constitute the *extended feature set*.

[2] An exhaustive list here would be too verbose.

8 Loss functions for reranker training

We formalise the reranker training procedure as follows. We are given a training corpus T containing information about n possibly disfluent sentences. For the ith sentence T specifies the sequence of words x_i, a set $\mathcal{Y}_i$ of 25-best candidate "edited" labellings produced by the noisy channel model, as well as the correct "edited" labelling $y_i^\star \in \mathcal{Y}_i$.[3]

We are also given a vector $\mathbf{f} = (f_1, \ldots, f_m)$ of *feature functions*, where each f_j maps a word sequence x and an "edit" labelling y for x to a real value $f_j(x, y)$. Abusing notation somewhat, we write $\mathbf{f}(x, y) = (f_1(x, y), \ldots, f_m(x, y))$. We interpret a vector $\mathbf{w} = (w_1, \ldots, w_m)$ of *feature weights* as defining a conditional probability distribution over a candidate set $\mathcal{Y}$ of "edited" labellings for a string x as follows:

$$\mathrm{P}_{\mathbf{w}}(y \mid x, \mathcal{Y}) = \frac{\exp(\mathbf{w} \cdot \mathbf{f}(x, y))}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w} \cdot \mathbf{f}(x, y'))}$$

We estimate the feature weights $\mathbf{w}$ from the training data T by finding a feature weight vector $\widehat{\mathbf{w}}$ that optimises a regularised objective function:

$$\widehat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}} L_T(\mathbf{w}) + \alpha \sum_{j=1}^{m} w_j^2$$

Here α is the *regulariser weight* and L_T is a *loss function*. We investigate two different loss functions in this paper. $LogLoss$ is the negative log conditional likelihood of the training data:

$$LogLoss_T(\mathbf{w}) = \sum_{i=1}^{m} -\log \mathrm{P}(y_i^\star \mid x_i, \mathcal{Y}_i)$$

Optimising $LogLoss$ finds the $\widehat{\mathbf{w}}$ that define (regularised) conditional Maximum Entropy models.

It turns out that optimising $LogLoss$ yields suboptimal weight vectors $\widehat{\mathbf{w}}$ here. $LogLoss$ is a symmetric loss function (i.e., each mistake is equally weighted), while our f-score evaluation metric weights "edited" labels more highly, as explained in section 4. Because our data is so skewed (i.e., "edited" words are comparatively infrequent), we

[3] In the situation where the true "edited" labelling does not appear in the 25-best list $\mathcal{Y}_i$ produced by the noisy-channel model, we choose $y_i^\star$ to be a labelling in $\mathcal{Y}_i$ closest to the true labelling.

can improve performance by using an asymmetric loss function.

Inspired by our evaluation metric, we devised an *approximate expected f-score loss function* $FLoss$.

$$FLoss_T(\mathbf{w}) \;=\; 1 - \frac{2\mathrm{E}_{\mathbf{w}}[c]}{g + \mathrm{E}_{\mathbf{w}}[e]}$$

This approximation assumes that the expectations approximately distribute over the division: see Jansche (2005) and Smith and Eisner (2006) for other approximations to expected f-score and methods for optimising them. We experimented with other asymmetric loss functions (e.g., the expected error rate) and found that they gave very similar results.

An advantage of $FLoss$ is that it and its derivatives with respect to $\mathbf{w}$ (which are required for numerical optimisation) are easy to calculate exactly. For example, the expected number of correct "edited" words is:

$$\begin{aligned}\mathrm{E}_{\mathbf{w}}[c] &= \sum_{i=1}^{n} \mathrm{E}_{\mathbf{w}}[c_{y_i^\star} \mid \mathcal{Y}_i], \text{ where:} \\ \mathrm{E}_{\mathbf{w}}[c_{y_i^\star} \mid \mathcal{Y}_i] &= \sum_{y \in \mathcal{Y}_i} c_{y_i^\star}(y)\, \mathrm{P}_{\mathbf{w}}(y \mid x_i, \mathcal{Y}_i)\end{aligned}$$

and $c_{y^\star}(y)$ is the number of correct "edited" labels in y given the gold labelling $y^\star$. The derivatives of $FLoss$ are:

$$\frac{\partial FLoss_T}{\partial w_j}(\mathbf{w}) \;=\; \frac{1}{g + \mathrm{E}_{\mathbf{w}}[e]} \left(FLoss_T(\mathbf{w}) \frac{\partial \mathrm{E}_{\mathbf{w}}[e]}{\partial w_j} - 2 \frac{\partial \mathrm{E}_{\mathbf{w}}[c]}{\partial w_j} \right)$$

where:

$$\frac{\partial \mathrm{E}_{\mathbf{w}}[c]}{\partial w_j} \;=\; \sum_{i=1}^{n} \frac{\partial \mathrm{E}_{\mathbf{w}}[c_{y_i^\star} \mid x_i, \mathcal{Y}_i]}{\partial w_j}$$

$$\frac{\partial \mathrm{E}_{\mathbf{w}}[c_{y^\star} \mid x, \mathcal{Y}]}{\partial w_j} \;=\; \mathrm{E}_{\mathbf{w}}[f_j c_{y^\star} \mid x, \mathcal{Y}] - \mathrm{E}_{\mathbf{w}}[f_j \mid x, \mathcal{Y}]\, \mathrm{E}_{\mathbf{w}}[c_{y^\star} \mid x, \mathcal{Y}].$$

$\partial \mathrm{E}[e] / \partial w_j$ is given by a similar formula.

9 Results

We follow Charniak and Johnson (2001) and split the corpus into main training data, held-out training data and test data as follows: main training consisted of all sw[23]$*$.dps files, held-out training consisted of all sw4[5-9]$*$.dps files and test consisted of all sw4[0-1]$*$.dps files. However, we follow (Johnson and Charniak, 2004) in deleting all partial words and punctuation from the training and test data (they argued that this is more realistic in a speech processing application).

Table 1 shows the results for the different models on held-out data. To avoid over-fitting on the test data, we present the f-scores over held-out training data instead of test data. We used the held-out data to select the best-performing set of reranker features, which consisted of features for all of the language models plus the extended (i.e., indicator) features, and used this model to analyse the test data. The f-score of this model on test data was 0.838. In this table, the set of *Extended Features* is defined as all the boolean features as described in Section 7.

We first observe that adding different external language models does increase the final score. The difference between the external language models is relatively small, although the differences in choice are several orders of magnitude. Despite the putative noise in the corpus, a language model built on Google's Web1T data seems to perform very well. Only the model where Switchboard 4-grams are used scores slightly lower, we explain this because the internal bigram model of the noisy channel model is already trained on Switchboard and so this model adds less new information to the reranker than the other models do.

Including additional features to describe the problem space is very productive. Indeed the best performing model is the model which has all extended features and all language model features. The differences among the different language models when extended features are present are relatively small. We assume that much of the information expressed in the language models overlaps with the lexical features.

We find that using a loss function related to our evaluation metric, rather than optimising $LogLoss$, consistently improves edit-word f-score. The standard $LogLoss$ function, which estimates the "maximum entropy" model, consistently performs worse than the loss function minimising expected errors.

The best performing model (Base + Ext. Feat. + All LM, using expected f-score loss) scores an f-score of **0.838** on **test data**. The results as indicated by the f-score outperform state-of-the-art models re-

Model	F-score	
Base (noisy channel, no reranking)	0.756	
Model	**log loss**	**expected f-score loss**
Base + Switchboard	0.776	0.791
Base + Fisher	0.771	0.797
Base + Gigaword	0.777	0.797
Base + Web1T	0.781	0.798
Base + Ext. Feat.	0.824	0.827
Base + Ext. Feat. + Switchboard	0.827	0.828
Base + Ext. Feat. + Fisher	0.841	0.856
Base + Ext. Feat. + Gigaword	0.843	0.852
Base + Ext. Feat. + Web1T	0.843	0.850
Base + Ext. Feat. + All LM	0.841	**0.857**

Table 1: Edited word detection f-score on held-out data for a variety of language models and loss functions

ported in literature operating on identical data, even though we use vastly less features than other do.

10 Conclusion and Future work

We have described a disfluency detection algorithm which we believe improves upon current state-of-the-art competitors. This model is based on a noisy channel model which scores putative analyses with a language model; its channel model is inspired by the observation that reparandum and repair are often very similar. As Johnson and Charniak (2004) noted, although this model performs well, a log-linear reranker can be used to increase performance.

We built language models from a variety of speech and non-speech corpora, and examine the effect they have on disfluency detection. We use language models derived from different larger corpora effectively in a maximum reranker setting. We show that the actual choice for a language model seems to be less relevant and newswire text can be used equally well for modelling fluent speech.

We describe different features to improve disfluency detection even further. Especially these features seem to boost performance significantly.

Finally we investigate the effect of different loss functions. We observe that using a loss function directly optimising our interest yields a performance increase which is at least at large as the effect of using very large language models.

We obtained an f-score which outperforms other models reported in literature operating on identical data, even though we use vastly fewer features than others do.

Acknowledgements

This work was supported was supported under Australian Research Council's Discovery Projects funding scheme (project number DP110102593) and by the Australian Research Council as part of the Thinking Head Project the Thinking Head Project, ARC/NHMRC Special Research Initiative Grant # TS0669874. We thank the anonymous reviewers for their helpful comments.

References

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram Version 1. Published by *Linguistic Data Consortium*, Philadelphia.

Erik Brill and Michele Banko. 2001. Mitigating the Paucity-of-Data Problem: Exploring the Effect of Training Corpus Size on Classifier Performance for Natural Language Processing. In *Proceedings of the First International Conference on Human Language Technology Research*.

Eugene Charniak and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 118–126.

Christopher Cieri David, David Miller, and Kevin Walker. 2004. Fisher English Training Speech Part 1 Transcripts. Published by *Linguistic Data Consortium*, Philadelphia.

Christopher Cieri David, David Miller, and Kevin Walker. 2005. Fisher English Training Speech Part 2 Transcripts. Published by *Linguistic Data Consortium*, Philadelphia.

John J. Godfrey and Edward Holliman. 1997. Switchboard-1 Release 2. Published by *Linguistic Data Consortium, Philadelphia*.

David Graff and Christopher Cieri. 2003. English gigaword. Published by *Linguistic Data Consortium*, Philadelphia.

Martin Jansche. 2005. Maximum Expected F-Measure Training of Logistic Regression Models. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 692–699, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Mark Johnson and Eugene Charniak. 2004. A TAG-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 33–39.

Mark Johnson, Eugene Charniak, and Matthew Lease. 2004. An Improved Model for Recognizing Disfluencies in Conversational Speech. In *Proceedings of the Rich Transcription Fall Workshop*.

Jeremy G. Kahn, Matthew Lease, Eugene Charniak, Mark Johnson, and Mari Ostendorf. 2005. Effective Use of Prosody in Parsing Conversational Speech. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 233–240, Vancouver, British Columbia, Canada.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 181–184.

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

William Schuler, Samir AbdelRahman, Tim Miller, and Lane Schwartz. 2010. Broad-Coverage Parsing using Human-Like Memory Constraints. *Computational Linguistics*, 36(1):1–30.

Richard Schwartz, Long Nguyen, Francis Kubala, George Chou, George Zavaliagkos, and John Makhoul. 1994. On Using Written Language Training Data for Spoken Language Modeling. In *Proceedings of the Human Language Technology Workshop*, pages 94–98.

Elizabeth Shriberg and Andreas Stolcke. 1998. How far do speakers back up in repairs? A quantitative model. In *Proceedings of the International Conference on Spoken Language Processing*, pages 2183–2186.

Elizabeth Shriberg. 1994. *Preliminaries to a Theory of Speech Disuencies*. Ph.D. thesis, University of California, Berkeley.

David A. Smith and Jason Eisner. 2006. Minimum Risk Annealing for Training Log-Linear Models. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 787–794.

Matthew Snover, Bonnie Dorr, and Richard Schwartz. 2004. A Lexically-Driven Algorithm for Disfluency Detection. In *Proceedings of Human Language Technologies and North American Association for Computational Linguistics*, pages 157–160.

Andreas Stolcke. 2002. SRILM - An Extensible Language Modeling Toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 901–904.

Qi Zhang, Fuliang Weng, and Zhe Feng. 2006. A progressive feature selection algorithm for ultra large feature spaces. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 561–568.

Learning Sub-Word Units for Open Vocabulary Speech Recognition

Carolina Parada[1], **Mark Dredze**[1], **Abhinav Sethy**[2], **and Ariya Rastrow**[1]
[1]Human Language Technology Center of Excellence, Johns Hopkins University
3400 N Charles Street, Baltimore, MD, USA
carolinap@jhu.edu, mdredze@cs.jhu.edu, ariya@jhu.edu
[2]IBM T.J. Watson Research Center, Yorktown Heights, NY, USA
asethy@us.ibm.com

Abstract

Large vocabulary speech recognition systems fail to recognize words beyond their vocabulary, many of which are information rich terms, like named entities or foreign words. Hybrid word/sub-word systems solve this problem by adding sub-word units to large vocabulary word based systems; new words can then be represented by combinations of sub-word units. Previous work heuristically created the sub-word lexicon from phonetic representations of text using simple statistics to select common phone sequences. We propose a probabilistic model to *learn* the sub-word lexicon optimized for a given task. We consider the task of out of vocabulary (OOV) word detection, which relies on output from a hybrid model. A hybrid model with our learned sub-word lexicon reduces error by 6.3% and 7.6% (absolute) at a 5% false alarm rate on an English Broadcast News and MIT Lectures task respectively.

1 Introduction

Most automatic speech recognition systems operate with a large but limited vocabulary, finding the most likely words in the vocabulary for the given acoustic signal. While large vocabulary continuous speech recognition (LVCSR) systems produce high quality transcripts, they fail to recognize out of vocabulary (OOV) words. Unfortunately, OOVs are often information rich nouns, such as named entities and foreign words, and mis-recognizing them can have a disproportionate impact on transcript coherence.

Hybrid word/sub-word recognizers can produce a sequence of *sub-word units* in place of OOV words. Ideally, the recognizer outputs a complete word for in-vocabulary (IV) utterances, and sub-word units for OOVs. Consider the word "Slobodan", the given name of the former president of Serbia. As an uncommon English word, it is unlikely to be in the vocabulary of an English recognizer. While a LVCSR system would output the closest known words (e.x. "slow it dawn"), a hybrid system could output a sequence of multi-phoneme units: `s_l_ow`, `b_ax`, `d_ae_n`. The latter is more useful for automatically recovering the word's orthographic form, identifying that an OOV was spoken, or improving performance of a spoken term detection system with OOV queries. In fact, hybrid systems have improved OOV spoken term detection (Mamou et al., 2007; Parada et al., 2009), achieved better phone error rates, especially in OOV regions (Rastrow et al., 2009b), and obtained state-of-the-art performance for OOV detection (Parada et al., 2010).

Hybrid recognizers vary in a number of ways: sub-word unit type: variable-length phoneme units (Rastrow et al., 2009a; Bazzi and Glass, 2001) or joint letter sound sub-words (Bisani and Ney, 2005); unit creation: data-driven or linguistically motivated (Choueiter, 2009); and how they are incorporated in LVCSR systems: hierarchical (Bazzi, 2002) or flat models (Bisani and Ney, 2005).

In this work, we consider how to optimally create sub-word units for a hybrid system. These units are variable-length phoneme sequences, although in principle our work can be use for other unit types. Previous methods for creating the sub-word lexi-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 712–721,
Portland, Oregon, June 19-24, 2011.

con have relied on simple statistics computed from the phonetic representation of text (Rastrow et al., 2009a). These units typically represent the most frequent phoneme sequences in English words. However, it isn't clear why these units would produce the best hybrid output. Instead, we introduce a probabilistic model for *learning* the optimal units for a given task. Our model learns a segmentation of a text corpus given some side information: a mapping between the vocabulary and a label set; learned units are predictive of class labels.

In this paper, we learn sub-word units optimized for OOV detection. OOV detection aims to identify regions in the LVCSR output where OOVs were uttered. Towards this goal, we are interested in selecting units such that the recognizer outputs them only for OOV regions while prefering to output a complete word for in-vocabulary regions. Our approach yields improvements over state-of-the-art results.

We begin by presenting our log-linear model for learning sub-word units with a simple but effective inference procedure. After reviewing existing OOV detection approaches, we detail how the learned units are integrated into a hybrid speech recognition system. We show improvements in OOV detection, and evaluate impact on phone error rates.

2 Learning Sub-Word Units

Given raw text, our objective is to produce a lexicon of sub-word units that can be used by a hybrid system for open vocabulary speech recognition. Rather than relying on the text alone, we also utilize side information: a mapping of words to classes so we can optimize learning for a specific task.

The provided mapping assigns labels Y to the corpus. We maximize the probability of the observed labeling sequence Y given the text W: $P(Y|W)$. We assume there is a latent segmentation S of this corpus which impacts Y. The complete data likelihood becomes: $P(Y|W) = \sum_S P(Y, S|W)$ during training. Since we are maximizing the observed Y, segmentation S must discriminate between different possible labels.

We learn variable-length multi-phone units by segmenting the phonetic representation of each word in the corpus. Resulting segments form the sub-word lexicon.[1] Learning input includes a list of words to segment taken from raw text, a mapping between words and classes (side information indicating whether token is IV or OOV), a pronunciation dictionary D, and a letter to sound model (L2S), such as the one described in Chen (2003). The corpus W is the list of types (unique words) in the raw text input. This forces each word to have a unique segmentation, shared by all common tokens. Words are converted into phonetic representations according to their most likely dictionary pronunciation; non-dictionary words use the L2S model.[2]

2.1 Model

Inspired by the morphological segmentation model of Poon et al. (2009), we assume $P(Y, S|W)$ is a log-linear model parameterized by Λ:

$$P_\Lambda(Y, S|W) = \frac{1}{Z(W)} u_\Lambda(Y, S, W) \qquad (1)$$

where $u_\Lambda(Y, S, W)$ defines the score of the proposed segmentation S for words W and labels Y according to model parameters Λ. Sub-word units σ compose S, where each σ is a phone sequence, including the full pronunciation for vocabulary words; the collection of σs form the lexicon. Each unit σ is present in a segmentation with some context $c = (\phi_l, \phi_r)$ of the form $\phi_l \sigma \phi_r$. Features based on the context and the unit itself parameterize u_Λ.

In addition to scoring a segmentation based on features, we include two priors inspired by the Minimum Description Length (MDL) principle suggested by Poon et al. (2009). The **lexicon prior** favors smaller lexicons by placing an exponential prior with negative weight on the length of the lexicon $\sum_\sigma |\sigma|$, where $|\sigma|$ is the length of the unit σ in number of phones. Minimizing the lexicon prior favors a trivial lexicon of only the phones. The **corpus prior** counters this effect, an exponential prior with negative weight on the number of units in each word's segmentation, where $|s_i|$ is the segmentation length and $|w_i|$ is the length of the word in phones. Learning strikes a balance between the two priors. Using these definitions, the segmentation score $u_\Lambda(Y, S, W)$ is given as:

[1] Since sub-word units can expand full-words, we refer to both words and sub-words simply as units.

[2] The model can also take multiple pronunciations (§3.1).

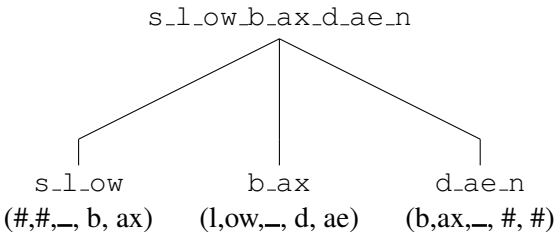


Figure 1: Units and bigram phone context (in parenthesis) for an example segmentation of the word "slobodan".

$$u_\Lambda(Y,S,W) = \exp\Bigg(\sum_{\sigma,y}\lambda_{\sigma,y}f_{\sigma,y}(S,Y) + \sum_{c,y}\lambda_{c,y}f_{c,y}(S,Y) + \alpha\cdot\sum_{\sigma\in S}|\sigma| + \beta\cdot\sum_{i\in W}|s_i|/|w_i|\Bigg) \quad (2)$$

$f_{\sigma,y}(S,Y)$ are the co-occurrence counts of the pair (σ,y) where σ is a unit under segmentation S and y is the label. $f_{c,y}(S,Y)$ are the co-occurrence counts for the context c and label y under S. The model parameters are $\Lambda=\{\lambda_{\sigma,y},\lambda_{c,y}:\forall\sigma,c,y\}$. The negative weights for the lexicon (α) and corpus priors (β) are tuned on development data. The normalizer Z sums over all possible segmentations and labels:

$$Z(W)=\sum_{S'}\sum_{Y'}u_\Lambda(Y',S',W) \quad (3)$$

Consider the example segmentation for the word "slobodan" with pronunciation s,l,ow,b,ax,d,ae,n (Figure 1). The bigram phone context as a four-tuple appears below each unit; the first two entries correspond to the left context, and last two the right context. The example corpus (Figure 2) demonstrates how unit features $f_{\sigma,y}$ and context features $f_{c,y}$ are computed.

3 Model Training

Learning maximizes the log likelihood of the observed labels Y^* given the words W:

$$\ell(Y^*|W)=\log\sum_S\frac{1}{Z(W)}u_\Lambda(Y^*,S,W) \quad (4)$$

We use the Expectation-Maximization algorithm, where the *expectation step* predicts segmentations S given the model's current parameters Λ (§3.1), and the *maximization step* updates these parameters using gradient ascent. The partial derivatives of the objective (4) with respect to each parameter λ_i are:

$$\frac{\partial\ell(Y^*|W)}{\partial\lambda_i}=E_{S|Y^*,W}[f_i]-E_{S,Y|W}[f_i] \quad (5)$$

The gradient takes the usual form, where we encourage the expected segmentation from the current model given the correct labels to equal the expected segmentation and expected labels. The next section discusses computing these expectations.

Labeled corpus: president/$y=0$ milosevic/$y=1$
Segmented corpus: p_r_eh_z_ih_d_ih_n_t/0 m_ih/1 l_aa/1 s_ax/1 v_ih_ch/1
Unit-feature:Value p_r_eh_z_ih_d_ih_n_t/0:1 m_ih/1:1 l_aa/1:1 s_ax/1:1 v_ih_ch/1:1
Context-feature:Value
(#/0,#/0,_,l/1,aa/1):1,
(m/1,ih/1,_,s/1,ax/1):1,
(l/1,aa/1,_,v/1,ih/1):1,
(s/1,ax/1,_,#/0,#/0):1,
(#/0,#/0,_,#/0,#/0):1

Figure 2: A small example corpus with segmentations and corresponding features. The notation m_ih/1:1 represents unit/label:feature-value. Overlapping context features capture rich segmentation regularities associated with each class.

3.1 Inference

Inference is challenging since the lexicon prior renders all word segmentations interdependent. Consider a simple two word corpus: cesar (s,iy,z,er), and cesium (s,iy,z,iy,ax,m). Numerous segmentations are possible; each word has 2^{N-1} possible segmentations, where N is the number of phones in its pronunciation (i.e., $2^3\times2^5=256$). However, if we decide to segment the first word as: {s_iy, z_er}, then the segmentation for "cesium":{s_iy, z_iy_ax_m} will incur a lexicon prior penalty for including the new segment z_iy_ax_m. If instead we segment "cesar" as {s_iy_z, er}, the segmentation {s_iy, z_iy_ax_m} incurs double penalty for the lexicon prior (since we are including two new units in the lexicon: s_iy and z_iy_ax_m). This dependency requires joint segmentation of the entire corpus, which is intractable. Hence, we resort to approximations of the expectations in Eq. (5).

One approach is to use Gibbs Sampling: iterating through each word, sampling a new seg-

mentation conditioned on the segmentation of all other words. The sampling distribution requires enumerating all possible segmentations for each word (2^{N-1}) and computing the conditional probabilities for each segmentation: $P(S|Y^*, W) = P(Y^*, S|W)/P(Y^*|W)$ (the features are extracted from the remaining words in the corpus). Using M sampled segmentations $S_1, S_2, \ldots S_m$ we compute $E_{S|Y^*,W}[f_i]$ as follows:

$$E_{S|Y^*,W}[f_i] \approx \frac{1}{M} \sum_j f_i[S_j]$$

Similarly, to compute $E_{S,Y|W}$ we sample a segmentation and a label for each word. We compute the joint probability of $P(Y, S|W)$ for each segmentation-label pair using Eq. (1). A sampled segmentation can introduce new units, which may have higher probability than existing ones.

Using these approximations in Eq. (5), we update the parameters using gradient ascent:

$$\bar{\lambda}_{new} = \bar{\lambda}_{old} + \gamma \nabla \ell_{\bar{\lambda}}(Y^*|W)$$

where $\gamma > 0$ is the learning rate.

To obtain the best segmentation, we use deterministic annealing. Sampling operates as usual, except that the parameters are divided by a value, which starts large and gradually drops to zero. To make burn in faster for sampling, the sampler is initialized with the most likely segmentation from the previous iteration. To initialize the sampler the first time, we set all the parameters to zero (only the priors have non-zero values) and run deterministic annealing to obtain the first segmentation of the corpus.

3.2 Efficient Sampling

Sampling a segmentation for the corpus requires computing the normalization constant (3), which contains a summation over all possible corpus segmentations. Instead, we approximate this constant by sampling words independently, keeping fixed all other segmentations. Still, even sampling a single word's segmentation requires enumerating probabilities for all possible segmentations.

We sample a segmentation efficiently using dynamic programming. We can represent all possible segmentations for a word as a finite state machine (FSM) (Figure 3), where arcs weights arise from scoring the segmentation's features. This weight is the negative log probability of the resulting model after adding the corresponding features and priors.

However, the lexicon prior poses a problem for this construction since the penalty incurred by a new unit in the segmentation depends on whether that unit is present elsewhere in that segmentation. For example, consider the segmentation for the word ANJANI: AA_N, JH, AA_N, IY. If none of these units are in the lexicon, this segmentation yields the lowest prior penalty since it repeats the unit AA_N. [3] This global dependency means paths must encode the full unit history, making computing forward-backward probabilities inefficient.

Our solution is to use the *Metropolis-Hastings* algorithm, which samples from the true distribution $P(Y, S|W)$ by first sampling a new label and segmentation (y', s') from a simpler proposal distribution $Q(Y, S|W)$. The new assignment (y', s') is accepted with probability:

$$\alpha(Y', S'|Y, S, W) = \min\left(1, \frac{P(Y', S'|W)Q(Y, S|Y', S', W)}{P(Y, S|W)Q(Y', S'|Y, S, W)}\right)$$

We choose the proposal distribution $Q(Y, S|W)$ as Eq. (1) omitting the lexicon prior, removing the challenge for efficient computation. The probability of accepting a sample becomes:

$$\alpha(Y', S'|Y, S, W) = \min\left(1, \frac{\sum_{\sigma \in S'} |\sigma|}{\sum_{\sigma \in S} |\sigma|}\right) \quad (6)$$

We sample a path from the FSM by running the forward-backward algorithm, where the backward computations are carried out explicitly, and the forward pass is done through sampling, i.e. we traverse the machine only computing forward probabilities for arcs leaving the sampled state.[4] Once we sample a segmentation (and label) we accept it according to Eq. (6) or keep the previous segmentation if rejected.

Alg. 1 shows our full sub-word learning procedure, where `sampleSL` (Alg. 2) samples a segmentation and label sequence for the entire corpus from $P(Y, S|W)$, and `sampleS` samples a segmentation from $P(S|Y^*, W)$.

[3] Splitting at phone boundaries yields the same lexicon prior but a higher corpus prior.

[4] We use OpenFst's RandGen operation with a costumed arc-selector (http://www.openfst.org/).

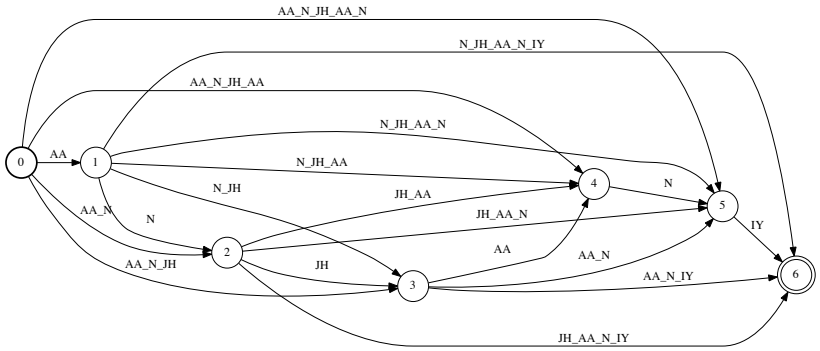


Figure 3: FSM representing all segmentations for the word ANJANI with pronunciation: AA,N,JH,AA,N,IY

Algorithm 1 Training

Input: Lexicon L from training text W, Dictionary D, Mapping M, L2S pronunciations, Annealing temp T.
Initialization:
Assign label $y^*_m = M[w_m]$. $\bar{\lambda}_0 = \bar{0}$
S_0 = random segmentation for each word in L.

for $i = 1$ **to** K **do**
/* **E-Step** */
S_i = bestSegmentation(T, λ_{i-1}, S_{i-1}).
for $k = 1$ **to** NumSamples **do**
(S'_k, Y'_k) = sampleSL($P(Y, S_i|W)$,$Q(Y, S_i|W)$)
$\tilde{S}_k$ = sampleS($P(S_i|Y^*, W)$,$Q(S_i|Y^*, W)$)
end for
/* **M-Step** */
$E_{S,Y|W}[f_i] = \frac{1}{NumSamples} \sum_k f_{\sigma,l}[S'_k, Y'_k]$
$E_{S|Y^*,W}[f_{\sigma,l}] = \frac{1}{NumSamples} \sum_k f_{\sigma,l}[\tilde{S}_k, Y^*]$
$\bar{\lambda}_i = \bar{\lambda}_{i-1} + \gamma \nabla L_{\bar{\lambda}}(Y^*|W)$
end for
S = bestSegmentation(T, λ_K, S_0)
Output: Lexicon L_o from S

Algorithm 2 sampleSL($P(S, Y|W), Q(S, Y|W)$)

for $m = 1$ **to** M (NumWords) **do**
(s'_m, y'_m) = Sample segmentation/label pair for word w_m according to $Q(S, Y|W)$
$Y' = \{y_1 \dots y_{m-1} y'_m y_{m+1} \dots y_M\}$
$S' = \{s_1 \dots s_{m-1} s'_m s_{m+1} \dots s_M\}$
α=min $\left(1, \frac{\sum_{\sigma \in S'} |\sigma|}{\sum_{\sigma \in S} |\sigma|}\right)$
with prob $\alpha : y_{m,k} = y'_m, s_{m,k} = s'_m$
with prob $(1 - \alpha) : y_{m,k} = y_m, s_{m,k} = s_m$
end for
return $(S'_k, Y'_k) = [(s_{1,k}, y_{1,k}) \dots (s_{M,k}, y_{M,k})]$

4 OOV Detection Using Hybrid Models

To evaluate our model for learning sub-word units, we consider the task of out-of-vocabulary (OOV) word detection. OOV detection for ASR output can be categorized into two broad groups: 1) *hybrid (filler) models*: which explicitly model OOVs using either filler, sub-words, or generic word models (Bazzi, 2002; Schaaf, 2001; Bisani and Ney, 2005; Klakow et al., 1999; Wang, 2009); and 2) *confidence-based approaches*: which label unreliable regions as OOVs based on different confidence scores, such as acoustic scores, language models, and lattice scores (Lin et al., 2007; Burget et al., 2008; Sun et al., 2001; Wessel et al., 2001).

In the next section we detail the OOV detection approach we employ, which combines hybrid and confidence-based models, achieving state-of-the art performance for this task.

4.1 OOV Detection Approach

We use the state-of-the-art OOV detection model of Parada et al. (2010), a second order CRF with features based on the output of a hybrid recognizer. This detector processes hybrid recognizer output, so we can evaluate different sub-word unit lexicons for the hybrid recognizer and measure the change in OOV detection accuracy.

Our model (§2.1) can be applied to this task by using a dictionary D to label words as IV ($y_i = 0$ if $w_i \in D$) and OOV ($y_i = 1$ if $w_i \notin D$). This results in a labeled corpus, where the labeling sequence Y indicates the presence of out-of-vocabulary words (OOVs). For comparison we evaluate a baseline method (Rastrow et al., 2009b) for selecting units.

Given a sub-word lexicon, the word and sub-words are combined to form a hybrid language model (LM) to be used by the LVCSR system. This hybrid LM captures dependencies between word and sub-words. In the LM training data, all OOVs are represented by the smallest number of sub-words which corresponds to their pronunciation. Pronunciations for all OOVs are obtained using grapheme

to phone models (Chen, 2003).

Since sub-words represent OOVs while building the hybrid LM, the existence of sub-words in ASR output indicate an OOV region. A simple solution to the OOV detection problem would then be reduced to a search for the sub-words in the output of the ASR system. The search can be on the one-best transcripts, lattices or confusion networks. While lattices contain more information, they are harder to process; confusion networks offer a trade-off between richness (posterior probabilities are already computed) and compactness (Mangu et al., 1999).

Two effective indications of OOVs are the existence of sub-words (Eq. 7) and high entropy in a network region (Eq. 8), both of which are used as features in the model of Parada et al. (2010).

$$\text{Sub-word Posterior} = \sum_{\sigma \in t_j} p(\sigma|t_j) \quad (7)$$

$$\text{Word-Entropy} = -\sum_{w \in t_j} p(w|t_j) \log p(w|t_j) \quad (8)$$

t_j is the current bin in the confusion network and σ is a sub-word in the hybrid dictionary. Improving the sub-word unit lexicon, improves the quality of the confusion networks for OOV detection.

5 Experimental Setup

We used the data set constructed by Can et al. (2009) (OOVCORP) for the evaluation of Spoken Term Detection of OOVs since it focuses on the OOV problem. The corpus contains 100 hours of transcribed Broadcast News English speech. There are 1290 unique OOVs in the corpus, which were selected with a minimum of 5 acoustic instances per word and short OOVs inappropriate for STD (less than 4 phones) were explicitly excluded. Example OOVs include: NATALIE, PUTIN, QAEDA, HOLLOWAY, COROLLARIES, HYPERLINKED, etc. This resulted in roughly 24K (2%) OOV tokens.

For LVCSR, we used the IBM Speech Recognition Toolkit (Soltau et al., 2005)[5] to obtain a transcript of the audio. Acoustic models were trained on 300 hours of HUB4 data (Fiscus et al., 1998) and utterances containing OOV words as marked in OOVCORP were excluded. The language model was trained on 400M words from various text sources with a 83K word vocabulary. The LVCSR system's WER on the standard RT04 BN test set was 19.4%. Excluded utterances amount to 100hrs. These were divided into 5 hours of training for the OOV detector and 95 hours of test. Note that the OOV detector training set is different from the LVCSR training set.

We also use a hybrid LVCSR system, combining word and sub-word units obtained from either our approach or a state-of-the-art baseline approach (Rastrow et al., 2009a) (§5.2). Our hybrid system's lexicon has 83K words and 5K or 10K sub-words. Note that the word vocabulary is common to both systems and only the sub-words are selected using either approach. The word vocabulary used is close to most modern LVCSR system vocabularies for English Broadcast News; the resulting OOVs are more challenging but more realistic (i.e. mostly named entities and technical terms). The 1290 words are OOVs to both the word and hybrid systems.

In addition we report OOV detection results on a MIT lectures data set (Glass et al., 2010) consisting of 3 Hrs from two speakers with a 1.5% OOV rate. These were divided into 1 Hr for training the OOV detector and 2 Hrs for testing. Note that the LVCSR system is trained on Broadcast News data. This out-of-domain test-set help us evaluate the cross-domain performance of the proposed and baseline hybrid systems. OOVs in this data set correspond mainly to technical terms in computer science and math. e.g. ALGORITHM, DEBUG, COMPILER, LISP.

5.1 Learning parameters

For learning the sub-words we randomly selected from training 5,000 words which belong to the 83K vocabulary and 5,000 OOVs[6]. For development we selected an additional 1,000 IV and 1,000 OOVs. This was used to tune our model hyper parameters (set to $\alpha = -1$, $\beta = -20$). There is no overlap of OOVs in training, development and test sets. All feature weights were initialized to zero and had a Gaussian prior with variance $\sigma = 100$. Each of the words in training and development was converted to their most-likely pronunciation using the dictionary

[5]The IBM system used speaker adaptive training based on maximum likelihood with no discriminative training.

[6]This was used to obtain the 5K hybrid system. To learn sub-words for the 10K hybrid system we used 10K in-vocabulary words and 10K OOVs. All words were randomly selected from the LM training text.

for IV words or the L2S model for OOVs.[7]

The learning rate was $\gamma_k = \frac{\gamma}{(k+1+A)^\tau}$, where k is the iteration, A is the stability constant (set to $0.1K$), $\gamma = 0.4$, and $\tau = 0.6$. We used $K = 40$ iterations for learning and 200 samples to compute the expectations in Eq. 5. The sampler was initialized by sampling for 500 iterations with deterministic annealing for a temperature varying from 10 to 0 at 0.1 intervals. Final segmentations were obtained using $10,000$ samples and the same temperature schedule. We limit segmentations to those including units of at most 5 phones to speed sampling with no significant degradation in performance. We observed improved performance by dis-allowing whole word units.

5.2 Baseline Unit Selection

We used Rastrow et al. (2009a) as our baseline unit selection method, a data driven approach where the language model training text is converted into phones using the dictionary (or a letter-to-sound model for OOVs), and a N-gram phone LM is estimated on this data and pruned using a relative entropy based method. The hybrid lexicon includes resulting sub-words – ranging from unigrams to 5-gram phones, and the 83K word lexicon.

5.3 Evaluation

We obtain confusion networks from both the word and hybrid LVCSR systems. We align the LVCSR transcripts with the reference transcripts and tag each confusion region as either IV or OOV. The OOV detector classifies each region in the confusion network as IV/OOV. We report OOV detection accuracy using standard detection error tradeoff (DET) curves (Martin et al., 1997). DET curves measure tradeoffs between false alarms (x-axis) and misses (y-axis), and are useful for determining the optimal operating point for an application; lower curves are better. Following Parada et al. (2010) we separately evaluate unobserved OOVs.[8]

[7] In this work we ignore pronunciation variability and simply consider the most likely pronunciation for each word. It is straightforward to extend to multiple pronunciations by first sampling a pronunciation for each word and then sampling a segmentation for that pronunciation.

[8] Once an OOV word has been observed in the OOV detector training data, even if it was not in the LVCSR training data, it is no longer truly OOV.

6 Results

We compare the performance of a hybrid system with baseline units[9] (§5.2) and one with units learned by our model on OOV detection and phone error rate. We present results using a hybrid system with 5k and 10k sub-words.

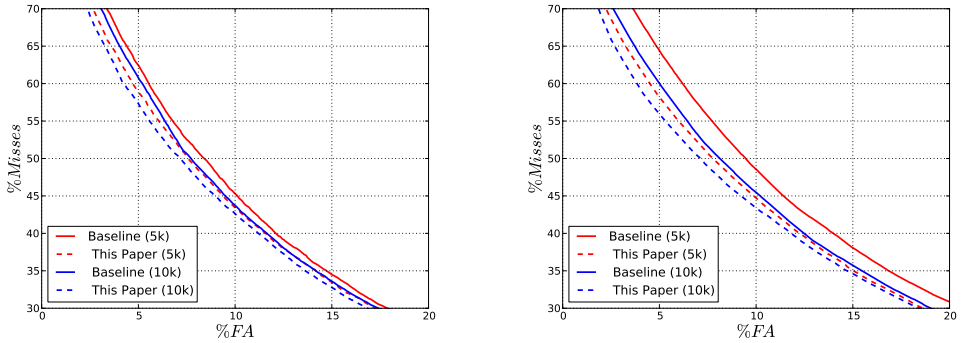
We evaluate the CRF OOV detector with two different feature sets. The first uses only Word Entropy and Sub-word Posterior (Eqs. 7 and 8) (Figure 4)[10]. The second (`context`) uses the extended context features of Parada et al. (2010) (Figure 5). Specifically, we include all trigrams obtained from the best hypothesis of the recognizer (a window of 5 words around current confusion bin). Predictions at different FA rates are obtained by varying a probability threshold.

At a 5% FA rate, our system (`This Paper 5k`) reduces the miss OOV rate by 6.3% absolute over the baseline (`Baseline 5k`) when evaluating all OOVs. For unobserved OOVs, it achieves 3.6% absolute improvement. A larger lexicon (`Baseline 10k` and `This Paper 10k`) shows similar relative improvements. Note that the features used so far do not necessarily provide an advantage for unobserved versus observed OOVs, since they ignore the decoded word/sub-word sequence. In fact, the performance on un-observed OOVs is better.

OOV detection improvements can be attributed to increased coverage of OOV regions by the learned sub-words compared to the baseline. Table 1 shows the percent of Hits: sub-word units predicted in OOV regions, and False Alarms: sub-word units predicted for in-vocabulary words. We can see that the proposed system increases the Hits by over 8% absolute, while increasing the False Alarms by 0.3%. Interestingly, the average sub-word length for the proposed units exceeded that of the baseline units by 0.3 phones (`Baseline 5K` average length was 2.92, while that of `This Paper 5K` was 3.2).

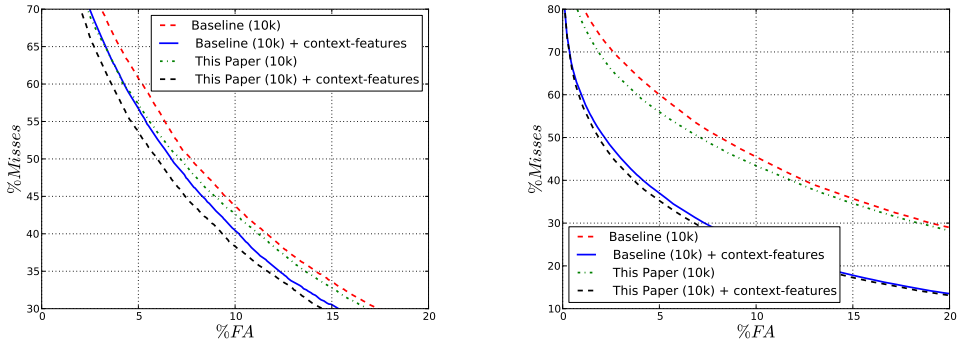
[9] Our baseline results differ from Parada et al. (2010). When implementing the lexicon baseline, we discovered that their hybrid units were mistakenly derived from text containing test OOVs. Once excluded, the relative improvements of previous work remain, but the absolute error rates are higher.

[10] All real-valued features were normalized and quantized using the uniform-occupancy partitioning described in White et al. (2007). We used 50 partitions with a minimum of 100 training values per partition.



(a) (b)

Figure 4: DET curves for OOV detection using baseline hybrid systems for different lexicon size and proposed discriminative hybrid system on **OOVCORP** data set. Evaluation on **un-observed** OOVs (a) and **all** OOVs (b).



(a) (b)

Figure 5: Effect of adding context features to baseline and discriminative hybrid systems on **OOVCORP** data set. Evaluation on **un-observed** OOVs (a) and **all** OOVs (b).

Consistent with previously published results, including context achieves large improvement in performance. The proposed hybrid system (This Paper 10k + context-features) still improves over the baseline (Baseline 10k + context-features), however the relative gain is reduced. In this case, we obtain larger gains for un-observed OOVs which benefit less from the context clues learned in training.

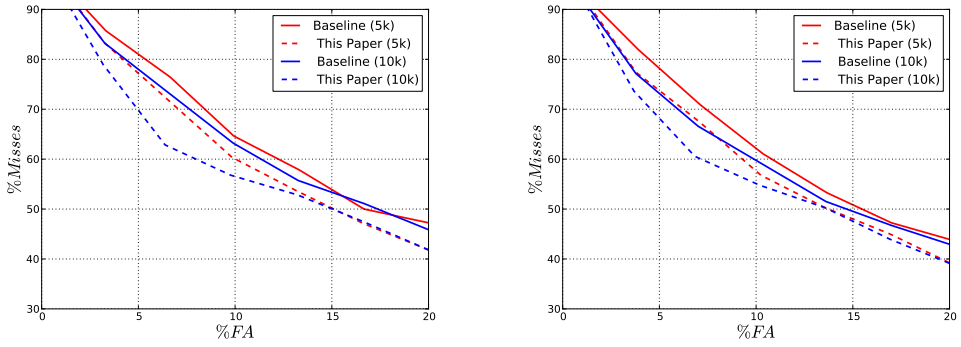
Lastly, we report OOV detection performance on MIT Lectures. Both the sub-word lexicon and the LVCSR models were trained on Broadcast News data, helping us evaluate the robustness of learned sub-words across domains. Note that the OOVs in these domains are quite different: MIT Lectures' OOVs correspond to technical computer science and math terms, while in Broadcast News they are mainly named-entities.

Hybrid System	**Hits**	**FAs**
Baseline (5k)	18.25	1.49
This Paper (5k)	26.78	1.78
Baseline (10k)	24.26	1.82
This Paper (10k)	28.96	1.92

Table 1: Coverage of OOV regions by baseline and proposed sub-words in OOVCORP.

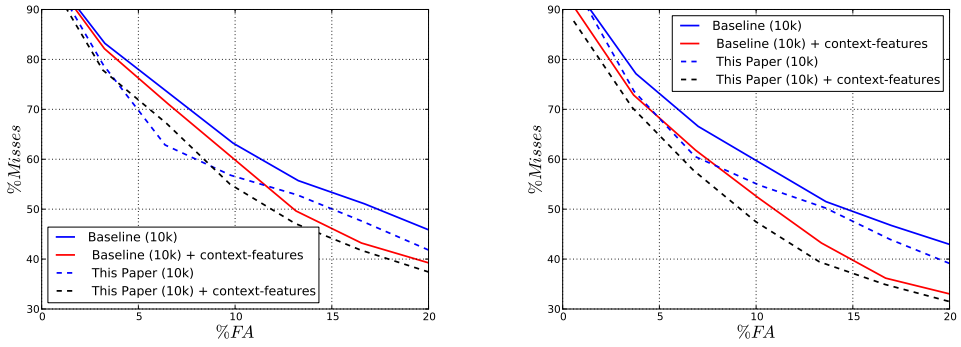
Figure 6 and 7 show the OOV detection results in the MIT Lectures data set. For un-observed OOVs, the proposed system (`This Paper 10k`) reduces the miss OOV rate by 7.6% with respect to the baseline (`Baseline 10k`) at a 5% FA rate. Similar to Broadcast News results, we found that the learned sub-words provide larger coverage of OOV regions in MIT Lectures domain. These results suggest that the proposed sub-words are not simply modeling the training OOVs (named-entities) better than the baseline sub-words, but also describe better novel unexpected words. Furthermore, including context features does not seem as helpful. We conjecture that this is due to the higher WER[11] and the less structured nature of the domain: i.e. ungrammatical sentences, disfluencies, incomplete sentences, making it more difficult to predict OOVs based on context.

[11] $WER = 32.7\%$ since the LVCSR system was trained on Broadcast News data as described in Section 5.



(a) (b)

Figure 6: DET curves for OOV detection using baseline hybrid systems for different lexicon size and proposed discriminative hybrid system on **MIT Lectures** data set. Evaluation on **un-observed** OOVs (a) and **all** OOVs (b).



(a) (b)

Figure 7: Effect of adding context features to baseline and discriminative hybrid systems on **MIT Lectures** data set. Evaluation on **un-observed** OOVs (a) and **all** OOVs (b).

6.1 Improved Phonetic Transcription

We consider the hybrid lexicon's impact on Phone Error Rate (PER) with respect to the reference transcription. The reference phone sequence is obtained by doing *forced alignment* of the audio stream to the reference transcripts using acoustic models. This provides an alignment of the pronunciation variant of each word in the reference and the recognizer's one-best output. The aligned words are converted to the phonetic representation using the dictionary.

Table 2 presents PERs for the word and different hybrid systems. As previously reported (Rastrow et al., 2009b), the hybrid systems achieve better PER, specially in OOV regions since they predict sub-word units for OOVs. Our method achieves modest improvements in PER compared to the hybrid baseline. No statistically significant improvements in PER were observed on MIT Lectures.

System	OOV	IV	All
Word	1.62	6.42	8.04
Hybrid: Baseline (5k)	1.56	6.44	8.01
Hybrid: Baseline (10k)	1.51	6.41	7.92
Hybrid: This Paper (5k)	1.52	6.42	7.94
Hybrid: This Paper (10k)	1.45	6.39	7.85

Table 2: Phone Error Rate for OOVCORP.

7 Conclusions

Our probabilistic model learns sub-word units for hybrid speech recognizers by segmenting a text corpus while exploiting side information. Applying our method to the task of OOV detection, we obtain an absolute error reduction of 6.3% and 7.6% at a 5% false alarm rate on an English Broadcast News and MIT Lectures task respectively, when compared to a baseline system. Furthermore, we have confirmed previous work that hybrid systems achieve better phone accuracy, and our model makes modest improvements over a baseline with a similarly sized sub-word lexicon. We plan to further explore our new lexicon's performance for other languages and tasks, such as OOV spoken term detection.

Acknowledgments

We gratefully acknowledge Bhuvaha Ramabhadran for many insightful discussions and the anonymous reviewers for their helpful comments. This work was funded by a Google PhD Fellowship.

References

Issam Bazzi and James Glass. 2001. Learning units for domain-independent out-of-vocabulary word modeling. In *EuroSpeech*.

Issam Bazzi. 2002. *Modelling out-of-vocabulary words for robust speech recognition*. Ph.D. thesis, Massachusetts Institute of Technology.

M. Bisani and H. Ney. 2005. Open vocabulary speech recognition with flat hybrid models. In *INTERSPEECH*.

L. Burget, P. Schwarz, P. Matejka, M. Hannemann, A. Rastrow, C. White, S. Khudanpur, H. Hermansky, and J. Cernocky. 2008. Combination of strongly and weakly constrained recognizers for reliable detection of OOVS. In *ICASSP*.

D. Can, E. Cooper, A. Sethy, M. Saraclar, and C. White. 2009. Effect of pronounciations on OOV queries in spoken term detection. *Proceedings of ICASSP*.

Stanley F. Chen. 2003. Conditional and joint models for grapheme-to-phoneme conversion. In *Eurospeech*, pages 2033–2036.

G. Choueiter. 2009. *Linguistically-motivated subword modeling with applications to speech recognition*. Ph.D. thesis, Massachusetts Institute of Technology.

Jonathan Fiscus, John Garofolo, Mark Przybocki, William Fisher, and David Pallett, 1998. *1997 English Broadcast News Speech (HUB4)*. Linguistic Data Consortium, Philadelphia.

James Glass, Timothy Hazen, Lee Hetherington, and Chao Wang. 2010. Analysis and processing of lecture audio data: Preliminary investigations. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Dietrich Klakow, Georg Rose, and Xavier Aubert. 1999. OOV-detection in large vocabulary system using automatically defined word-fragments as fillers. In *Eurospeech*.

Hui Lin, J. Bilmes, D. Vergyri, and K. Kirchhoff. 2007. OOV detection by joint word/phone lattice alignment. In *ASRU*, pages 478–483, Dec.

Jonathan Mamou, Bhuvana Ramabhadran, and Olivier Siohan. 2007. Vocabulary independent spoken term detection. In *Proceedings of SIGIR*.

L. Mangu, E. Brill, and A. Stolcke. 1999. Finding consensus among words. In *Eurospeech*.

A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocky. 1997. The det curve in assessment of detection task performance. In *Eurospeech*.

Carolina Parada, Abhinav Sethy, and Bhuvana Ramabhadran. 2009. Query-by-example spoken term detection for oov terms. In *ASRU*.

Carolina Parada, Mark Dredze, Denis Filimonov, and Fred Jelinek. 2010. Contextual information improves oov detection in speech. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

H. Poon, C. Cherry, and K. Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *ACL*.

Ariya Rastrow, Abhinav Sethy, and Bhuvana Ramabhadran. 2009a. A new method for OOV detection using hybrid word/fragment system. *Proceedings of ICASSP*.

Ariya Rastrow, Abhinav Sethy, Bhuvana Ramabhadran, and Fred Jelinek. 2009b. Towards using hybrid, word, and fragment units for vocabulary independent LVCSR systems. *INTERSPEECH*.

T. Schaaf. 2001. Detection of OOV words using generalized word models and a semantic class language model. In *Eurospeech*.

H. Soltau, B. Kingsbury, L. Mangu, D. Povey, G. Saon, and G. Zweig. 2005. The ibm 2004 conversational telephony system for rich transcription. In *ICASSP*.

H. Sun, G. Zhang, f. Zheng, and M. Xu. 2001. Using word confidence measure for OOV words detection in a spontaneous spoken dialog system. In *Eurospeech*.

Stanley Wang. 2009. Using graphone models in automatic speech recognition. Master's thesis, Massachusetts Institute of Technology.

F. Wessel, R. Schluter, K. Macherey, and H. Ney. 2001. Confidence measures for large vocabulary continuous speech recognition. *IEEE Transactions on Speech and Audio Processing*, 9(3).

Christopher White, Jasha Droppo, Alex Acero, and Julian Odell. 2007. Maximum entropy confidence estimation for speech recognition. In *ICASSP*.

Computing and Evaluating Syntactic Complexity Features for Automated Scoring of Spontaneous Non-Native Speech

Miao Chen
School of Information Studies
Syracuse University
Syracuse, NY, USA
mchen14@syr.edu

Klaus Zechner
NLP & Speech Group
Educational Testing Service
Princeton, NJ, USA
kzechner@ets.org

Abstract

This paper focuses on identifying, extracting and evaluating features related to syntactic complexity of spontaneous spoken responses as part of an effort to expand the current feature set of an automated speech scoring system in order to cover additional aspects considered important in the construct of communicative competence.

Our goal is to find effective features, selected from a large set of features proposed previously and some new features designed in analogous ways from a syntactic complexity perspective that correlate well with human ratings of the same spoken responses, and to build automatic scoring models based on the most promising features by using machine learning methods.

On human transcriptions with manually annotated clause and sentence boundaries, our best scoring model achieves an overall Pearson correlation with human rater scores of r=0.49 on an unseen test set, whereas correlations of models using sentence or clause boundaries from automated classifiers are around r=0.2.

1 Introduction

Past efforts directed at automated scoring of speech have used mainly features related to fluen cy (e.g., speaking rate, length and distribution of pauses), pronunciation (e.g., using log-likelihood scores from the acoustic model of an Automatic Speech Recognition (ASR) system), or prosody (e.g., information related to pitch contours or syllable stress) (e.g., Bernstein, 1999; Bernstein et al., 2000; Bernstein et al., 2010; Cucchiarini et al., 1997; Cucchiarini et al., 2000; Franco et al., 2000a; Franco et al., 2000b; Zechner et al., 2007, Zechner et al., 2009).

While this approach is a good match to most of the important properties related to low entropy speech (i.e., speech which is highly predictable), such as reading a passage aloud, it lacks many important aspects of spontaneous speech which are relevant to be evaluated both by a human rater and an automated scoring system. Examples of such aspects of speech, which are considered part of the construct[1] of "communicative competence (Bachman, 1990), include grammatical accuracy, syntactic complexity, vocabulary diversity, and aspects of spoken discourse structure, e.g., coherence and cohesion. These different aspects of speaking proficiency are often highly correlated in a non-native speaker (Xi and Mollaun, 2006; Bernstein et al., 2010), and so scoring models built solely on features of fluency and pronunciation may achieve reasonably high correlations with holistic human rater scores. However, it is important to point out that such systems would still be unable to assess many important aspects of the speaking construct and therefore cannot be seen as ideal from a validity point of view.[2]

The purpose of this paper is to address one of these important aspects of spoken language in more detail, namely syntactic complexity. This paper can be seen as a first step toward including

[1] A construct is a set of knowledge, skills, and abilities measured by a test.

[2] "Construct validity" refers to the extent that a test measures what it is designed to measure, in this case, communicative competence via speaking.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 722–731,
Portland, Oregon, June 19-24, 2011.

features related to this part of the speaking construct into an already existing automated speech scoring system for spontaneous speech which so far mostly uses features related to fluency and pronunciation (Zechner et al., 2009).

We use data from the speaking section of the TOEFL® Practice Online (TPO) test, which is a low stakes practice test for non-native speakers where they are asked to provide six spontaneous speech samples of about one minute in length each in response to a variety of prompts. Some prompts may be simple questions, and others may involve reading or listening to passages first and then answering related questions. All responses were scored holistically by human raters according to pre-defined scoring rubrics (i.e., specific scoring guidelines) on a scale of 1 to 4, 4 being the highest proficiency level.

In our automated scoring system, the first component is an ASR system that decodes the digitized speech sample, generating a time-annotated hypothesis for every response. Next, fluency and pronunciation features are computed based on the ASR output hypotheses, and finally a multiple regression scoring model, trained on human rater scores, computes the score for a given spoken response (see Zechner et al. (2009) for more details). We conducted the study in three steps: (1) finding important measures of syntactic complexity from second language acquisition (SLA) and English language learning (ELL) literature, and extending this feature set based on our observations of the TPO data in analogous ways; (2) computing features based on transcribed speech responses and selecting features with highest correlations to human rater scores, also considering their comparative values for native speakers taking the same test; and (3) building scoring models for the selected sub-set of the features to generate a proficiency score for each speaker, using all six responses of that speaker.

In the remainder of the paper, we will address related work in syntactic complexity (Section 2), introduce the speech data sets of our study (Section 3), describe the methods we used for feature extraction (Section 4), provide the experiment design and results (Section 5), analyze and discuss the results in Section 6, before concluding the paper (Section 7).

2 Related Work

2.1 Literature on Syntactic Complexity

Syntactic complexity is defined as "the range of forms that surface in language production and the degree of sophistication of such forms" (Ortega, 2003). It is an important factor in the second language assessment construct as described in Bachman's (1990) conceptual model of language ability, and therefore is often used as an index of language proficiency and development status of L2 learners. Various studies have proposed and investigated measures of syntactic complexity as well as examined its predictiveness for language proficiency, in both L2 writing and speaking settings, which will be reviewed respectively.

Writing

Wolfe-Quintero et al. (1998) reviewed a number of grammatical complexity measures in L2 writing from thirty-nine studies, and their usage for predicting language proficiency was discussed. Some examples of syntactic complexity measures are: mean number of clauses per T-unit[3], mean length of clauses, mean number of verbs per sentence, etc. The various measures can be grouped into two categories: (1) clauses, sentences, and T-units in terms of each other; and (2) specific grammatical structures (e.g., passives, nominals) in relation to clauses, sentences, or T-units (Wolfe-Quintero et al., 1998). Three primary methods of calculating syntactic complexity measures are frequency, ratio, and index, where frequency is the count of occurrences of a specific grammatical structure, ratio is the number of one type of unit divided by the total number of another unit, and index is computing numeric scores by specific formulae (Wolfe-Quintero et al., 1998). For example, the measure "mean number of clauses per T-unit" is obtained by using the ratio calculation method and the clause and T-unit grammatical structures. Some structures such as clauses and T-units only need shallow linguistic processing to acquire, while some require parsing. There are numerous combinations for measures and we need empirical evi-

[3] T-units are defined as "shortest grammatically allowable sentences into which (writing can be split) or minimally terminable units" (Hunt, 1965:20).

dence to select measures with the highest performance.

There have been a series of empirical studies examining the relationship of syntactic complexity measures to L2 proficiency using real-world data (Cooper, 1976; Larsen-Freeman, 1978; Perkins, 1980; Ho-Peng, 1983; Henry, 1996; Ortega, 2003; Lu, 2010). The studies investigate measures that highly correlate with proficiency levels or distinguish between different proficiency levels. Many T-unit related measures were identified as statistically significant indicators to L2 proficiency, such as mean length of T-unit (Henry, 1996; Lu, 2010), mean number of clauses per T-unit (Cooper, 1976; Lu, 2010), mean number of complex nominals per T-unit (Lu, 2010), or the mean number of error-free T-units per sentence (Ho-Peng, 1983). Other significant measures are mean length of clause (Lu, 2010), or frequency of passives in composition (Kameen, 1979).

Speaking

Syntactic complexity analysis in speech mainly inherits measures from the writing domain, and the abovementioned measures can be employed in the same way on speech transcripts for complexity computation. A series of studies have examined relations between the syntactic complexity of speech and the speakers' holistic speaking proficiency levels (Halleck, 1995; Bernstein et al., 2010; Iwashita, 2006). Three objective measures of syntactic complexity, including mean T-unit length, mean error-free T-unit length, and percent of error-free T-units were found to correlate with holistic evaluations of speakers in Halleck (1995). Iwashita's (2006) study on Japanese L2 speakers found that length-based complexity features (i.e., number of T-units and number of clauses per T-unit) are good predictors for oral proficiency. In studies directly employing syntactic complexity measures in other contexts, ratio-based measures are frequently used. Examples are mean length of utterance (Condouris et al., 2003), word count or tree depth (Roll et al., 2007), or mean length of T-units and mean number of clauses per T-unit (Bernstein et al., 2010). Frequency-based measures were used less, such as number of full phrases in Roll et al. (2007).

The speaking output is usually less clean than writing data (e.g., considering disfluencies such as false starts, repetitions, filled pauses etc.). Therefore we may need to remove these disfluencies first before computing syntactic complexity features. Also, importantly, ASR output does not contain interpunctuation but both for sentential-based features as well as for parser-based features, the boundaries of clauses and sentences need to be known. For this purpose, we will use automated classifiers that are trained to predict clause and sentence boundaries, as described in Chen et al. (2010). With previous studies providing us a rich pool of complexity features, additionally we also develop features analogous to the ones from the literature, mostly by using different calculation methods. For instance, the frequency of Prepositional Phrases (PPs) is a feature from the literature, and we add some variants such as number of PPs per clause as a new feature to our extended feature set.

2.2 Devising the Initial Feature Set

Through this literature review, we identified some important features that were frequently used in previous studies in both L2 speaking and writing, such as length of sentences and number of clauses per sentence. In addition, we also collected candidate features that were less frequently mentioned in the literature, in order to start with a larger field of potential candidate features. We further extended the feature set by inspecting our data, described in the following section, and created suitable additional features by means of analogy. This process resulted in a set of 91 features, 11 of which are related to clausal and sentential unit measurements (frequency-based) and 80 to measurements within such units (ratio-based). From the perspective of extracting measures, in our study, some measures can be computed using only clause and sentence boundary information, and some can be derived only if the spoken responses are syntactically parsed. In our feature set, there are two types of features: clause and sentence boundary based (26 in total) and parsing based (65). The features will be described in detail in Section 4.

3 Data

Our data set contains (1) 1,060 non-native speech responses of 189 speakers from the TPO test (NN set), and (2) 100 responses from 48 native speakers that took the same test (Nat set). All responses were verbatim transcribed manually and scored

holistically by human raters. (We only made use of the scores for the non-native data set in this study, since we purposefully selected speakers with perfect or near perfect scores for the Nat set from a larger native speech data set.) As mentioned above, there are four proficiency levels for human scoring, levels 1 to 4, with higher levels indicating better speaking proficiency.

The NN set was randomly partitioned into a training (NN-train) and a test set with 760 and 300 responses, respectively, and no speaker overlap.

Data Set	**Responses**	**Speakers**	**Responses per Speaker (average)**
NN-train	760	137	5.55
	Description: used to train sentence and clause boundary detectors, evaluate features and train scoring models		
1: NN-test-1-Hum	300	52	5.77
	Description: human transcriptions and annotations of sentence and clause boundaries		
2: NN-test-2-CB	300	52	5.77
	Description: human transcriptions, automatically predicted clause boundaries		
3: NN-test-3-SB	300	52	5.77
	Description: human transcriptions, automatically predicted sentence boundaries		
4: NN-test-4-ASR-CB	300	52	5.77
	Description: ASR hypotheses, automatically predicted clause boundaries		
5: NN-test-5-ASR-SB	300	52	5.77
	Description: ASR hypotheses, automatically predicted sentence boundaries		

Table 1. Overview of non-native data sets.

A second version of the test set contains ASR hypotheses instead of human transcriptions. The word error rate (WER[4]) on this data set is 50.5%. We used a total of five variants of the test sets, as described in Table 1. Sets 1-3 are based on human transcriptions, whereas sets 4 and 5 are based on ASR output. Further, set 1 contains human annotated clause and sentence boundaries, whereas the other 4 sets have clause or sentence boundaries predicted by a classifier.

All human transcribed files from the NN data set were annotated for clause boundaries, clause types, and disfluencies by human annotators (see Chen et al. (2010)).

For the Nat data set, all of the 100 transcribed responses were annotated in the same manner by a human annotator. They are not used for any training purposes but serve as a comparative reference for syntactic complexity features derived from the non-native corpus.

The NN-train set was used both for training clause and sentence boundary classifiers, as well as for feature selection and training of the scoring models. The two boundary detectors were machine learning based Hidden Markov Models, trained by using a language model derived from the 760 training files which had sentence and clause boundary labels (NN-train; see also Chen et al. (2010)).

Since a speaker's response to a single test item can be quite short (fewer than 100 words in many cases), it may contain only very few syntactic complexity features we are looking for. (Note that much of the previous work focused on written language with much longer texts to be considered.) However, if we aggregate responses of a single speaker, we have a better chance of finding a larger number of syntactic complexity features in the aggregated file. Therefore we joined files from the same speaker to one file for the training set and the five test sets, resulting in 52 aggregated files in each test set. Accordingly, we averaged the response scores of a single speaker to obtain the total speaker score to be used later in scoring model training and evaluation (Section 5).[5]

While disfluencies were used for the training of the boundary detectors, they were removed afterwards from the annotated data sets to obtain a tran-

[4] Word error rate (WER) is the ratio of errors from a string between the ASR hypothesis and the reference transcript, where the sum of substitutions, insertions, and deletions is divided by the length of the reference. To obtain WER in percent, this ratio is multiplied by 100.0.

[5] Although in most operational settings, features are derived from single responses, this may not be true in all cases. Furthermore, scores of multiple responses are often combined for score reporting, which would make such an approach easier to implement and argue for operationally.

scription which is "cleaner" and lends itself better to most of the feature extraction methods we use.

4 Feature Extraction

4.1 Feature Set

As mentioned in Section 2, we gathered 91 candidate syntactic complexity features based on our literature review as initial feature set, which is grouped into two categories: (1) Clause and sentence Boundary based features (CB features); and (2) Parse Tree based features (PT features). Clause based features are based on both clause boundaries and clause types and can be generated from human clause annotations, e.g., "frequency of adjective clauses[6] per one thousand words", "mean number of dependent clauses per clause", etc. Parse tree based features refer to features that are generated from parse trees and cannot be extracted from human annotated clauses directly.

We first selected features showing high correlation to human assigned scores. In this process the CB features were computed from human labeled clause boundaries in transcripts for best accuracy, and PT features were calculated from using parsing and other tools because we did not have human parse tree annotations for our data.

We used the Stanford Parser (Klein and Manning, 2003) in conjunction with the Stanford Tregex package (Levy and Andrew, 2006) which supports using rules to extract specific configurations from parse trees, in a package put together by Lu (Lu, 2011). When given a sentence, the Stanford Parser outputs its grammatical structure by grouping words (and phrases) in a tree structure and identifies grammatical roles of words and phrases.

Tregex is a tree query tool that takes Stanford parser trees as input and queries the trees to find subtrees that meet specific rules written in Tregex syntax (Levy and Andrew, 2006). It uses relational operators regulated by Tregex, for example, "A << B" stands for "subtree A dominates subtree B". The operators primarily function in subtree precedence, dominance, negation, regular expression, tree node identity, headship, or variable groups, among others (Levy and Andrew, 2006).

Lu's tool (Lu, 2011), built upon the Stanford Parser and Tregex, does syntactic complexity analysis given textual data. Lu's tool contributed 8 of the initial CB features and 6 of the initial PT features, and we computed the remaining CB and PT features using Perl scripts, the Stanford Parser, and Tregex.

Table 2 lists the sub-set of 17 features (out of 91 features total) that were used for building the scoring models described later (Section 5).

4.2 Feature Selection

We determined the importance of the features by computing each feature's correlation with human raters' proficiency scores based on the training set NN-train. We also used criteria related to the speaking construct, comparisons with native speaker data, and feature inter-correlations. While approaches coming from a pure machine learning perspective would likely use the entire feature pool as input for a classifier, our goal here is to obtain an initial feature set by judicious and careful feature selection that can withstand the scrutiny of construct validity in assessment development.

As noted earlier, the disfluencies in the training set had been removed to obtain a "cleaner" text that looks somewhat more akin to a written passage and is easier to process by NLP modules such as parsers and part-of-speech (POS) taggers.[7] The extracted features partly were taken directly from proposals in the literature and partly were slightly modified to fit our clause annotation scheme. In order to have a unified framework for computing syntactic complexity features, we used a combination of the Stanford Parser and Tregex for computing both clause- and sentence-based features as well as parse-tree-based features, i.e., we did not make use of the human clause boundary label annotations here. The only exception to this

[6] An adjective clause is a clause that functions as an adjective in modifying a noun. E.g., "This cat is a cat that is difficult to deal with."

[7] We are aware that disfluencies can provide valuable clues about spoken proficiency in and of themselves; however, this study is focused exclusively on syntactic complexity analysis, and in this context, disfluencies would distort the picture considerably due to the introduction of parsing errors, e.g.

Name	Type[8]	Meaning	Correlation	Regression
MLS	CB	Mean length of sentences	0.329	0.101
MLT	CB	Mean length of T-units	0.300	-0.059
DC/C	CB	Mean number of dependent clauses per clause	0.291	2.873
SSfreq	CB	Frequency of simple sentences per 1000 words	-.0242	0.001
MLSS	CB	Mean length of simple sentences	0.255	0.040
ADJCfreq	CB	Frequency of adjective clauses per 1000 words	0.253	0.004
Ffreq	CB	Frequency of fragments per 1000 words	-0.386	-0.057
MLCC	CB	Mean length of coordinate clauses	0.224	0.017
CT/T	PT	Mean number of complex T-units per T-unit	0.248	0.908
PP_ling/S	PT	Mean number of linguistically meaningful prepositional phrases (PP) per sentence[9]	0.310	0.423
NP/S	PT	Mean number of noun phrases (NP) per sentence	0.244	-0.411
CN/S	PT	Mean number of complex nominal per sentence	0.325	0.653
VB _ling/T	PT	Mean number of linguistically meaningful[10] verb phrases per T-unit	0.273	-0.780
PAS/S	PT	Mean number of passives per sentence	0.260	1.520
DI/T	PT	Mean number of dependent infinitives per T-unit	0.325	1.550
MLev	PT	Mean number of parsing tree levels per sentence	0.306	-0.134
MPSam	PT	Mean P-based Sampson[11] per sentence	0.254	0.234

Table 2. List of syntactic complexity features selected to be included in building the scoring models.

is that we are using human clause and sentence labels to create a candidate set for the clause boundary features evaluated by the Stanford Parser and Tregex, as explained in the following subsection.

[8] Feature type: CB=Clause boundary based feature type, PT=Parse tree based feature type

[9] A "linguistically meaningful PP" (PP_ling) is defined as a PP immediately dominated by another PP in cases where a preposition contains a noun such as "in spite of" or "in front of". An example would be "she stood in front of a house" where "in front of a house" would be parsed as two embedded PPs but only the top PP would be counted in this case.

[10] A "linguistically meaningful VP" (VP_ling) is defined as a verb phrase immediately dominated by a clausal phrase, in order to avoid VPs embedded in another VP, e.g., "should go to work" is identified as one VP instead of two embedded VPs.

[11] The "P-based Sampson" is a raw production-based measure (Sampson, 1997), defined as "proportion of the daughters of a nonterminal node which are themselves nonterminal and nonrightmost, averaged over the nonterminals of a sentence".

Clause and Sentence based Features (CB features)

Firstly, we extracted all 26 initial CB features directly from human annotated data of NN-train, using information from the clause and sentence type labels. The reasoning behind this was to create an initial pool of clause-based features that reflects the distribution of clauses and sentences as accurately as possible, even though we did not plan to use this extraction method operationally, where the parser decides on clause and sentence types. After computing the values of each CB feature, we calculated correlations between each feature and human-rated scores. Then we created an initial CB feature pool by selecting features that met two criteria: (1) the absolute Pearson correlation coefficient with human scores was larger than 0.2; and (2) the mean value of the feature on non-native speakers was at least 20% lower than that for na-

tive speakers in case of positive correlation and at least by 20% higher than for native speakers in case of negative correlation, using the Nat data set for the latter criterion. Note that all of these features were computed without using a parser. This resulted in 13 important features.

Secondly, Tregex rules were developed based on Lu's tool to extract these 13 CB features from parsing results where the parser is provided with one sentence at a time. By applying the same selection criteria as before, except for allowing for correlations above 0.1 and giving preference to linguistically more meaningful features, we found 8 features that matched our criteria:
MLS, MLT, DC/C, SSfreq, MLSS, ADJCfreq, Ffreq, MLCC

All 28 pairwise inter-correlations between these 8 features were computed and inspected to avoid including features with high inter-correlations in the scoring model. Since we did not find any inter-correlations larger than 0.9, the features were considered moderately independent and none of them were removed from this set so it also maintains linguistic richness for the feature set.

Due to the importance of T-units in complexity analysis, we briefly introduce how we obtain them from annotations. Three types of clauses labeled in our transcript can serve as T-units, including simple sentences, independent clauses, and conjunct (coordination) clauses. These clauses were identified in the human-annotated text and extracted as T-units in this phase. T-units in parse trees are identified using rules in Lu's tool.

Parse Tree based Features (PT features)

We evaluated 65 features in total and selected features with highest importance using the following two criteria (which are very similar as before): (1) the absolute Pearson correlation coefficient with human scores is larger than 0.2; and (2) the feature mean value on native speakers (Nat) is higher than on score 4 for non-native speakers in case of positive correlation, or lower for negative correlation. 20 of 65 features were found to meet the requirements.

Next, we examined inter-correlations between these features and found some correlations larger than 0.85.[12] For each feature pair exhibiting high inter-correlation, we removed one feature according to the criterion that the removed feature should be linguistically less meaningful than the remaining one. After this filtering, the 9 remaining PT features are:
CT/T, PP_ling/S, NP/S, CN/S, VP_ling/T, PAS/S, DI/T, MLev, MPSam

In summary, as a result of the feature selection process, a total of 17 features were identified as important features to be used in scoring models for predicting speakers' proficiency scores. Among them 8 are clause boundary based and the other 9 are parse tree based.

5 Experiments and Results

In the previous section, we identified 17 syntactic features that show promising correlations with human rater speaking proficiency scores. These features as well as the human-rated scores will be used to build scoring models by using machine learning methods. As introduced in Section 3, we have one training set (N=137 speakers with all of their responses combined) for model building and five testing sets (N=52 for each of them) for evaluation.

The publicly available machine learning package Weka was used in our experiments (Hall et al. 2009). We experimented with two algorithms in Weka: multiple regression (called "LinearRegression" in Weka) and decision tree (called "M5P"in Weka). The score values to be predicted are real numbers (i.e., non-integer), because we have to compute the average score of one speaker's responses. Our initial runs showed that decision tree models were consistently outperformed by multiple regression (MR) models and thus decided to only focus on MR models henceforth.

We set the "AttributeSelectionMethod" parameter in Weka's LinearRegression algorithm to all 3 of its possible values in turn: (Model-1) M5 method; (Model-2) no attribute selection; and (Model-3) greedy method. The resulting three multiple regression models were then tested against the five testing sets. Overall, correlations for all models for the NN-test-1-Hum set were between 0.45 and 0.49, correlations for sets NN-test-2-CB and NN-

[12] The reason for using a lower threshold than above was to obtain a roughly equal number of CB and PT features in the end.

test-3-SB (human transcript based, and using automated boundaries) around 0.2, and for sets NN-test-4-ASR-CB and NN-test-5-ASR-SB (ASR hypotheses, and using automated boundaries), the correlations were not significant. Model-2 (using all 17 features) had the highest correlation on NN-test-1-Hum and we provide correlation results of this model in Table 3.

Test set	Correlation coefficient	Correlation significance ($p < 0.05$)
NN-test-1-Hum	0.488	Significant
NN-test-2-CB	0.220	Significant
NN-test-3-SB	0.170	Significant
NN-test-4-ASR-CB	-0.025	Not significant
NN-test-5-ASR-SB	-0.013	Not significant

Table 3. Multiple regression model testing results for Model-2.

6 Discussion

As we can see from the result table (Table 3) in the previous section, using only syntactic complexity features, based on clausal or parse tree information derived from human transcriptions of spoken test responses, can predict holistic human rater scores for combined speaker responses over a whole test with an overall correlation of r=0.49. While this is a promising result for this study with a focus on a broad spectrum of syntactic complexity features, the results also show significant limitations for an immediate operational use of such features. First, the imperfect prediction of clause and sentence boundaries by the two automatic classifiers causes a substantial degradation of scoring model performance to about r=0.2, and secondly, the rather high error rate of the ASR system (50.5%) does not allow for the computation of features that would result in any significant correlation with human scores. We want to note here that while ASR systems can be found that exhibit WERs below 10% for certain tasks, such as restricted dictation in low-noise environments by native speakers, our ASR task is significantly harder in several ways: (1) we have to recognize non-native speakers'rresponses where speakers have a number of different native language backgrounds; (2) the proficiency level of the test takers varies widely; and (3) the responses are spontaneous and unconstrained in terms of vocabulary.

As for the automatic clause and sentence boundary classifiers, we can observe (in Table 4) that although the sentence boundary classifier has a slightly higher F-score than the clause boundary classifier, errors in sentence boundary detection have more negative effects on the accuracy of score prediction than those made by the clause boundary classifier. In fact, the lower F-score of the latter is mainly due to its lower precision which indicates that there are more spurious clause boundaries in its output which apparently cause little harm to the feature extraction processes.

Among the 17 final features, 3 of them are frequency-based and the remaining 14 are ratio-based, which mirrors our findings from previous work that frequency features have been used less successfully than ratio features. As for ratio features, 5 of them are grammatical structure counts against sentence units, 4 are counts against T-units, and only 1 is based on counts against clause units. The feature set covers a wide range of grammatical structures, such as T-units, verb phrases, noun phrases, complex nominals, adjective clauses, coordinate clauses, prepositional phrases, etc. While this wide coverage provides for richness of the construct of syntactic complexity, some of the features exhibit relatively high correlation with each other which reduces their overall contributions to the scoring model's performance.

Going through the workflow of our system, we find at least five major stages that can generate errors which in turn can adversely affect feature computation and scoring model building. Errors may appear in each stage of our workflow, passing or even enlarging their effects from previous stages to later stages:

1) grammatical errors by the speakers (test takers);
2) errors by the ASR system;
3) sentence/clause boundary detection errors;
4) parser errors; and
5) rule extraction errors.

In future work we will need to address each error source to obtain a higher overall system performance.

Classifier	Accuracy	Precision	Recall	F score
Clause boundary	0.954	0.721	0.748	0.734
Sentence boundary	0.975	0.811	0.755	0.782

Table 4. Performance of clause and sentence boundary detectors.

7 Conclusion and Future Work

In this paper, we investigated associations between speakers' syntactic complexity features and their speaking proficiency scores provided by human raters. By exploring empirical evidence from non-native and native speakers' data sets of spontaneous speech test responses, we identified 17 features related to clause types and parse trees as effective predictors of human speaking scores. The features were implemented based on Lu's L2 Syntactic Complexity Analyzer toolkit (Lu, 2011) to be automatically extracted from human or ASR transcripts. Three multiple regression models were built from non-native speech training data with different parameter setup and were tested against five testing sets with different preprocessing steps. The best model used the complete set of 17 features and exhibited a correlation with human scores of r=0.49 on human transcripts with boundary annotations.

When using automated classifiers to predict clause or sentence boundaries, correlations with human scores are around r=0.2. Our experiments indicate that by enhancing the accuracy of the two main automated preprocessing components, namely ASR and automatic sentence and clause boundary detectors, scoring model performance will increase substantially, as well. Furthermore, this result demonstrates clearly that syntactic complexity features can be devised that are able to predict human speaking proficiency scores.

Since this is a preliminary study, there is ample space to improve all major stages in the feature extraction process. The errors listed in the previous section are potential working directions for preprocessing enhancements prior to machine learning. Among the five types of errors, we can work on improving the accuracy of the speech recognizer, sentence and clause boundary detectors, parser, and feature extraction rules; as for the grammatical errors produced by test takers, we are envisioning to automatically identify and correct such errors. We will further experiment with syntactic complexity measures to balance construct richness and model simplicity. Furthermore, we can also experiment with additional types of machine learning models and tune parameters to derive scoring models with better performance.

Acknowledgements

The authors wish to thank Lei Chen and Su-Youn Yoon for their help with the sentence and clause boundary classifiers. We also would like to thank our colleagues Jill Burstein, Keelan Evanini, Yoko Futagi, Derrick Higgins, Nitin Madnani, and Joel Tetreault, as well as the four anonymous ACL reviewers for their valuable and helpful feedback and comments on our paper.

References

Bachman, L.F. (1990). Fundamental considerations in language testing. Oxford: Oxford University Press.

Bernstein, J. (1999). PhonePass testing: Structure and construct. Menlo Park, CA: Ordinate Corporation.

Bernstein, J., DeJong, J., Pisoni, D. & Townshend, B. (2000). Two experiments in automatic scoring of spoken language proficiency. Proceedings of In-STILL 2000, Dundee, Scotland.

Bernstein, J., Cheng, J., & Suzuki, M. (2010). Fluency and structural complexity as predictors of L2 oral proficiency. Proceedings of Interspeech 2010, Tokyo, Japan, September.

Chen, L., Tetreault, J. & Xi, X. (2010). Towards using structural events to assess non-native speech. NAACL-HLT 2010. 5th Workshop on Innovative Use of NLP for Building Educational Applications, Los Angeles, CA, June.

Condouris, K., Meyer, E. & Tagger-Flusberg, H. (2003). The relationship between standardized measures of language and measures of spontaneous speech in children with autism. American Journal of Speech-Language Pathology, 12(3), 349-358.

Cooper, T.C. (1976). Measuring written syntactic patterns of second language learners of German. The Journal of Educational Research, 69(5), 176-183.

Cucchiarini, C., Strik, H. & Boves, L. (1997). Automatic evaluation of Dutch pronunciation by using speech recognition technology. IEEE Automatic Speech Recognition and Understanding Workshop, Santa Barbara, CA.

Cucchiarini, C., Strik, H. & Boves, L. (2000). Quantitative assessment of second language learners' fluency by means of automatic speech recognition technology. Journal of the Acoustical Society of America, 107, 989-999.

Franco, H., Abrash, V., Precoda, K., Bratt, H., Rao, R. & Butzberger, J. (2000a). The SRI EduSpeak system: Recognition and pronunciation scoring for language learning. Proceedings of InSTiLL-2000 (Intelligent Speech Technology in Language Learning), Dundee, Scotland.

Franco, H., Neumeyer, L., Digalakis, V. & Ronen, O. (2000b). Combination of machine scores for automatic grading of pronunciation quality. Speech Communication, 30, 121-130.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. & Witten, I.H. (2009). The WEKA Data Mining Software: An Update. SIGKDD Explorations, 11(1).

Halleck, G.B. (1995). Assessing oral proficiency: A comparison of holistic and objective measures. The Modern Language Journal, 79(2), 223-234.

Henry, K. (1996). Early L2 writing development: A study of autobiographical essays by university-level students on Russian. The Modern Language Journal, 80(3), 309-326.

Ho-Peng, L. (1983). Using T-unit measures to assess writing proficiency of university ESL students. RELC Journal, 14(2), 35-43.

Hunt, K. (1965). Grammatical structures written at three grade levels. NCTE Research report No.3. Champaign, IL: NCTE.

Iwashita, N. (2006). Syntactic complexity measures and their relations to oral proficiency in Japanese as a foreign language. Language Assessment Quarterly, 3(20), 151-169.

Kameen, P.T. (1979). Syntactic skill and ESL writing quality. In C. Yorio, K. Perkins, & J. Schachter (Eds.), On TESOL '79: The learner in focus (pp.343-364). Washington, D.C.: TESOL.

Klein, D. & Manning, C.D. (2003). Fast exact inference with a factored model for a natural language parsing. In S.Becker, S. Thrun & K. Obermayer (Eds.), Advances in Neural Information Processing Systems 15 (pp.3-10). Cambridge, MA: MIT Press.

Larsen-Freeman, D. (1978). An ESL index of development. Teachers of English to Speakers of Other Languages Quarterly, 12(4), 439-448.

Levy, R. & Andrew, G. (2006). Tregex and Tsurgeon: Tools for querying and manipulating tree data structures. Proceedings of the Fifth International Conference on Language Resources and Evaluation.

Lu, X. (2010). Automatic analysis of syntactic complexity in second language writing. International Journal of Corpus Linguistics, 15(4), 474-496.

Lu, X. (2011). L2 Syntactic Complexity Analyzer. Retrieved from http://www.personal.psu.edu/xxl13/downloads/l2sca.html

Ortega, L. (2003). Syntactic complexity measures and their relationship to L2 proficiency: A research synthesis of college-level L2 writing. Applied Linguistics, 24(4), 492-518.

Perkins, K. (1980). Using objective methods of attained writing proficiency to discriminate among holistic evaluations. Teachers of English to Speakers of Other Languages Quarterly, 14(1), 61-69.

Roll, M., Frid, J. & Horne, M. (2007). Measuring syntactic complexity in spontaneous spoken Swedish. Language and Speech, 50(2), 227-245.

Sampson, G. (1997). Depth in English grammar. Journal of Linguistics, 33, 131-151.

Wolfe-Quintero, K., Inagaki, S. & Kim, H. Y. (1998). Second language development in writing: Measures of fluency, accuracy, & complexity. Honolulu, HI: University of Hawaii Press.

Xi, X., & Mollaun, P. (2006). Investigating the utility of analytic scoring for the TOEFL® Academic Speaking Test (TAST). TOEFL iBT Research Report No. TOEFLiBT-01.

Zechner, K., Higgins, D. & Xi, X. (2007). SpeechRater(SM): A construct-driven approach to score spontaneous non-native speech. Proceedings of the 2007 Workshop of the International Speech Communication Association (ISCA) Special Interest Group on Speech and Language Technology in Education (SLaTE), Farmington, PA, October.

Zechner, K., Higgins, D., Xi, X, & Williamson, D.M. (2009). Automatic scoring of non-native spontaneous speech in tests of spoken English. Speech Communication, 51 (10), October.

N-Best Rescoring Based on Pitch-accent Patterns

Je Hun Jeon[1] **Wen Wang**[2] **Yang Liu**[1]

[1]Department of Computer Science, The University of Texas at Dallas, USA

[2]Speech Technology and Research Laboratory, SRI International, USA

{jhjeon,yangl}@hlt.utdallas.edu, wwang@speech.sri.com

Abstract

In this paper, we adopt an n-best rescoring scheme using pitch-accent patterns to improve automatic speech recognition (ASR) performance. The pitch-accent model is decoupled from the main ASR system, thus allowing us to develop it independently. N-best hypotheses from recognizers are rescored by additional scores that measure the correlation of the pitch-accent patterns between the acoustic signal and lexical cues. To test the robustness of our algorithm, we use two different data sets and recognition setups: the first one is English radio news data that has pitch accent labels, but the recognizer is trained from a small amount of data and has high error rate; the second one is English broadcast news data using a state-of-the-art SRI recognizer. Our experimental results demonstrate that our approach is able to reduce word error rate relatively by about 3%. This gain is consistent across the two different tests, showing promising future directions of incorporating prosodic information to improve speech recognition.

1 Introduction

Prosody refers to the suprasegmental features of natural speech, such as rhythm and intonation, since it normally extends over more than one phoneme segment. Speakers use prosody to convey paralinguistic information such as emphasis, intention, attitude, and emotion. Humans listening to speech with natural prosody are able to understand the content with low cognitive load and high accuracy. However, most modern ASR systems only use an acoustic model and a language model. Acoustic information in ASR is represented by spectral features that are usually extracted over a window length of a few tens of milliseconds. They miss useful information contained in the prosody of the speech that may help recognition.

Recently a lot of research has been done in automatic annotation of prosodic events (Wightman and Ostendorf, 1994; Sridhar et al., 2008; Ananthakrishnan and Narayanan, 2008; Jeon and Liu, 2009). They used acoustic and lexical-syntactic cues to annotate prosodic events with a variety of machine learning approaches and achieved good performance. There are also many studies using prosodic information for various spoken language understanding tasks. However, research using prosodic knowledge for speech recognition is still quite limited. In this study, we investigate leveraging prosodic information for recognition in an n-best rescoring framework.

Previous studies showed that prosodic events, such as pitch-accent, are closely related with acoustic prosodic cues and lexical structure of utterance. The pitch-accent pattern given acoustic signal is strongly correlated with lexical items, such as syllable identity and canonical stress pattern. Therefore as a first study, we focus on pitch-accent in this paper. We develop two separate pitch-accent detection models, using acoustic (observation model) and lexical information (expectation model) respectively, and propose a scoring method for the correlation of pitch-accent patterns between the two models for recognition hypotheses. The n-best list is rescored using the pitch-accent matching scores

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 732–741,
Portland, Oregon, June 19-24, 2011.

combined with the other scores from the ASR system (acoustic and language model scores). We show that our method yields a word error rate (WER) reduction of about 3.64% and 2.07% relatively on two baseline ASR systems, one being a state-of-the-art recognizer for the broadcast news domain. The fact that it holds across different baseline systems suggests the possibility that prosody can be used to help improve speech recognition performance.

The remainder of this paper is organized as follows. In the next section, we review previous work briefly. Section 3 explains the models and features for pitch-accent detection. We provide details of our n-best rescoring approach in Section 4. Section 5 describes our corpus and baseline ASR setup. Section 6 presents our experiments and results. The last section gives a brief summary along with future directions.

2 Previous Work

Prosody is of interest to speech researchers because it plays an important role in comprehension of spoken language by human listeners. The use of prosody in speech understanding applications has been quite extensive. A variety of applications have been explored, such as sentence and topic segmentation (Shriberg et al., 2000; Rosenberg and Hirschberg, 2006), word error detection (Litman et al., 2000), dialog act detection (Sridhar et al., 2009), speaker recognition (Shriberg et al., 2005), and emotion recognition (Benus et al., 2007), just to name a few.

Incorporating prosodic knowledge is expected to improve the performance of speech recognition. However, how to effectively integrate prosody within the traditional ASR framework is a difficult problem, since prosodic features are not well defined and they come from a longer region, which is different from spectral features used in current ASR systems. Various research has been conducted trying to incorporate prosodic information in ASR. One way is to directly integrate prosodic features into the ASR framework (Vergyri et al., 2003; Ostendorf et al., 2003; Chen and Hasegawa-Johnson, 2006). Such efforts include prosody dependent acoustic and pronunciation model (allophones were distinguished according to different prosodic phenomenon), language model (words were augmented by prosody events), and duration modeling (different prosodic events were modeled separately and combined with conventional HMM). This kind of integration has advantages in that spectral and prosodic features are more tightly coupled and jointly modeled. Alternatively, prosody was modeled independently from the acoustic and language models of ASR and used to rescore recognition hypotheses in the second pass. This approach makes it possible to independently model and optimize the prosodic knowledge and to combine with ASR hypotheses without any modification of the conventional ASR modules. In order to improve the rescoring performance, various prosodic knowledge was studied. (Ananthakrishnan and Narayanan, 2007) used acoustic pitch-accent pattern and its sequential information given lexical cues to rescore n-best hypotheses. (Kalinli and Narayanan, 2009) used acoustic prosodic cues such as pitch and duration along with other knowledge to choose a proper word among several candidates in confusion networks. Prosodic boundaries based on acoustic cues were used in (Szaszak and Vicsi, 2007).

We take a similar approach in this study as the second approach above in that we develop prosodic models separately and use them in a rescoring framework. Our proposed method differs from previous work in the way that the prosody model is used to help ASR. In our approach, we explicitly model the symbolic prosodic events based on acoustic and lexical information. We then capture the correlation of pitch-accent patterns between the two different cues, and use that to improve recognition performance in an n-best rescoring paradigm.

3 Prosodic Model

Among all the prosodic events, we use only pitch-accent pattern in this study, because previous studies have shown that acoustic pitch-accent is strongly correlated with lexical items, such as canonical stress pattern and syllable identity that can be easily acquired from the output of conventional ASR and pronunciation dictionary. We treat pitch-accent detection as a binary classification task, that is, a classifier is used to determine whether the base unit is prominent or not. Since pitch-accent is usually

carried by syllables, we use syllables as our units, and the syllable definition of each word is based on CMU pronunciation dictionary which has lexical stress and syllable boundary marks (Bartlett et al., 2009). We separately develop acoustic-prosodic and lexical-prosodic models and use the correlation between the two models for each syllable to rescore the n-best hypotheses of baseline ASR systems.

3.1 Acoustic-prosodic Features

Similar to most previous work, the prosodic features we use include pitch, energy, and duration. We also add delta features of pitch and energy. Duration information for syllables is derived from the speech waveform and phone-level forced alignment of the transcriptions. In order to reduce the effect by both inter-speaker and intra-speaker variation, both pitch and energy values are normalized (z-value) with utterance specific means and variances. For pitch, energy, and their delta values, we apply several categories of 12 functions to generate derived features.

- Statistics (7): minimum, maximum, range, mean, standard deviation, skewness and kurtosis value. These are used widely in prosodic event detection and emotion detection.
- Contour (5): This is approximated by taking 5 leading terms in the Legendre polynomial expansion. The approximation of the contour using the Legendre polynomial expansion has been successfully applied in quantitative phonetics (Grabe et al., 2003) and in engineering applications (Dehak et al., 2007). Each term models a particular aspect of the contour, such as the slope, and information about the curvature.

We use 6 duration features, that is, raw, normalized, and relative durations (ms) of the syllable and vowel. Normalization (z-value) is performed based on statistics for each syllable and vowel. The relative value is the difference between the normalized current duration and the following one.

In the above description, we assumed that the event of a syllable is only dependent on its observations, and did not consider contextual effect. To alleviate this restriction, we expand the features by incorporating information about the neighboring syllables. Based on the study in (Jeon and Liu, 2010) that evaluated using left and right contexts, we choose to use one previous and one following context in the features. The total number of features used in this study is 162.

3.2 Lexical-prosodic Features

There is a very strong correlation between pitch-accent in an utterance and its lexical information. Previous studies have shown that the lexical features perform well for pitch-accent prediction. The detailed features for training the lexical-prosodic model are as follows.

- Syllable identity: We kept syllables that appear more than 5 times in the training corpus. The other syllables that occur less are collapsed into one syllable representation.
- Vowel phone identity: We used vowel phone identity as a feature.
- Lexical stress: This is a binary feature to represent if the syllable corresponds to a lexical stress based on the pronunciation dictionary.
- Boundary information: This is a binary feature to indicate if there is a word boundary before the syllable.

For lexical features, based on the study in (Jeon and Liu, 2010), we added two previous and two following contexts in the final features.

3.3 Prosodic Model Training

We choose to use a support vector machine (SVM) classifier[1] for the prosodic model based on previous work on prosody labeling study in (Jeon and Liu, 2010). We use RBF kernel for the acoustic model, and 3-order polynomial kernel for the lexical model.

In our experiments, we investigate two kinds of training methods for prosodic modeling. The first one is a supervised method where models are trained using all the labeled data. The second is a semi-supervised method using co-training algorithm (Blum and Mitchell, 1998), described in Algorithm 1. Given a set L of labeled data and a set U of unlabeled data with two views, it then iterates in the

[1]LIBSVM – A Library for Support Vector Machines, location: http://www.csie.ntu.edu.tw/˜cjlin/libsvm/

Algorithm 1 Co-training algorithm.

Given:

- L: labeled examples; U: unlabeled examples
- there are two views V_1 and V_2 on an example x

Initialize:

- L_1=L, samples used to train classifiers h_1
- L_2=L, samples used to train classifiers h_2

Loop for *k* iterations

- create a small pool U' choosing from U
- use $V_1(L_1)$ to train classifier h_1 and $V_2(L_2)$ to train classifier h_2
- let h_1 label/select examples D_{h_1} from U'
- let h_2 label/select examples D_{h_2} from U'
- add self-labeled examples D_{h_1} to L_2 and D_{h_2} to L_1
- remove D_{h_1} and D_{h_2} from U

following procedure. The algorithm first creates a smaller pool U' containing unlabeled data from U. It uses L_i $(i = 1, 2)$ to train two distinct classifiers: the acoustic classifier h_1, and the lexical classifier h_2. We use function V_i $(i = 1, 2)$ to represent that only a single view is used for training h_1 or h_2. These two classifiers are used to make predictions for the unlabeled set U', and only when they agree on the prediction for a sample, their predicted class is used as the label for this sample. Then among these self-labeled samples, the most confident ones by one classifier are added to the data set L_i for training the other classifier. This iteration continues until reaching the defined number of iterations. In our experiment, the size of the pool U' is 5 times of the size of training data L_i, and the size of the added self-labeled example set, D_{h_i}, is 5% of L_i. For the newly selected D_{h_i}, the distribution of the positive and negative examples is the same as that of the training data L_i.

This co-training method is expected to cope with two problems in prosodic model training. The first problem is the different decision patterns between the two classifiers: the acoustic model has relatively higher precision, while the lexical model has relatively higher recall. The goal of the co-training algorithm is to learn from the difference of each classifier, thus it can improve the performance as well as reduce the mismatch of two classifiers. The second problem is the mismatch of data used for model training and testing, which often results in system performance degradation. Using co-training, we can use the unlabeled data from the domain that matches the test data, adapting the model towards test domain.

4 N-Best Rescoring Scheme

In order to leverage prosodic information for better speech recognition performance, we augment the standard ASR equation to include prosodic information as following:

$$\begin{aligned} \hat{W} &= \arg\max_W p(W|A_s, A_p) \\ &= \arg\max_W p(A_s, A_p|W)p(W) \end{aligned} \quad (1)$$

where A_s and A_p represent acoustic-spectral features and acoustic-prosodic features. We can further assume that spectral and prosodic features are conditionally independent given a word sequence W, therefore, Equation 1 can be rewritten as following:

$$\hat{W} \approx \arg\max_W p(A_s|W)p(W)p(A_p|W) \quad (2)$$

The first two terms stand for the acoustic and language models in the original ASR system, and the last term means the prosody model we introduce. Instead of using the prosodic model in the first pass decoding, we use it to rescore n-best candidates from a speech recognizer. This allows us to train the prosody models independently and better optimize the models.

For $p(A_p|W)$, the prosody score for a word sequence W, in this work we propose a method to estimate it, also represented as $score_{W-prosody}(W)$. The idea of scoring the prosody patterns is that there is some expectation of pitch-accent patterns given the lexical sequence (W), and the acoustic pitch-accent should match with this expectation. For instance, in the case of a prominent syllable, both acoustic and lexical evidence show pitch-accent, and vice versa. In order to maximize the agreement between the two sources, we measure how good the acoustic pitch-accent in speech signal matches the given lexical cues. For each syllable S_i in the n-best list, we use acoustic-prosodic cues (a_i) to estimate the posterior probability that the syllable is prominent (*P*), $p(P|a_i)$. Similarly, we use lexical cues (l_i)

to determine the syllable's pitch-accent probability $p(P|l_i)$. Then the prosody score for a syllable S_i is estimated by the match of the pitch-accent patterns between acoustic and lexical information using the difference of the posteriors from the two models:

$$score_{S-prosody}(S_i) \approx 1 - \mid p(P|a_i) - p(P|l_i) \mid \quad (3)$$

Furthermore, we take into account the effect due to varying durations for different syllables. We notice that syllables without pitch-accent have much shorter duration than the prominent ones, and the prosody scores for the short syllables tend to be high. This means that if a syllable is split into two consecutive non-prominent syllables, the agreement score may be higher than a long prominent syllable. Therefore, we introduce a weighting factor based on syllable duration ($dur(i)$). For a candidate word sequence (W) consisting of n syllables, its prosodic score is the sum of the prosodic scores for all the syllables in it weighted by their duration (measured using milliseconds), that is:

$$score_{W-prosody}(W) \approx \sum_{i=1}^{n} log(score_{S-prosody}(S_i)) \cdot dur(i) \quad (4)$$

We then combine this prosody score with the original acoustic and language model likelihood ($P(A_s|W)$ and $P(W)$ in Equation 2). In practice, we need to weight them differently, therefore, the combined score for a hypothesis W is:

$$\begin{aligned} Score(W) &= \lambda \cdot score_{W-prosody}(W) \\ &\quad + score_{ASR}(W) \end{aligned} \quad (5)$$

where $score_{ASR}(W)$ is generated by ASR systems (composed of acoustic and language model scores) and λ is optimized using held out data.

5 Data and Baseline Systems

Our experiments are carried out using two different data sets and two different recognition systems as well in order to test the robustness of our proposed method.

The first data set is the Boston University Radio News Corpus (BU) (Ostendorf et al., 1995), which consists of broadcast news style read speech. The BU corpus has about 3 hours of read speech from 7 speakers (3 female, 4 male). Part of the data has been labeled with ToBI-style prosodic annotations. In fact, the reason that we use this corpus, instead of other corpora typically used for ASR experiments, is because of its prosodic labels. We divided the entire data corpus into a training set and a test set. There was no speaker overlap between training and test sets. The training set has 2 female speakers (*f2* and *f3*) and 3 male ones (*m2*, *m3*, *m4*). The test set is from the other two speakers (*f1* and *m1*). We use 200 utterances for the recognition experiments. Each utterance in BU corpus consists of more than one sentences, so we segmented each utterance based on pause, resulting in a total number of 713 segments for testing. We divided the test set roughly equally into two sets, and used one for parameter tuning and the other for rescoring test. The recognizer used for this data set was based on Sphinx-3[2]. The context-dependent triphone acoustic models with 32 Gaussian mixtures were trained using the training partition of the BU corpus described above, together with the broadcast new data. A standard back-off trigram language model with Kneser-Ney smoothing was trained using the combined text from the training partition of the BU, Wall Street Journal data, and part of Gigaword corpus. The vocabulary size was about 10K words and the out-of-vocabulary (OOV) rate on the test set was 2.1%.

The second data set is from broadcast news (BN) speech used in the GALE program. The recognition test set contains 1,001 utterances. The n-best hypotheses for this data set are generated by a state-of-the-art SRI speech recognizer, developed for broadcast news speech (Stolcke et al., 2006; Zheng et al., 2007). This system yields much better performance than the first one. We also divided the test set roughly equally into two sets for parameter tuning and testing. From the data used for training the speech recognizer, we randomly selected 5.7 hours of speech (4,234 utterances) for the co-training algorithm for the prosodic models.

For prosodic models, we used a simple binary representation of pitch-accent in the form of presence versus absence. The reference labels are de-

[2]CMU Sphinx - Speech Recognition Toolkit, location: http://www.speech.cs.cmu.edu/sphinx/tutorial.html

rived from the ToBI annotation in the BU corpus, and the ratio of pitch-accented syllables is about 34%. Acoustic-prosodic and lexical-prosodic models were separately developed using the features described in Section 3. Feature extraction was performed at the syllable level from force-aligned data. For the supervised approach, we used those utterances in the training data partition with ToBI labels in the BU corpus (245 utterances, 14,767 syllables). For co-training, the labeled data from BU corpus is used as initial training, and the other unlabeled data from BU and BN are used as unlabeled data.

	Accuracy(%)		**Prosody score**	
	Acoustic	Lexical	Pos	Neg
Supervised	83.97	84.48	0.747	0.852
Co-training	84.54	84.99	0.771	0.867
Reference	83.53	87.92	-	-

Table 1: Pitch accent detection results: performance of individual acoustic and lexical models, and the agreement between the two models (i.e., prosody score for a syllable, Equation 3) for positive and negative classes. Also shown is the reference result for pitch accent detection from Jeon and Liu (2009).

6 Experimental Results

6.1 Pitch-accent Detection

First we evaluate the performance of our acoustic-prosodic and lexical-prosodic models for pitch-accent detection. For rescoring, not only the accuracies of the two individual prosodic models are important, but also the pitch-accent agreement score between the two models (as shown in Equation 3) is critical, therefore, we present results using these two metrics. Table 1 shows the accuracy of each model for pitch-accent detection, and also the average prosody score of the two models (i.e., Equation 3) for positive and negative classes (using reference labels). These results are based on the BU labeled data in the test set. To compare our pitch accent detection performance with previous work, we include the result of (Jeon and Liu, 2009) as a reference. Compared to previous work, the acoustic model achieved similar performance, while the performance of lexical model is a bit lower. The lower performance of lexical model is mainly because we do not use part-of-speech (POS) information in the features, since we want to only use the word output from the ASR system (without additional POS tagging).

As shown in Table 1, when using the co-training algorithm, as described in Section 3.3, the overall accuracies improve slightly and therefore the prosody score is also increased. We expect this improved model will be more beneficial for rescoring.

6.2 N-Best Rescoring

For the rescoring experiment, we use 100-best hypotheses from the two different ASR systems, as described in Section 5. We apply the acoustic and lexical prosodic models to each hypothesis to obtain its prosody score, and combine it with ASR scores to find the top hypothesis. The weights were optimized using one test set and applied to the other. We report the average result of the two testings.

Table 2 shows the rescoring results using the first recognition system on BU data, which was trained with a relatively small amount of data. The 1-best baseline uses the first hypothesis that has the best ASR score. The oracle result is from the best hypothesis that gives the lowest WER by comparing all the candidates to the reference transcript. We used two prosodic models as described in Section 3.3. The first one is the base prosodic model using supervised training (*S-model*). The second is the prosodic model with the co-training algorithm (*C-model*). For these rescoring experiments, we tuned λ (in Equation 5) when combining the ASR acoustic and language model scores with the additional prosody score. The value in parenthesis in Table 2 means the relative WER reduction when compared to the baseline result. We show the WER results for both the development and the test set.

As shown in Table 2, we observe performance improvement using our rescoring method. Using the base *S-model* yields reasonable improvement, and *C-model* further reduces WER. Even though the prosodic event detection performance of these two prosodic models is similar, the improved prosody score between the acoustic and lexical prosodic models using co-training helps rescoring. After rescoring using prosodic knowledge, the WER is reduced by 0.82% (3.64% relative). Furthermore, we notice that the difference between development and

		WER (%)
1-best baseline		22.64
S-model	Dev	21.93 (3.11%)
	Test	22.10 (2.39%)
C-model	Dev	21.76 (3.88%)
	Test	21.81 (3.64%)
Oracle		15.58

Table 2: WER of the baseline system and after rescoring using prosodic models. Results are based on the first ASR system.

test data is smaller when using the *C-model* than *S-model*, which means that the prosodic model with co-training is more stable. In fact, we found that the optimal value of λ is 94 and 57 for the two folds using *S-model*, and is 99 and 110 for the *C-model*. These verify again that the prosodic scores contribute more in the combination with ASR likelihood scores when using the *C-model*, and are more robust across different tuning sets. Ananthakrishnan and Narayanan (2007) also used acoustic/lexical prosodic models to estimate a prosody score and reported 0.3% recognition error reduction on BU data when rescoring 100-best list (their baseline WER is 22.8%). Although there is some difference in experimental setup (data, classifier, features) between ours and theirs, our *S-model* showed comparable performance gain and the result of *C-model* is significantly better than theirs.

Next we test our n-best rescoring approach using a state-of-the-art SRI speech recognizer on BN data to verify if our approach can generalize to better ASR n-best lists. This is often the concern that improvements observed on a poor ASR system do not hold for better ASR systems. The rescoring results are shown in Table 3. We can see that the baseline performance of this recognizer is much better than that of the first ASR system (even though the recognition task is also harder). Our rescoring approach still yields performance gain even using this state-of-the-art system. The WER is reduced by 0.29% (2.07% relative). This error reduction is lower than that in the first ASR system. There are several possible reasons. First, the baseline ASR performance is higher, making further improvement hard; second, and more importantly, the prosody models do not match well to the test domain. We trained the prosody model using the BU data. Even though co-training is used to leverage unlabeled BN data to reduce data mismatch, it is still not as good as using labeled in-domain data for model training.

		WER (%)
1-best baseline		13.77
S-model	Dev	13.53 (1.78%)
	Test	13.55 (1.63%)
C-model	Dev	13.48 (2.16%)
	Test	13.49 (2.07%)
Oracle		9.23

Table 3: WER of the baseline system and after rescoring using prosodic models. Results are based on the second ASR system.

6.3 Analysis and Discussion

We also analyze what kinds of errors are reduced using our rescoring approach. Most of the error reduction came from substitution and insertion errors. Deletion error rate did not change much or sometimes even increased. For a better understanding of the improvement using the prosody model, we analyzed the pattern of corrections (the new hypothesis after rescoring is correct while the original 1-best is wrong) and errors. Table 4 shows some positive and negative examples from rescoring results using the first ASR system. In this table, each word is associated with some binary expressions inside a parenthesis, which stand for pitch-accent markers. Two bits are used for each syllable: the first one is for the acoustic-prosodic model and the second one is for the lexical-prosodic model. For both bits, 1 represents pitch-accent, and 0 indicates none. These hard decisions are obtained by setting a threshold of 0.5 for the posterior probabilities from the acoustic or lexical models. For example, when the acoustic classifier predicts a syllable as pitch-accented and the lexical one as not accented, '*10*' marker is assigned to the syllable. The number of such pairs of pitch-accent markers is the same as the number of syllables in a word. The bold words indicate correct words and italic means errors. As shown in the positive example of Table 4, we find that our prosodic model is effective at identifying an erroneous word when it is split into two words, resulting in different pitch-accent patterns. Language models are

Positive example	1-best :	most	*of*	*the*	massachusetts
		(11)	***(10)***	***(00)***	(11 00 01 00)
	rescored :	most	**other**		massachusetts
		(11)	**(11 00)**		(11 00 01 00)
Negative example	1-best :	robbery	and	**on**	**a** theft
		(11 00 00)	(00)	**(10)**	**(00)** (11)
	rescored :	robbery	and	*lot*	*of* theft
		(11 00 00)	(00)	***(11)***	***(00)*** (11)

Table 4: Examples of rescoring results. Binary expressions inside the parenthesis below a word represent pitch-accent markers for the syllables in the word.

not good at correcting this kind of errors since both word sequences are plausible. Our model also introduces some errors, as shown in the negative example, which is mainly due to the inaccurate prosody model.

We conducted more prosody rescoring experiments in order to understand the model behavior. These analyses are based on the n-best list from the first ASR system for the entire test set. In the first experiment, among the 100 hypotheses in n-best list, we gave a prosody score of 0 to the 100^{th} hypothesis, and used automatically obtained prosodic scores for the other hypotheses. A zero prosody score means the perfect agreement given acoustic and lexical cues. The original scores from the recognizer were combined with the prosodic scores for rescoring. This was to verify that the range of the weighting factor λ estimated on the development data (using the original, not the modified prosody scores for all candidates) was reasonable to choose proper hypothesis among all the candidates. We noticed that 27% of the times the last hypothesis on the list was selected as the best hypothesis. This hypothesis has the highest prosodic scores, but lowest ASR score. This result showed that if the prosodic models were accurate enough, the correct candidate could be chosen using our rescoring framework.

In the second experiment, we put the reference text together with the other candidates. We use the same ASR scores for all candidates, and generated prosodic scores using our prosody model. This was to test that our model could pick up correct candidate using only the prosodic score. We found that for 26% of the utterances, the reference transcript was chosen as the best one. This was significantly better than random selection (i.e., 1/100), suggesting the benefit of the prosody model; however, this percentage is not very high, implying the limitation of prosodic information for ASR or the current imperfect prosodic models.

In the third experiment, we replaced the 100^{th} candidate with the reference transcript and kept its ASR score. When using our prosody rescoring approach, we obtained a relative error rate reduction of 6.27%. This demonstrates again that our rescoring method works well – if the correct hypothesis is on the list, even though with a low ASR score, using prosodic information can help identify the correct candidate.

Overall the performance improvement we obtained from rescoring by incorporating prosodic information is very promising. Our evaluation using two different ASR systems shows that the improvement holds even when we use a state-of-the-art recognizer and the training data for the prosody model does not come from the same corpus. We believe the consistent improvements we observed for different conditions show that this is a direction worthy of further investigation.

7 Conclusion

In this paper, we attempt to integrate prosodic information for ASR using an n-best rescoring scheme. This approach decouples the prosodic model from the main ASR system, thus the prosodic model can be built independently. The prosodic scores that we use for n-best rescoring are based on the matching of pitch-accent patterns by acoustic and lexical features. Our rescoring method achieved a WER reduction of 3.64% and 2.07% relatively using two different ASR systems. The fact that the gain holds across different baseline systems (including a state-of-the-

art speech recognizer) suggests the possibility that prosody can be used to improve speech recognition performance.

As suggested by our experiments, better prosodic models can result in more WER reduction. The performance of our prosodic model was improved with co-training, but there are still problems, such as the imbalance of the two classifiers' prediction, as well as for the two events. In order to address these problems, we plan to improve the labeling and selection method in the co-training algorithm, and also explore other training algorithms to reduce domain mismatch. Furthermore, we are also interested in evaluating our approach on the spontaneous speech domain, which is quite different from the data we used in this study.

In this study, we used n-best rather than lattice rescoring. Since the prosodic features we use include cross-word contextual information, it is not straightforward to apply it directly to lattices. In our future work, we will develop models with only within-word context, and thus allowing us to explore lattice rescoring, which we expect will yield more performance gain.

References

Sankaranarayanan Ananthakrishnan and Shrikanth Narayanan. 2007. Improved speech recognition using acoustic and lexical correlated of pitch accent in a n-best rescoring framework. *Proc. of ICASSP*, pages 65–68.

Sankaranarayanan Ananthakrishnan and Shrikanth Narayanan. 2008. Automatic prosodic event detection using acoustic, lexical and syntactic evidence. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(1):216–228.

Susan Bartlett, Grzegorz Kondrak, and Colin Cherry. 2009. On the syllabification of phonemes. *Proc. of NAACL-HLT*, pages 308–316.

Stefan Benus, Agustín Gravano, and Julia Hirschberg. 2007. Prosody, emotions, and whatever. *Proc. of Interspeech*, pages 2629–2632.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. *Proc. of the Workshop on Computational Learning Theory*, pages 92–100.

Ken Chen and Mark Hasegawa-Johnson. 2006. Prosody dependent speech recognition on radio news corpus of American English. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):232– 245.

Najim Dehak, Pierre Dumouchel, and Patrick Kenny. 2007. Modeling prosodic features with joint factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(7):2095–2103.

Esther Grabe, Greg Kochanski, and John Coleman. 2003. Quantitative modelling of intonational variation. *Proc. of SASRTLM*, pages 45–57.

Je Hun Jeon and Yang Liu. 2009. Automatic prosodic events detection suing syllable-based acoustic and syntactic features. *Proc. of ICASSP*, pages 4565–4568.

Je Hun Jeon and Yang Liu. 2010. Syllable-level prominence detection with acoustic evidence. *Proc. of Interspeech*, pages 1772–1775.

Ozlem Kalinli and Shrikanth Narayanan. 2009. Continuous speech recognition using attention shift decoding with soft decision. *Proc. of Interspeech*, pages 1927–1930.

Diane J. Litman, Julia B. Hirschberg, and Marc Swerts. 2000. Predicting automatic speech recognition performance using prosodic cues. *Proc. of NAACL*, pages 218–225.

Mari Ostendorf, Patti Price, and Stefanie Shattuck-Hufnagel. 1995. The Boston University radio news corpus. *Linguistic Data Consortium*.

Mari Ostendorf, Izhak Shafran, and Rebecca Bates. 2003. Prosody models for conversational speech recognition. *Proc. of the 2nd Plenary Meeting and Symposium on Prosody and Speech Processing*, pages 147–154.

Andrew Rosenberg and Julia Hirschberg. 2006. Story segmentation of broadcast news in English, Mandarin and Arabic. *Proc. of HLT-NAACL*, pages 125–128.

Elizabeth Shriberg, Andreas Stolcke, Dilek Hakkani-Tür, and Gökhan Tür. 2000. Prosody-based automatic segmentation of speech into sentences and topics. *Speech Communication*, 32(1-2):127–154.

Elizabeth Shriberg, Luciana Ferrer, Sachin S. Kajarekar, Anand Venkataraman, and Andreas Stolcke. 2005. Modeling prosodic feature sequences for speaker recognition. *Speech Communication*, 46(3-4):455–472.

Vivek Kumar Rangarajan Sridhar, Srinivas Bangalore, and Shrikanth S. Narayanan. 2008. Exploiting acoustic and syntactic features for automatic prosody labeling in a maximum entropy framework. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(4):797–811.

Vivek Kumar Rangarajan Sridhar, Srinivas Bangalore, and Shrikanth Narayanan. 2009. Combining lexical, syntactic and prosodic cues for improved online

dialog act tagging. *Computer Speech and Language*, 23(4):407–422.

Andreas Stolcke, Barry Chen, Horacio Franco, Venkata Ramana Rao Gadde, Martin Graciarena, Mei-Yuh Hwang, Katrin Kirchhoff, Arindam Mandal, Nelson Morgan, Xin Lin, Tim Ng, Mari Ostendorf, Kemal Sönmez, Anand Venkataraman, Dimitra Vergyri, Wen Wang, Jing Zheng, and Qifeng Zhu. 2006. Recent innovations in speech-to-text transcription at SRI-ICSI-UW. *IEEE Transactions on Audio, Speech and Language Processing*, 14(5):1729–1744. Special Issue on Progress in Rich Transcription.

Gyorgy Szaszak and Klara Vicsi. 2007. Speech recognition supported by prosodic information for fixed stress languages. *Proc. of TSD Conference*, pages 262–269.

Dimitra Vergyri, Andreas Stolcke, Venkata R. R. Gadde, Luciana Ferrer, and Elizabeth Shriberg. 2003. Prosodic knowledge sources for automatic speech recognition. *Proc. of ICASSP*, pages 208–211.

Colin W. Wightman and Mari Ostendorf. 1994. Automatic labeling of prosodic patterns. *IEEE Transaction on Speech and Auido Processing*, 2(4):469–481.

Jing Zheng, Ozgur Cetin, Mei-Yuh Hwang, Xin Lei, Andreas Stolcke, and Nelson Morgan. 2007. Combining discriminative feature, transform, and model training for large vocabulary speech recognition. *Proc. of ICASSP*, pages 633–636.

Lexically-Triggered Hidden Markov Models for Clinical Document Coding

Svetlana Kiritchenko **Colin Cherry**
Institute for Information Technology
National Research Council Canada
{Svetlana.Kiritchenko,Colin.Cherry}@nrc-cnrc.gc.ca

Abstract

The automatic coding of clinical documents is an important task for today's healthcare providers. Though it can be viewed as multi-label document classification, the coding problem has the interesting property that most code assignments can be supported by a single phrase found in the input document. We propose a Lexically-Triggered Hidden Markov Model (LT-HMM) that leverages these phrases to improve coding accuracy. The LT-HMM works in two stages: first, a lexical match is performed against a term dictionary to collect a set of candidate codes for a document. Next, a discriminative HMM selects the best subset of codes to assign to the document by tagging candidates as present or absent. By confirming codes proposed by a dictionary, the LT-HMM can share features across codes, enabling strong performance even on rare codes. In fact, we are able to recover codes that do not occur in the training set at all. Our approach achieves the best ever performance on the 2007 Medical NLP Challenge test set, with an F-measure of 89.84.

1 Introduction

The clinical domain presents a number of interesting challenges for natural language processing. Conventionally, most clinical documentation, such as doctor's notes, discharge summaries and referrals, are written in a free-text form. This narrative form is flexible, allowing healthcare professionals to express any kind of concept or event, but it is not particularly suited for large-scale analysis, search, or decision support. Converting clinical narratives into a structured form would support essential activities such as administrative reporting, quality control, biosurveillance and biomedical research (Meystre et al., 2008). One way of representing a document is to code the patient's conditions and the performed procedures into a nomenclature of clinical codes. The International Classification of Diseases, 9th and 10th revisions, Clinical Modification (ICD-9-CM, ICD-10-CM) are the official administrative coding schemes for healthcare organizations in several countries, including the US and Canada. Typically, coding is performed by trained coding professionals, but this process can be both costly and error-prone. Automated methods can speed-up the coding process, improve the accuracy and consistency of internal documentation, and even result in higher reimbursement for the healthcare organization (Benson, 2006).

Traditionally, statistical document coding is viewed as multi-class multi-label document classification, where each clinical free-text document is labelled with one or several codes from a pre-defined, possibly very large set of codes (Patrick et al., 2007; Suominen et al., 2008). One classification model is learned for each code, and then all models are applied in turn to a new document to determine which codes should be assigned to the document. The drawback of this approach is poor predictive performance on low-frequency codes, which are ubiquitous in the clinical domain.

This paper presents a novel approach to document coding that simultaneously models code-specific as well as general patterns in the data. This allows

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 742–751,
Portland, Oregon, June 19-24, 2011.

us to predict any code label, even codes for which no training data is available. Our approach, the lexically-triggered HMM (LT-HMM), is based on the fact that a code assignment is often indicated by short lexical triggers in the text. Consequently, a two-stage coding method is proposed. First, the LT-HMM identifies candidate codes by matching terms from a medical terminology dictionary. Then, it confirms or rejects each of the candidates by applying a discriminative sequence model. In this architecture, low-frequency codes can still be matched and confirmed using general characteristics of their trigger's local context, leading to better prediction performance on these codes.

2 Document Coding and Lexical Triggers

Document coding is a special case of multi-class multi-label text classification. Given a fixed set of possible codes, the ultimate goal is to assign a set of codes to documents, based on their content. Furthermore, we observe that for each code assigned to a document, there is generally at least one corresponding *trigger term* in the text that accounts for the code's assignment. For example, if an ICD-9-CM coding professional were to see "allergic bronchitis" somewhere in a clinical narrative, he or she would immediately consider adding code *493.9 (Asthma, unspecified)* to the document's code set. The presence of these trigger terms separates document coding from text classification tasks, such as topic or genre classification, where evidence for a particular label is built up throughout a document. However, this does not make document coding a term recognition task, concerned only with the detection of triggers. Codes are assigned to a document as a whole, and code assignment decisions within a document may interact. It is an interesting combination of sentence and document-level processing.

Formally, we define the document coding task as follows: given a set of documents X and a set of available codes C, assign to each document x_i a subset of codes $C_i \subset C$. We also assume access to a (noisy) mechanism to detect candidate triggers in a document. In particular, we will assume that an (incomplete) dictionary $D(c)$ exists for each code $c \in C$, which lists specific *code terms* associated with c.[1] To continue our running example: $D(\textit{493.9})$ would include the term "allergic bronchitis". Each code can have several corresponding terms while each term indicates the presence of exactly one code. A *candidate code* c is proposed each time a term from $D(c)$ is found in a document.

2.1 From triggers to codes

The presence of a term from $D(c)$ does not automatically imply the assignment of code c to a document. Even with extremely precise dictionaries, there are three main reasons why a candidate code may not appear in a document's code subset.

1. The context of the trigger term might indicate the irrelevancy of the code. In the clinical domain, such irrelevancy can be specified by a negative or speculative statement (e.g., "evaluate for pneumonia") or a family-related context (e.g., "family history of diabetes"). Only definite diagnosis of the patient should be coded.

2. There can be several closely related candidate codes; yet only one, the best fitted code should be assigned to the document. For example, the triggers "left-sided flank pain" (code *789.09*) and "abdominal pain" (code *789.00*) may both appear in the same clinical report, but only the most specific code, *789.09*, should end up in the document code set.

3. The domain can have code dependency rules. For example, the ICD-9-CM coding rules state that no symptom codes should be given to a document if a definite diagnosis is present. That is, if a document is coded with pneumonia, it should not be coded with a fever or cough. On the other hand, if the diagnosis is uncertain, then codes for the symptoms should be assigned.

This suggests a paradigm where a candidate code, suggested by a detected trigger term, is assessed in terms of both its local context (item 1) and the presence of other candidate codes for the document (items 2 and 3).

[1] Note that dictionary-based trigger detection could be replaced by tagging approaches similar to those used in named-entity-recognition or information extraction.

2.2 ICD-9-CM Coding

As a specific application we have chosen the task of assigning ICD-9-CM codes to free-form clinical narratives. We use the dataset collected for the 2007 Medical NLP Challenge organized by the Computational Medicine Center in Cincinnati, Ohio, hereafter refereed to as "CMC Challenge" (Pestian et al., 2007). For this challenge, 1954 radiology reports on outpatient chest x-ray and renal procedures were collected, disambiguated, and anonymized. The reports were annotated with ICD-9-CM codes by three coding companies, and the majority codes were selected as a gold standard. In total, 45 distinct codes were used.

For this task, our use of a dictionary to detect lexical triggers is quite reasonable. The medical domain is rich with manually-created and carefully-maintained knowledge resources. In particular, the ICD-9-CM coding guidelines come with an index file that contains hundreds of thousands of terms mapped to corresponding codes. Another valuable resource is Metathesaurus from the Unified Medical Language System (UMLS) (Lindberg et al., 1993). It has millions of terms related to medical problems, procedures, treatments, organizations, etc. Often, hospitals, clinics, and other healthcare organizations maintain their own vocabularies to introduce consistency in their internal and external documentation and to support reporting, reviewing, and meta-analysis.

This task has some very challenging properties. As mentioned above, the ICD-9-CM coding rules create strong code dependencies: codes are assigned to a document as a set and not individually. Furthermore, the code distribution throughout the CMC training documents has a very heavy tail; that is, there are a few heavily-used codes and a large number of codes that are used only occasionally. An ideal approach will work well with both high-frequency and low-frequency codes.

3 Related work

Automated clinical coding has received much attention in the medical informatics literature. Stanfill et al. reviewed 113 studies on automated coding published in the last 40 years (Stanfill et al., 2010). The authors conclude that there exists a variety of tools covering different purposes, healthcare specialties, and clinical document types; however, these tools are not generalizable and neither are their evaluation results. One major obstacle that hinders the progress in this domain is data privacy issues. To overcome this obstacle, the CMC Challenge was organized in 2007. The purpose of the challenge was to provide a common realistic dataset to stimulate the research in the area and to assess the current level of performance on the task. Forty-four teams participated in the challenge. The top-performing system achieved micro-averaged F1-score of 0.8908, and the mean score was 0.7670.

Several teams, including the winner, built pure symbolic (i.e., hand-crafted rule-based) systems (e.g., (Goldstein et al., 2007)). This approach is feasible for the small code set used in the challenge, but it is questionable in real-life settings where thousands of codes need to be considered. Later, the winning team showed how their hand-crafted rules can be built in a semi-automatic way: the initial set of rules adopted from the official coding guidelines were automatically extended with additional synonyms and code dependency rules generated from the training data (Farkas and Szarvas, 2008).

Statistical systems trained on only text-derived features (such as n-grams) did not show good performance due to a wide variety of medical language and a relatively small training set (Goldstein et al., 2007). This led to the creation of hybrid systems: symbolic and statistical classifiers used together in an ensemble or cascade (Aronson et al., 2007; Crammer et al., 2007) or a symbolic component providing features for a statistical component (Patrick et al., 2007; Suominen et al., 2008). Strong competition systems had good answers for dealing with negative and speculative contexts, taking advantage of the competition's limited set of possible code combinations, and handling of low-frequency codes.

Our proposed approach is a combination system as well. We combine a symbolic component that matches lexical strings of a document against a medical dictionary to determine possible codes (Lussier et al., 2000; Kevers and Medori, 2010) and a statistical component that finalizes the assignment of codes to the document. Our statistical component is similar to that of Crammer et al. (2007), in that we train a single model for all codes with code-

specific and generic features. However, Crammer et al. (2007) did not employ our lexical trigger step or our sequence-modeling formulation. In fact, they considered all possible code subsets, which can be infeasible in real-life settings.

4 Method

To address the task of document coding, our lexically-triggered HMM operates using a two-stage procedure:

1. Lexically match text to the dictionary to get a set of candidate codes;
2. Using features derived from the candidates and the document, select the best code subset.

In the first stage, dictionary terms are detected in the document using exact string matching. All codes corresponding to matches become candidate codes, and no other codes can be proposed for this document.

In the second stage, a single classifier is trained to select the best code subset from the matched candidates. By training a single classifier, we use all of the training data to assign binary labels (present or absent) to candidates. This is the key distinction of our method from the traditional statistical approach where a separate classifier is trained for each code. The LT-HMM allows features learned from a document coded with c_i to transfer at test time to predict code c_j, provided their respective triggers appear in similar contexts. Training one common classifier improves our chances to reliably predict codes that have few training instances, and even codes that do not appear at all in the training data.

4.1 Trigger Detection

We have manually assembled a dictionary of terms for each of the 45 codes used in the CMC challenge.[2] The dictionaries were built by collecting relevant medical terminology from UMLS, the ICD-9-CM coding guidelines, and the CMC training data. The test data was not consulted during dictionary construction. The dictionaries contain 440 terms, with 9.78 terms per code on average. Given these dictionaries, the exact-matching of terms to input documents is straightforward. In our experiments, this process finds on average 1.83 distinct candidate codes per document.

The quality of the dictionary significantly affects the prediction performance of the proposed two-stage approach. Especially important is the coverage of the dictionary. If a trigger term is missing from the dictionary and, as the result, the code is not selected as a candidate code, it will not be recovered in the following stage, resulting in a false negative. Preliminary experiments show that our dictionary recovers 94.42% of the codes in the training set and 93.20% in the test set. These numbers provide an upper bound on recall for the overall approach.

4.2 Sequence Construction

After trigger detection, we view the input document as a sequence of candidate codes, each corresponding to a detected trigger (see Figure 1). By tagging these candidates in sequence, we can label each candidate code as present or absent and use previous tagging decisions to model code interactions. The final code subset is constructed by collecting all candidate codes tagged as present.

Our training data consists of [document, code set] pairs, augmented with the trigger terms detected through dictionary matching. We transform this into a sequence to be tagged using the following steps:

Ordering: The candidate code sequence is presented in reverse chronological order, according to when their corresponding trigger terms appear in the document. That is, the last candidate to be detected by the dictionary will be the first code to appear in our candidate sequence. Reverse order was chosen because clinical documents often close with a final (and informative) diagnosis.

Merging: Each detected trigger corresponds to exactly one code; however, several triggers may be detected for the same code throughout a document. If a code has several triggers, we keep only the last occurrence. When possible, we collect relevant features (such as negation information) of all occurrences and associate them with this last occurrence.

Labelling: Each candidate code is assigned a binary label (present or absent) based on whether it appears in the gold-standard code set. Note that this

[2] Online at https://sites.google.com/site/colinacherry/ICD9CM_ACL11.txt

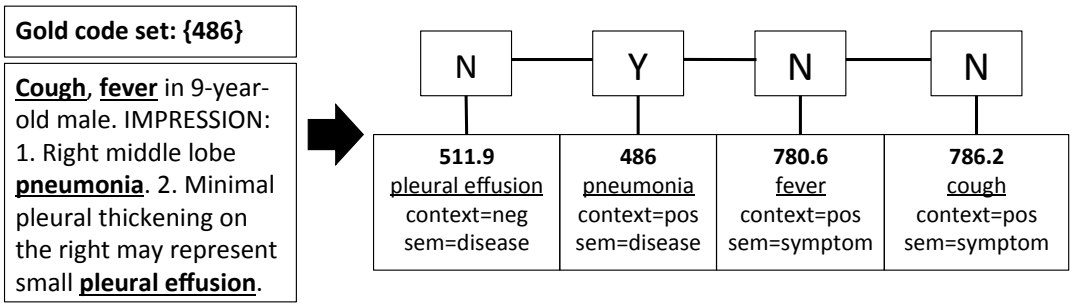


Figure 1: An example document and its corresponding gold-standard tag sequence. The top binary layer is the correct output tag sequence, which confirms or rejects the presence of candidate codes. The bottom layer shows the candidate code sequence derived from the text, with corresponding trigger phrases and some prominent features.

process can not introduce gold-standard codes that were not proposed by the dictionary.

The final output of these steps is depicted in Figure 1. To the left, we have an input text with underlined trigger phrases, as detected by our dictionary. This implies an input sequence (bottom right), which consists of detected codes and their corresponding trigger phrases. The gold-standard code set for the document is used to infer a gold-standard label sequence for these codes (top right). At test time, the goal of the classifier is to correctly predict the correct binary label sequence for new inputs. We discuss the construction of the features used to make this prediction in section 4.3.

4.3 Model

We model this sequence data using a discriminative SVM-HMM (Taskar et al., 2003; Altun et al., 2003). This allows us to use rich, over-lapping features of the input while also modeling interactions between labels. A discriminative HMM has two major categories of features: emission features, which characterize a candidate's tag in terms of the input document x, and transition features, which characterize a tag in terms of the tags that have come before it. We describe these two feature categories and then our training mechanism. All feature engineering discussed below was carried out using 10-fold cross-validation on the training set.

Transition Features

The transition features are modeled as simple indicators over n-grams of present codes, for values of n up to 10, the largest number of codes proposed by our dictionary in the training set.[3] This allows the system to learn sequences of codes that are (and are not) likely to occur in the gold-standard data.

We found it useful to pad our n-grams with "beginning of document" tokens for sequences when fewer than n codes have been labelled as present, but found it harmful to include an end-of-document tag once labelling is complete. We suspect that the small training set for the challenge makes the system prone to over-fit when modeling code-set length.

Emission Features

The vast majority of our training signal comes from emission features, which carefully model both the trigger term's local context and the document as a whole. For each candidate code, three types of features are generated: document features, ConText features, and code-semantics features (Table 1).

Document: Document features include indicators on all individual words, 2-grams, 3-grams, and 4-grams found in the document. These *n-gram features* have the candidate code appended to them, making them similar to features traditionally used in multiclass document categorization.

ConText: We take advantage of the ConText algorithm's output. ConText is publicly available software that determines the presence of negated, hypothetical, historical, and family-related context for a given phrase in a clinical text (Harkema et al., 2009).

[3] We can easily afford such a long history because input sequences are generally short and the tagging is binary, resulting in only a small number of possible histories for a document.

Features	gen.	spec.
Document		
n-gram		x
ConText		
current match		
context	x	x
only in context	x	x
more than once in context	x	x
other matches		
present	x	x
present in context = pos	x	x
code present in context	x	x
Code Semantics		
current match		
sem_type	x	
other matches		
sem_type, context = pos	x	x

Table 1: The emission features used in LT-HMM. Typeset words represent variables replaced with specific values, i.e. context $\in$ {pos,neg}, sem_type $\in$ {symptom,disease}, code is one of 45 challenge codes, n-gram is a document n-gram. Features can come in generic and/or code-specific version.

The algorithm is based on regular expression matching of the context to a precompiled list of context indicators. Regardless of its simplicity, the algorithm has shown very good performance on a variety of clinical document types. We run ConText for each trigger term located in the text and produce two types of features: features related to the candidate code in question and features related to other candidate codes of the document. Negated, hypothetical, and family-related contexts are clustered into a single *negative* context for the term. Absence of the negative context implies the *positive* context.

We used the following ConText derived indicator features: for the current candidate code, if there is at least one trigger term found in a positive (negative) context, if all trigger terms for this code are found in a positive (negative) context, if there are more than one trigger terms for the code found in a positive (negative) context; for other candidate codes of the document, if there is at least one other candidate code, if there is another candidate code with at least one trigger term found in a positive context, if there is a trigger term for candidate code c_i found in a positive (negative) context.

Code Semantics: We include features that indicate if the code itself corresponds to a disease or a symptom. This assignment was determined based on the UMLS semantic type of the code. Like the ConText features, code features come in two types: those regarding the candidate code in question and those regarding other candidate codes from the same document.

Generic versus Specific: Most of our features come in two versions: generic and code-specific. Generic features are concerned with classifying any candidate as present or absent based on characteristics of its trigger or semantics. Code-specific features append the candidate code to the feature. For example, the feature context=pos represents that the current candidate has a trigger term in a positive context, while context=pos:486 adds the information that the code in question is 486. Note that n-grams features are only code-specific, as they are not connected to any specific trigger term.

To an extent, code-specific features allow us to replicate the traditional classification approach, which focuses on one code at a time. Using these features, the classifier is free to build complex submodels for a particular code, provided that this code has enough training examples. Generic versions of the features, on the other hand, make it possible to learn common rules applicable to all codes, including low-frequency ones. In this way, even in the extreme case of having zero training examples for a particular code, the model can still potentially assign the code to new documents, provided it is detected by our dictionary. This is impossible in a traditional document-classification setting.

Training

We train our SVM-HMM with the objective of separating the correct tag sequence from all others by a fixed margin of 1, using a primal stochastic gradient optimization algorithm that follows Shalev-Shwartz et al. (2007). Let S be a set of training points (x, y), where x is the input and y is the corresponding gold-standard tag sequence. Let $\phi(x, y)$ be a function that transforms complete input-output pairs into feature vectors. We also use $\phi(x, y', y)$ as shorthand for the difference in features between

begin
Input: S, λ, n
Initialize: Set w_0 to the 0 vector
for $t = 1, 2 \ldots, n|S|$
Choose $(x, y) \in S$ at random
Set the learning rate: $\eta_t = \frac{1}{\lambda t}$
Search:
$y' = \arg\max_{y''} \left[\delta(y, y'') + w_t \cdot \phi(x, y'')\right]$
Update:
$w_{t+1} = w_t + \eta_t\big(\phi(x, y, y') - \lambda w_t\big)$
Adjust:
$w_{t+1} = w_{t+1} \cdot \min\left[1, \frac{1/\sqrt{\lambda}}{\|w_{t+1}\|}\right]$
end
Output: $w_{n|S|+1}$
end

Figure 2: Training an SVM-HMM

two outputs: $\phi(x, y', y) = \phi(x, y') - \phi(x, y)$. With this notation in place, the SVM-HMM minimizes the regularized hinge-loss:

$$\min_w \frac{\lambda}{2} w^2 + \frac{1}{|S|} \sum_{(x,y) \in S} \ell(w;\ (x, y)) \qquad (1)$$

where

$$\ell(w;\ (x, y)) = \max_{y'} \left[\delta(y, y') + w \cdot \phi(x, y', y)\right] \qquad (2)$$

and where $\delta(y, y') = 0$ when $y = y'$ and 1 otherwise.[4] Intuitively, the objective attempts to find a small weight vector w that separates all incorrect tag sequences y' from the correct tag sequence y by a margin of 1. λ controls the trade-off between regularization and training hinge-loss.

The stochastic gradient descent algorithm used to optimize this objective is shown in Figure 2. It bears many similarities to perceptron HMM training (Collins, 2002), with theoretically-motivated alterations, such as selecting training points at random[5] and the explicit inclusion of a learning rate η and a regularization term λ. The search step can be carried out with a two-best version of the Viterbi algorithm; if the one-best answer y'_1 matches the gold-standard y, that is $\delta(y, y'_1) = 0$, then y'_2 is checked to see if its loss is higher.

We tune two hyper-parameters using 10-fold cross-validation: the regularization parameter λ and a number of passes n through the training data. Using F1 as measured by 10-fold cross-validation on the training set, we found values of $\lambda = 0.1$ with $n = 5$ to prove optimal. Training time is less than one minute on modern hardware.

[4] We did not experiment with structured versions of δ that account for the number of incorrect tags in the label sequence y', as a fixed margin was already working very well. We intend to explore structured costs in future work.

[5] Like many implementations, we make n passes through S, shuffling S before each pass, rather than sampling from S with replacement $n|S|$ times.

	training	**test**
# of documents	978	976
# of distinct codes	45	45
# of distinct code subsets	94	94
# of codes with < 10 ex.	24	24
avg # of codes per document	1.25	1.23

Table 2: The training and test set characteristics.

5 Experiments

5.1 Data

For testing purposes, we use the CMC Challenge dataset. The data consists of 978 training and 976 test medical records labelled with one or more ICD-9-CM codes from a set of 45 codes. The data statistics are presented in Table 2. The training and test sets have similar, very imbalanced distributions of codes. In particular, all codes in the test set have at least one training example. Moreover, for any code subset assigned to a test document there is at least one training document labelled with the same code subset. Notably, more than half of the codes have less than 10 instances in both training and test sets. Following the challenge's protocol, we use micro-averaged F1-measure for evaluation.

5.2 Baseline

As the first baseline for comparison, we built a one-classifier-per-code statistical system. A document's code subset is implied by the set of classifiers that assign it a positive label. The classifiers use a feature set designed to mimic our LT-HMM as closely as possible, including n-grams, dictionary matches, ConText output, and symptom/disease se-

mantic types. Each classifier is trained as an SVM with a linear kernel.

Unlike our approach, this baseline cannot share features across codes, and it does not allow coding decisions for a document to inform one another. It also cannot propose codes that have not been seen in the training data as it has no model for these codes. However, one should note that it is a very strong baseline. Like our proposed system, it is built with many features derived from dictionary matches and their contexts, and thus it shares many of our system's strengths. In fact, this baseline system outperforms all published statistical approaches tested on the CMC data.

Our second baseline is a symbolic system, designed to evaluate the quality of our rule-based components when used alone. It is based on the same hand-crafted dictionary, filtered according to the ConText algorithm and four code dependency rules from (Farkas and Szarvas, 2008). These rules address the problem of overcoding: some symptom codes should be omitted when a specific disease code is present.[6]

This symbolic system has access to the same hand-crafted resources as our LT-HMM and, therefore, has a good chance of predicting low-frequency and unseen codes. However, it lacks the flexibility of our statistical solution to accept or reject code candidates based on the whole document text, which prevents it from compensating for dictionary or ConText errors. Similarly, the structure of the code dependency rules may not provide the same flexibility as our features that look at other detected triggers and previous code assignments.

5.3 Coding Accuracy

We evaluate the proposed approach on both the training set (using 10-fold cross-validation) and the test set (Table 3). The experiments demonstrate the superiority of the proposed LT-HMM approach over the one-per-code statistical scheme as well as our symbolic baseline. Furthermore, the new approach shows the best results ever achieved on the dataset, beating the top-performing system in the challenge, a symbolic method.

[6]Note that we do not match the performance of the Farkas and Szarvas system, likely due to our use of a different (and simpler) dictionary.

	Cross-fold	**Test**
Symbolic baseline	N/A	85.96
Statistical baseline	87.39	88.26
LT-HMM	89.39	89.84
CMC Best	N/A	89.08

Table 3: Micro-averaged F1-scores for statistical and symbolic baselines, the proposed LT-HMM approach, and the best CMC hand-crafted rule-based system.

System	Prec.	Rec.	F1
Full	90.91	88.80	89.84
-ConText	88.54	85.89	87.19
-Document	89.89	88.55	89.21
-Code Semantics	90.10	88.38	89.23
-Append code-specific	88.96	88.30	88.63
-Transition	90.79	88.38	89.57
-ConText & Transition	86.91	85.39	86.14

Table 4: Results on the CMC test data with each major component removed.

5.4 Ablation

Our system employs a number of emission feature templates. We measure the impact of each by removing the template, re-training, and testing on the challenge test data, as shown in Table 4. By far the most important component of our system is the output of the ConText algorithm.

We also tested a version of the system that does not create a parallel code-specific feature set by appending the candidate code to emission features. This system tags code-candidates without any code-specific components, but it still does very well, outperforming the baselines.

Removing the sequence-based transition features from our system has only a small impact on accuracy. This is because several of our emission features look at features of other candidate codes. This provides a strong approximation to the actual tagging decisions for these candidates. If we remove the ConText features, the HMM's transition features become more important (compare line 2 of Table 4 to line 7).

5.5 Low-frequency codes

As one can see from Table 2, more than half of the available codes appear fewer than 10 times in the

System	Prec.	Rec.	F1
Symbolic baseline	42.53	56.06	48.37
Statistical baseline	73.33	33.33	45.83
LT-HMM	70.00	53.03	60.34

Table 5: Results on the CMC test set, looking only at the codes with fewer than 10 examples in the training set.

System	Prec.	Rec.	F1
Symbolic baseline	60.00	80.00	68.57
All training data	72.92	74.47	73.68
One code held out	79.31	48.94	60.53

Table 6: Results on the CMC test set when all instances of a low-frequency code are held-out during training.

training documents. This does not provide much training data for a one-classifier-per-code approach, which has been a major motivating factor in the design of our LT-HMM. In Table 5, we compare our system to the baselines on the CMC test set, considering only these low-frequency codes. We show a 15-point gain in F1 over the statistical baseline on these hard cases, brought on by an substantial increase in recall. Similarly, we improve over the symbolic baseline, due to a much higher precision. In this way, the LT-HMM captures the strengths of both approaches.

Our system also has the ability to predict codes that have not been seen during training, by labelling a dictionary match for a code as present according to its local context. We simulate this setting by dropping training data. For each low-frequency code c, we hold out all training documents that include c in their gold-standard code set. We then train our system on the reduced training set and measure its ability to detect c on the unseen test data. 11 of the 24 low-frequency codes have no dictionary matches in our test data; we omit them from our analysis as we are unable to predict them. The micro-averaged results for the remaining 13 low-frequency codes are shown in Table 6, with the results from the symbolic baseline and from our system trained on the complete training data provided for comparison.

We were able to recover 49% of the test-time occurrences of codes withheld from training, while maintaining our full system's precision. Considering that traditional statistical strategies would lead to recall dropping uniformly to 0, this is a vast improvement. However, the symbolic baseline recalls 80% of occurrences in aggregate, indicating that we are not yet making optimal use of the dictionary for cases when a code is missing from the training data. By holding out only *correct* occurrences of a code c, our system becomes biased against it: all trigger terms for c that are found in the training data *must* be labelled absent. Nonetheless, out of the 13 codes with dictionary matches, there were 9 codes that we were able to recall at a rate of 50% or more, and 5 codes that achieved 100% recall.

6 Conclusion

We have presented the lexically-triggered HMM, a novel and effective approach for clinical document coding. The LT-HMM takes advantage of lexical triggers for clinical codes by operating in two stages: first, a lexical match is performed against a trigger term dictionary to collect a set of candidates codes for a document; next, a discriminative HMM selects the best subset of codes to assign to the document. Using both generic and code-specific features, the LT-HMM outperforms a traditional one-per-code statistical classification method, with substantial improvements on low-frequency codes. Also, it achieves the best ever performance on a common testbed, beating the top-performer of the 2007 CMC Challenge, a hand-crafted rule-based system. Finally, we have demonstrated that the LT-HMM can correctly predict codes never seen in the training set, a vital characteristic missing from previous statistical methods.

In the future, we would like to augment our dictionary-based matching component with entity-recognition technology. It would be interesting to model triggers as latent variables in the document coding process, in a manner similar to how latent subjective sentences have been used in document-level sentiment analysis (Yessenalina et al., 2010). This would allow us to employ a learned matching component that is trained to compliment our classification component.

Acknowledgements

Many thanks to Berry de Bruijn, Joel Martin, and the ACL-HLT reviewers for their helpful comments.

References

Y. Altun, I. Tsochantaridis, and T. Hofmann. 2003. Hidden Markov support vector machines. In *ICML*.

A. R. Aronson, O. Bodenreider, D. Demner-Fushman, K. W. Fung, V. K. Lee, J. G. Mork, A. Nvol, L. Peters, and W. J. Rogers. 2007. From indexing the biomedical literature to coding clinical text: Experience with MTI and machine learning approaches. In *BioNLP*, pages 105–112.

S. Benson. 2006. Computer assisted coding software improves documentation, coding, compliance and revenue. *Perspectives in Health Information Management*, CAC Proceedings, Fall.

M. Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *EMNLP*.

K. Crammer, M. Dredze, K. Ganchev, P. P. Talukdar, and S. Carroll. 2007. Automatic code assignment to medical text. In *BioNLP*, pages 129–136.

R. Farkas and G. Szarvas. 2008. Automatic construction of rule-based ICD-9-CM coding systems. *BMC Bioinformatics*, 9(Suppl 3):S10.

I. Goldstein, A. Arzumtsyan, and Uzuner. 2007. Three approaches to automatic assignment of ICD-9-CM codes to radiology reports. In *AMIA*, pages 279–283.

H. Harkema, J. N. Dowling, T. Thornblade, and W. W. Chapman. 2009. Context: An algorithm for determining negation, experiencer, and temporal status from clinical reports. *Journal of Biomedical Informatics*, 42(5):839–851, October.

L. Kevers and J. Medori. 2010. Symbolic classification methods for patient discharge summaries encoding into ICD. In *Proceedings of the 7th International Conference on NLP (IceTAL)*, pages 197–208, Reykjavik, Iceland, August.

D. A. Lindberg, B. L. Humphreys, and A. T. McCray. 1993. The Unified Medical Language System. *Methods of Information in Medicine*, 32(4):281–291.

Y. A. Lussier, L. Shagina, and C. Friedman. 2000. Automating ICD-9-CM encoding using medical language processing: A feasibility study. In *AMIA*, page 1072.

S. M. Meystre, G. K. Savova, K. C. Kipper-Schuler, and J. F. Hurdle. 2008. Extracting information from textual documents in the electronic health record: a review of recent research. *Methods of Information in Medicine*, 47(Suppl 1):128–144.

J. Patrick, Y. Zhang, and Y. Wang. 2007. Developing feature types for classifying clinical notes. In *BioNLP*, pages 191–192.

J. P. Pestian, C. Brew, P. Matykiewicz, D. J. Hovermale, N. Johnson, K. B. Cohen, and W. Duch. 2007. A shared task involving multi-label classification of clinical free text. In *BioNLP*, pages 97–104.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. 2007. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *ICML*, Corvallis, OR.

M. H. Stanfill, M. Williams, S. H. Fenton, R. A. Jenders, and W. R. Hersh. 2010. A systematic literature review of automated clinical coding and classification systems. *JAMIA*, 17:646–651.

H. Suominen, F. Ginter, S. Pyysalo, A. Airola, T. Pahikkala, S. Salanter, and T. Salakoski. 2008. Machine learning to automate the assignment of diagnosis codes to free-text radiology reports: a method description. In *Proceedings of the ICML Workshop on Machine Learning for Health-Care Applications*, Helsinki, Finland.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin markov networks. In *Neural Information Processing Systems Conference (NIPS03)*, Vancouver, Canada, December.

A. Yessenalina, Y. Yue, and C. Cardie. 2010. Multi-level structured models for document-level sentiment classification. In *EMNLP*.

Learning to Grade Short Answer Questions using Semantic Similarity Measures and Dependency Graph Alignments

Michael Mohler
Dept. of Computer Science
University of North Texas
Denton, TX
mgm0038@unt.edu

Razvan Bunescu
School of EECS
Ohio University
Athens, Ohio
bunescu@ohio.edu

Rada Mihalcea
Dept. of Computer Science
University of North Texas
Denton, TX
rada@cs.unt.edu

Abstract

In this work we address the task of computer-assisted assessment of short student answers. We combine several graph alignment features with lexical semantic similarity measures using machine learning techniques and show that the student answers can be more accurately graded than if the semantic measures were used in isolation. We also present a first attempt to align the dependency graphs of the student and the instructor answers in order to make use of a structural component in the automatic grading of student answers.

1 Introduction

One of the most important aspects of the learning process is the assessment of the knowledge acquired by the learner. In a typical classroom assessment (e.g., an exam, assignment or quiz), an instructor or a grader provides students with feedback on their answers to questions related to the subject matter. However, in certain scenarios, such as a number of sites worldwide with limited teacher availability, online learning environments, and individual or group study sessions done outside of class, an instructor may not be readily available. In these instances, students still need some assessment of their knowledge of the subject, and so, we must turn to computer-assisted assessment (CAA).

While some forms of CAA do not require sophisticated text understanding (e.g., multiple choice or true/false questions can be easily graded by a system if the correct solution is available), there are also student answers made up of free text that may require textual analysis. Research to date has concentrated on two subtasks of CAA: grading essay responses, which includes checking the style, grammaticality, and coherence of the essay (Higgins et al., 2004), and the assessment of short student answers (Leacock and Chodorow, 2003; Pulman and Sukkarieh, 2005; Mohler and Mihalcea, 2009), which is the focus of this work.

An automatic short answer grading system is one that automatically assigns a grade to an answer provided by a student, usually by comparing it to one or more correct answers. Note that this is different from the related tasks of paraphrase detection and textual entailment, since a common requirement in student answer grading is to provide a grade on a certain scale rather than make a simple yes/no decision.

In this paper, we explore the possibility of improving upon existing bag-of-words (BOW) approaches to short answer grading by utilizing machine learning techniques. Furthermore, in an attempt to mirror the ability of humans to understand structural (e.g. syntactic) differences between sentences, we employ a rudimentary dependency-graph alignment module, similar to those more commonly used in the textual entailment community.

Specifically, we seek answers to the following questions. **First**, to what extent can machine learning be leveraged to improve upon existing approaches to short answer grading. **Second**, does the dependency parse structure of a text provide clues that can be exploited to improve upon existing BOW methodologies?

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 752–762,
Portland, Oregon, June 19-24, 2011.

2 Related Work

Several state-of-the-art short answer grading systems (Sukkarieh et al., 2004; Mitchell et al., 2002) require manually crafted patterns which, if matched, indicate that a question has been answered correctly. If an annotated corpus is available, these patterns can be supplemented by learning additional patterns semi-automatically. The Oxford-UCLES system (Sukkarieh et al., 2004) bootstraps patterns by starting with a set of keywords and synonyms and searching through windows of a text for new patterns. A later implementation of the Oxford-UCLES system (Pulman and Sukkarieh, 2005) compares several machine learning techniques, including inductive logic programming, decision tree learning, and Bayesian learning, to the earlier pattern matching approach, with encouraging results.

C-Rater (Leacock and Chodorow, 2003) matches the syntactical features of a student response (i.e., subject, object, and verb) to that of a set of correct responses. This method specifically disregards the BOW approach to take into account the difference between "dog bites man" and "man bites dog" while still trying to detect changes in voice (i.e., "the man was bitten by the dog").

Another short answer grading system, AutoTutor (Wiemer-Hastings et al., 1999), has been designed as an immersive tutoring environment with a graphical "talking head" and speech recognition to improve the overall experience for students. AutoTutor eschews the pattern-based approach entirely in favor of a BOW LSA approach (Landauer and Dumais, 1997). Later work on AutoTutor(Wiemer-Hastings et al., 2005; Malatesta et al., 2002) seeks to expand upon their BOW approach which becomes less useful as causality (and thus word order) becomes more important.

A text similarity approach was taken in (Mohler and Mihalcea, 2009), where a grade is assigned based on a measure of relatedness between the student and the instructor answer. Several measures are compared, including knowledge-based and corpus-based measures, with the best results being obtained with a corpus-based measure using Wikipedia combined with a "relevance feedback" approach that iteratively augments the instructor answer by integrating the student answers that receive the highest grades.

In the dependency-based classification component of the Intelligent Tutoring System (Nielsen et al., 2009), instructor answers are parsed, enhanced, and manually converted into a set of content-bearing dependency triples or facets. For each facet of the instructor answer each student's answer is labelled to indicate whether it has addressed that facet and whether or not the answer was contradictory. The system uses a decision tree trained on part-of-speech tags, dependency types, word count, and other features to attempt to learn how best to classify an answer/facet pair.

Closely related to the task of short answer grading is the task of textual entailment (Dagan et al., 2005), which targets the identification of a directional inferential relation between texts. Given a pair of two texts as input, typically referred to as *text* and *hypothesis*, a textual entailment system automatically finds if the hypothesis is entailed by the text.

In particular, the entailment-related works that are most similar to our own are the graph matching techniques proposed by Haghighi et al. (2005) and Rus et al. (2007). Both input texts are converted into a graph by using the dependency relations obtained from a parser. Next, a matching score is calculated, by combining separate vertex- and edge-matching scores. The vertex matching functions use word-level lexical and semantic features to determine the quality of the match while the the edge matching functions take into account the types of relations and the difference in lengths between the aligned paths.

Following the same line of work in the textual entailment world are (Raina et al., 2005), (MacCartney et al., 2006), (de Marneffe et al., 2007), and (Chambers et al., 2007), which experiment variously with using diverse knowledge sources, using a perceptron to learn alignment decisions, and exploiting natural logic.

3 Answer Grading System

We use a set of syntax-aware graph alignment features in a three-stage pipelined approach to short answer grading, as outlined in Figure 1.

In the first stage (Section 3.1), the system is provided with the dependency graphs for each pair of instructor (A_i) and student (A_s) answers. For each

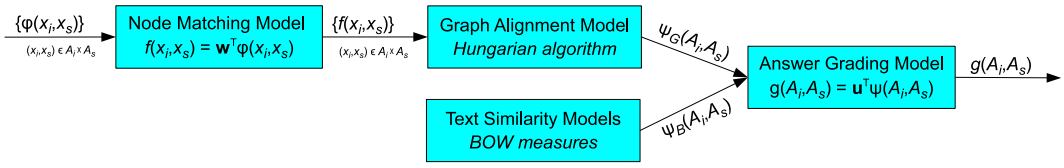


Figure 1: Pipeline model for scoring short-answer pairs.

node in the instructor's dependency graph, we compute a similarity score for each node in the student's dependency graph based upon a set of lexical, semantic, and syntactic features applied to both the pair of nodes and their corresponding subgraphs. The scoring function is trained on a small set of manually aligned graphs using the averaged perceptron algorithm.

In the second stage (Section 3.2), the node similarity scores calculated in the previous stage are used to weight the edges in a bipartite graph representing the nodes in A_i on one side and the nodes in A_s on the other. We then apply the Hungarian algorithm to find both an optimal matching and the score associated with such a matching. In this stage, we also introduce question demoting in an attempt to reduce the advantage of parroting back words provided in the question.

In the final stage (Section 3.4), we produce an overall grade based upon the alignment scores found in the previous stage as well as the results of several semantic BOW similarity measures (Section 3.3). Using each of these as features, we use Support Vector Machines (SVM) to produce a combined real-number grade. Finally, we build an Isotonic Regression (IR) model to transform our output scores onto the original [0,5] scale for ease of comparison.

3.1 Node to Node Matching

Dependency graphs for both the student and instructor answers are generated using the Stanford Dependency Parser (de Marneffe et al., 2006) in collapse/propagate mode. The graphs are further post-processed to propagate dependencies across the "APPOS" (apposition) relation, to explicitly encode negation, part-of-speech, and sentence ID within each node, and to add an overarching ROOT node governing the main verb or predicate of each sentence of an answer. The final representation is a list of (relation,governor,dependent) triples, where governor and dependent are both tokens described by the tuple (sentenceID:token:POS:wordPosition). For example: **(nsubj, 1:provide:VBZ:4, 1:program:NN:3)** indicates that the noun "program" is a subject in sentence 1 whose associated verb is "provide."

If we consider the dependency graphs output by the Stanford parser as directed (minimally cyclic) graphs,[1] we can define for each node x a set of nodes N_x that are reachable from x using a subset of the relations (i.e., edge types)[2]. We variously define "reachable" in four ways to create four subgraphs defined for each node. These are as follows:

- N_x^0 : All edge types may be followed
- N_x^1 : All edge types except for subject types, ADVCL, PURPCL, APPOS, PARATAXIS, ABBREV, TMOD, and CONJ
- N_x^2 : All edge types except for those in N_x^1 plus object/complement types, PREP, and RCMOD
- N_x^3 : No edge types may be followed (This set is the single starting node x)

Subgraph similarity (as opposed to simple node similarity) is a means to escape the rigidity involved in aligning parse trees while making use of as much of the sentence structure as possible. Humans intuitively make use of modifiers, predicates, and subordinate clauses in determining that two sentence entities are similar. For instance, the entity-describing phrase "men who put out fires" matches well with "firemen," but the words "men" and "firemen" have

[1]The standard output of the Stanford Parser produces rooted trees. However, the process of collapsing and propagating dependences violates the tree structure which results in a tree with a few cross-links between distinct branches.

[2]For more information on the relations used in this experiment, consult the Stanford Typed Dependencies Manual at http://nlp.stanford.edu/software/dependencies_manual.pdf

less inherent similarity. It remains to be determined how much of a node's subgraph will positively enrich its semantics. In addition to the complete N_x^0 subgraph, we chose to include N_x^1 and N_x^2 as tightening the scope of the subtree by first removing more abstract relations, then sightly more concrete relations.

We define a total of 68 features to be used to train our machine learning system to compute node-node (more specifically, subgraph-subgraph) matches. Of these, 36 are based upon the semantic similarity of four subgraphs defined by $N_x^{[0..3]}$. All eight WordNet-based similarity measures listed in Section 3.3 plus the LSA model are used to produce these features. The remaining 32 features are lexico-syntactic features[3] defined only for N_x^3 and are described in more detail in Table 2.

We use $\phi(x_i, x_s)$ to denote the feature vector associated with a pair of nodes $\langle x_i, x_s \rangle$, where x_i is a node from the instructor answer A_i and x_s is a node from the student answer A_s. A matching score can then be computed for any pair $\langle x_i, x_s \rangle \in A_i \times A_s$ through a linear scoring function $f(x_i, x_s) = \mathbf{w}^T \phi(x_i, x_s)$. In order to learn the parameter vector $\mathbf{w}$, we use the averaged version of the perceptron algorithm (Freund and Schapire, 1999; Collins, 2002).

As training data, we randomly select a subset of the student answers in such a way that our set was roughly balanced between good scores, mediocre scores, and poor scores. We then manually annotate each node pair $\langle x_i, x_s \rangle$ as matching, i.e. $A(x_i, x_s) = +1$, or not matching, i.e. $A(x_i, x_s) = -1$. Overall, 32 student answers in response to 21 questions with a total of 7303 node pairs (656 matches, 6647 non-matches) are manually annotated. The pseudocode for the learning algorithm is shown in Table 1. After training the perceptron, these 32 student answers are removed from the dataset, not used as training further along in the pipeline, and are not included in the final results. After training for 50 epochs,[4] the matching score $f(x_i, x_s)$ is calculated (and cached) for each node-node pair across all student answers for all assignments.

[3]Note that synonyms include negated antonyms (and vice versa). Hypernymy and hyponymy are restricted to at most two steps).

[4]This value was chosen arbitrarily and was not tuned in anyway

```
0. set w ← 0, w̄ ← 0, n ← 0
1. repeat for T epochs:
2.     foreach ⟨A_i; A_s⟩:
3.         foreach ⟨x_i, x_s⟩ ∈ A_i × A_s:
4.             if sgn(w^T φ(x_i, x_s)) ≠ sgn(A(x_i, x_s)):
5.                 set w ← w + A(x_i, x_s) φ(x_i, x_s)
6.             set w̄ ← w̄ + w, n ← n + 1
7. return w̄/n.
```

Table 1: Perceptron Training for Node Matching.

3.2 Graph to Graph Alignment

Once a score has been computed for each node-node pair across all student/instructor answer pairs, we attempt to find an optimal alignment for the answer pair. We begin with a bipartite graph where each node in the student answer is represented by a node on the left side of the bipartite graph and each node in the instructor answer is represented by a node on the right side. The score associated with each edge is the score computed for each node-node pair in the previous stage. The bipartite graph is then augmented by adding dummy nodes to both sides which are allowed to match any node with a score of zero. An optimal alignment between the two graphs is then computed efficiently using the Hungarian algorithm. Note that this results in an optimal matching, not a mapping, so that an individual node is associated with at most one node in the other answer.

At this stage we also compute several alignment-based scores by applying various transformations to the input graphs, the node matching function, and the alignment score itself.

The first and simplest transformation involves the normalization of the alignment score. While there are several possible ways to normalize a matching such that longer answers do not unjustly receive higher scores, we opted to simply divide the total alignment score by the number of nodes in the instructor answer.

The second transformation scales the node matching score by multiplying it with the idf[5] of the instructor answer node, i.e., replace $f(x_i, x_s)$ with $idf(x_i) * f(x_i, x_s)$.

The third transformation relies upon a certain real-world intuition associated with grading student

[5]Inverse document frequency, as computed from the British National Corpus (BNC)

Name	Type	# features	Description
RootMatch	binary	5	Is a ROOT node matched to: ROOT, N, V, JJ, or Other
Lexical	binary	3	Exact match, Stemmed match, close Levenshtein match
POSMatch	binary	2	Exact POS match, Coarse POS match
POSPairs	binary	8	Specific X-Y POS matches found
Ontological	binary	4	WordNet relationships: synonymy, antonymy, hypernymy, hyponymy
RoleBased	binary	3	Has as a child - subject, object, verb
VerbsSubject	binary	3	Both are verbs and neither, one, or both have a subject child
VerbsObject	binary	3	Both are verbs and neither, one, or both have an object child
Semantic	real	36	Nine semantic measures across four subgraphs each
Bias	constant	1	A value of 1 for all vectors
Total		68	

Table 2: Subtree matching features used to train the perceptron

answers – repeating words in the question is easy and is not necessarily an indication of student understanding. With this in mind, we remove any words in the question from both the instructor answer and the student answer.

In all, the application of the three transformations leads to eight different transform combinations, and therefore eight different alignment scores. For a given answer pair (A_i, A_s), we assemble the eight graph alignment scores into a feature vector $\psi_G(A_i, A_s)$.

3.3 Lexical Semantic Similarity

Haghighi et al. (2005), working on the entailment detection problem, point out that finding a good alignment is not sufficient to determine that the aligned texts are in fact entailing. For instance, two identical sentences in which an adjective from one is replaced by its antonym will have very similar structures (which indicates a good alignment). However, the sentences will have opposite meanings. Further information is necessary to arrive at an appropriate score.

In order to address this, we combine the graph alignment scores, which encode syntactic knowledge, with the scores obtained from semantic similarity measures.

Following Mihalcea et al. (2006) and Mohler and Mihalcea (2009), we use eight knowledge-based measures of semantic similarity: shortest path [PATH], Leacock & Chodorow (1998) [LCH], Lesk (1986), Wu & Palmer(1994) [WUP], Resnik (1995) [RES], Lin (1998), Jiang & Conrath (1997) [JCN], Hirst & St. Onge (1998) [HSO], and two corpus-based measures: Latent Semantic Analysis [LSA] (Landauer and Dumais, 1997) and Explicit Semantic Analysis [ESA] (Gabrilovich and Markovitch, 2007).

Briefly, for the knowledge-based measures, we use the maximum semantic similarity – for each open-class word – that can be obtained by pairing it up with individual open-class words in the second input text. We base our implementation on the WordNet::Similarity package provided by Pedersen et al. (2004). For the corpus-based measures, we create a vector for each answer by summing the vectors associated with each word in the answer – ignoring stopwords. We produce a score in the range [0..1] based upon the cosine similarity between the student and instructor answer vectors. The LSA model used in these experiments was built by training Infomap[6] on a subset of Wikipedia articles that contain one or more common computer science terms. Since ESA uses Wikipedia article associations as vector features, it was trained using a full Wikipedia dump.

3.4 Answer Ranking and Grading

We combine the alignment scores $\psi_G(A_i, A_s)$ with the scores $\psi_B(A_i, A_s)$ from the lexical semantic similarity measures into a single feature vector $\psi(A_i, A_s) = [\psi_G(A_i, A_s)|\psi_B(A_i, A_s)]$. The feature vector $\psi_G(A_i, A_s)$ contains the eight alignment scores found by applying the three transformations in the graph alignment stage. The feature vector $\psi_B(A_i, A_s)$ consists of eleven semantic features – the eight knowledge-based features plus LSA, ESA and a vector consisting only of tf*idf weights – both with and without question demoting. Thus, the entire feature vector $\psi(A_i, A_s)$ contains a total of 30 features.

[6] http://Infomap-nlp.sourceforge.net/

An input pair (A_i, A_s) is then associated with a grade $g(A_i, A_s) = \mathbf{u}^T \psi(A_i, A_s)$ computed as a linear combination of features. The weight vector $\mathbf{u}$ is trained to optimize performance in two scenarios:

Regression: An SVM model for regression (SVR) is trained using as target function the grades assigned by the instructors. We use the libSVM [7] implementation of SVR, with tuned parameters.

Ranking: An SVM model for ranking (SVMRank) is trained using as ranking pairs all pairs of student answers (A_s, A_t) such that $grade(A_i, A_s) > grade(A_i, A_t)$, where A_i is the corresponding instructor answer. We use the SVMLight [8] implementation of SVMRank with tuned parameters.

In both cases, the parameters are tuned using a grid-search. At each grid point, the training data is partitioned into 5 folds which are used to train a temporary SVM model with the given parameters. The regression passage selects the grid point with the minimal mean square error (MSE), and the SVMRank package tries to minimize the number of discordant pairs. The parameters found are then used to score the test set – a set not used in the grid training.

3.5 Isotonic Regression

Since the end result of any grading system is to give a student feedback on their answers, we need to ensure that the system's final score has some meaning. With this in mind, we use isotonic regression (Zadrozny and Elkan, 2002) to convert the system scores onto the same [0..5] scale used by the annotators. This has the added benefit of making the system output more directly related to the annotated grade, which makes it possible to report root mean square error in addition to correlation. We train the isotonic regression model on each type of system output (i.e., alignment scores, SVM output, BOW scores).

4 Data Set

To evaluate our method for short answer grading, we created a data set of questions from introductory computer science assignments with answers provided by a class of undergraduate students. The assignments were administered as part of a Data Structures course at the University of North Texas. For each assignment, the student answers were collected via an online learning environment.

The students submitted answers to 80 questions spread across ten assignments and two examinations.[9] Table 3 shows two question-answer pairs with three sample student answers each. Thirty-one students were enrolled in the class and submitted answers to these assignments. The data set we work with consists of a total of 2273 student answers. This is less than the expected $31 \times 80 = 2480$ as some students did not submit answers for a few assignments. In addition, the student answers used to train the perceptron are removed from the pipeline after the perceptron training stage.

The answers were independently graded by two human judges, using an integer scale from 0 (completely incorrect) to 5 (perfect answer). Both human judges were graduate students in the computer science department; one (grader1) was the teaching assistant assigned to the Data Structures class, while the other (grader2) is one of the authors of this paper. We treat the average grade of the two annotators as the gold standard against which we compare our system output.

Difference	Examples	% of examples
0	1294	57.7%
1	514	22.9%
2	231	10.3%
3	123	5.5%
4	70	3.1%
5	9	0.4%

Table 4: Annotator Analysis

The annotators were given no explicit instructions on how to assign grades other than the [0..5] scale. Both annotators gave the same grade 57.7% of the time and gave a grade only 1 point apart 22.9% of the time. The full breakdown can be seen in Table 4. In addition, an analysis of the grading patterns indicate that the two graders operated off of different grading policies where one grader (grader1) was more generous than the other. In fact, when the two differed, grader1 gave the higher grade 76.6% of the time. The average grade given by grader1 is 4.43,

[7] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[8] http://svmlight.joachims.org/

[9] Note that this is an expanded version of the dataset used by Mohler and Mihalcea (2009)

	Sample questions, correct answers, and student answers	Grades
Question:	What is the role of a prototype program in problem solving?	
Correct answer:	To simulate the behavior of portions of the desired software product.	
Student answer 1:	A prototype program is used in problem solving to collect data for the problem.	1, 2
Student answer 2:	It simulates the behavior of portions of the desired software product.	5, 5
Student answer 3:	To find problem and errors in a program before it is finalized.	2, 2
Question:	What are the main advantages associated with object-oriented programming?	
Correct answer:	Abstraction and reusability.	
Student answer 1:	They make it easier to reuse and adapt previously written code and they separate complex programs into smaller, easier to understand classes.	5, 4
Student answer 2:	Object oriented programming allows programmers to use an object with classes that can be changed and manipulated while not affecting the entire object at once.	1, 1
Student answer 3:	Reusable components, Extensibility, Maintainability, it reduces large problems into smaller more manageable problems.	4, 4

Table 3: A sample question with short answers provided by students and the grades assigned by the two human judges

while the average grade given by grader2 is 3.94. The dataset is biased towards correct answers. We believe all of these issues correctly mirror real-world issues associated with the task of grading.

5 Results

We independently test two components of our overall grading system: the node alignment detection scores found by training the perceptron, and the overall grades produced in the final stage. For the alignment detection, we report the precision, recall, and F-measure associated with correctly detecting matches. For the grading stage, we report a single Pearson's correlation coefficient tracking the annotator grades (average of the two annotators) and the output score of each system. In addition, we report the Root Mean Square Error (RMSE) for the full dataset as well as the median RMSE across each individual question. This is to give an indication of the performance of the system for grading a single question in isolation.[10]

5.1 Perceptron Alignment

For the purpose of this experiment, the scores associated with a given node-node matching are converted into a simple yes/no matching decision where positive scores are considered a match and negative scores a non-match. The threshold weight learned from the bias feature strongly influences the point at which real scores change from non-matches to matches, and given the threshold weight learned by the algorithm, we find an F-measure of 0.72, with precision(P) = 0.85 and recall(R) = 0.62. However, as the perceptron is designed to minimize error rate, this may not reflect an optimal objective when seeking to detect matches. By manually varying the threshold, we find a maximum F-measure of 0.76, with P=0.79 and R=0.74. Figure 2 shows the full precision-recall curve with the F-measure overlaid.

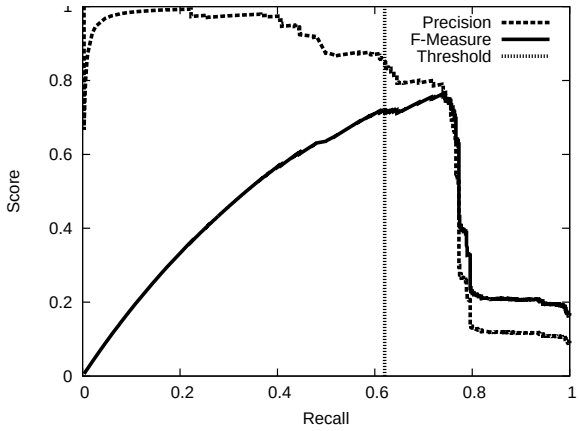


Figure 2: Precision, recall, and F-measure on node-level match detection

5.2 Question Demoting

One surprise while building this system was the consistency with which the novel technique of *question demoting* improved scores for the BOW similarity measures. With this relatively minor change the average correlation between the BOW methods' sim-

[10] We initially intended to report an aggregate of question-level Pearson correlation results, but discovered that the dataset contained one question for which each student received full points – leaving the correlation undefined. We believe that this casts some doubt on the applicability of Pearson's (or Spearman's) correlation coefficient for the short answer grading task. We have retained its use here alongside RMSE for ease of comparison.

ilarity scores and the student grades improved by up to 0.046 with an average improvement of 0.019 across all eleven semantic features. Table 5 shows the results of applying question demoting to our semantic features. When comparing scores using RMSE, the difference is less consistent, yielding an average improvement of 0.002. However, for one measure (tf*idf), the improvement is 0.063 which brings its RMSE score close to the lowest of all BOW metrics. The reasons for this are not entirely clear. As a baseline, we include here the results of assigning the average grade (as determined on the training data) for each question. The average grade was chosen as it minimizes the RMSE on the training data.

	ρ	w/ QD	RMSE	w/ QD	Med. RMSE	w/ QD
Lesk	**0.450**	**0.462**	1.034	1.050	**0.930**	**0.919**
JCN	0.443	0.461	**1.022**	1.026	0.954	0.923
HSO	0.441	0.456	1.036	1.034	0.966	0.935
PATH	0.436	0.457	1.029	1.030	0.940	**0.918**
RES	0.409	0.431	1.045	1.035	0.996	0.941
Lin	0.382	0.407	1.069	1.056	0.981	0.949
LCH	0.367	0.387	1.068	1.069	0.986	0.958
WUP	0.325	0.343	1.090	1.086	1.027	0.977
ESA	0.395	0.401	1.031	1.086	0.990	0.955
LSA	0.328	0.335	1.065	1.061	0.951	1.000
tf*idf	0.281	0.327	1.085	**1.022**	0.991	**0.918**
Avg.grade			1.097	1.097	0.973	0.973

Table 5: BOW Features with Question Demoting (QD). Pearson's correlation, root mean square error (RMSE), and median RMSE for all individual questions.

5.3 Alignment Score Grading

Before applying any machine learning techniques, we first test the quality of the eight graph alignment features $\psi_G(A_i, A_s)$ independently. Results indicate that the basic alignment score performs comparably to most BOW approaches. The introduction of *idf* weighting seems to degrade performance somewhat, while introducing question demoting causes the correlation with the grader to increase while increasing RMSE somewhat. The four normalized components of $\psi_G(A_i, A_s)$ are reported in Table 6.

5.4 SVM Score Grading

The SVM components of the system are run on the full dataset using a 12-fold cross validation. Each of the 10 assignments and 2 examinations (for a total of 12 folds) is scored independently with ten of the remaining eleven used to train the machine learning system. For each fold, one additional fold is held out for later use in the development of an isotonic regression model (see Figure 3). The parameters (for cost C and tube width ϵ) were found using a grid search. At each point on the grid, the data from the ten training folds was partitioned into 5 sets which were scored according to the current parameters. SVMRank and SVR sought to minimize the number of discordant pairs and the mean absolute error, respectively.

	Standard	w/ IDF	w/ QD	w/ QD+IDF
Pearson's ρ	0.411	0.277	**0.428**	0.291
RMSE	**1.018**	1.078	1.046	1.076
Median RMSE	**0.910**	0.970	0.919	0.992

Table 6: Alignment Feature/Grade Correlations using Pearson's ρ. Results are also reported when inverse document frequency weighting (IDF) and question demoting (QD) are used.

Both SVM models are trained using a linear kernel.[11] Results from both the SVR and the SVMRank implementations are reported in Table 7 along with a selection of other measures. Note that the RMSE score is computed after performing isotonic regression on the SVMRank results, but that it was unnecessary to perform an isotonic regression on the SVR results as the system was trained to produce a score on the correct scale.

We report the results of running the systems on three subsets of features $\psi(A_i, A_s)$: BOW features $\psi_B(A_i, A_s)$ only, alignment features $\psi_G(A_i, A_s)$ only, or the full feature vector (labeled "Hybrid"). Finally, three subsets of the alignment features are used: only unnormalized features, only normalized features, or the full alignment feature set.

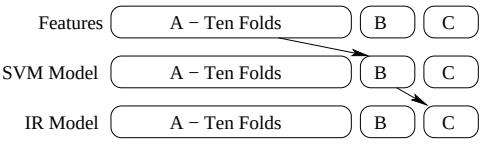


Figure 3: Dependencies of the SVM/IR training stages.

[11] We also ran the SVR system using quadratic and radial-basis function (RBF) kernels, but the results did not show significant improvement over the simpler linear kernel.

						Unnormalized		Normalized		Both	
	IAA	Avg. grade	tf*idf	Lesk	BOW	Align	Hybrid	Align	Hybrid	Align	Hybrid
					SVMRank						
Pearson's ρ	0.586		0.327	0.450	0.480	0.266	0.451	0.447	**0.518**	0.424	0.493
RMSE	0.659	1.097	1.022	1.050	1.042	1.093	1.038	1.015	**0.998**	1.029	1.021
Median RMSE	0.605	0.973	0.918	0.919	0.943	0.974	0.903	**0.865**	0.873	0.904	0.901
					SVR						
Pearson's ρ	0.586		0.327	0.450	0.431	0.167	0.437	0.433	0.459	0.434	**0.464**
RMSE	0.659	1.097	1.022	1.050	0.999	1.133	0.995	1.001	0.982	1.003	**0.978**
Median RMSE	0.605	0.973	0.918	0.919	0.910	0.987	0.893	0.894	0.877	0.886	**0.862**

Table 7: The results of the SVM models trained on the full suite of BOW measures, the alignment scores, and the hybrid model. The terms "normalized", "unnormalized", and "both" indicate which subset of the 8 alignment features were used to train the SVM model. For ease of comparison, we include in both sections the scores for the IAA, the "Average grade" baseline, and two of the top performing BOW metrics – both with question demoting.

6 Discussion and Conclusions

There are three things that we can learn from these experiments. First, we can see from the results that several systems appear better when evaluating on a correlation measure like Pearson's ρ, while others appear better when analyzing error rate. The SVM-Rank system seemed to outperform the SVR system when measuring correlation, however the SVR system clearly had a minimal RMSE. This is likely due to the different objective function in the corresponding optimization formulations: while the ranking model attempts to ensure a correct ordering between the grades, the regression model seeks to minimize an error objective that is closer to the RMSE. It is difficult to claim that either system is superior.

Likewise, perhaps the most unexpected result of this work is the differing analyses of the simple tf*idf measure – originally included only as a baseline. Evaluating with a correlative measure yields predictably poor results, but evaluating the error rate indicates that it is comparable to (or better than) the more intelligent BOW metrics. One explanation for this result is that the skewed nature of this "natural" dataset favors systems that tend towards scores in the 4 to 4.5 range. In fact, 46% of the scores output by the tf*idf measure (after IR) were within the 4 to 4.5 range and only 6% were below 3.5. Testing on a more balanced dataset, this tendency to fit to the average would be less advantageous.

Second, the supervised learning techniques are clearly able to leverage multiple BOW measures to yield improvements over individual BOW metrics. The correlation for the BOW-only SVM model for SVMRank improved upon the best BOW feature from .462 to .480. Likewise, using the BOW-only SVM model for SVR reduces the RMSE by .022 overall compared to the best BOW feature.

Third, the rudimentary alignment features we have introduced here are not sufficient to act as a standalone grading system. However, even with a very primitive attempt at alignment detection, we show that it is possible to improve upon grade learning systems that only consider BOW features. The correlations associated with the hybrid systems (esp. those using normalized alignment data) frequently show an improvement over the BOW-only SVM systems. This is true for both SVM systems when considering either evaluation metric.

Future work will concentrate on improving the quality of the answer alignments by training a model to directly output graph-to-graph alignments. This learning approach will allow the use of more complex alignment features, for example features that are defined on pairs of aligned edges or on larger subtrees in the two input graphs. Furthermore, given an alignment, we can define several phrase-level grammatical features such as negation, modality, tense, person, number, or gender, which make better use of the alignment itself.

Acknowledgments

This work was partially supported by a National Science Foundation CAREER award #0747340. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

N. Chambers, D. Cer, T. Grenager, D. Hall, C. Kiddon, B. MacCartney, M.C. de Marneffe, D. Ramage, E. Yeh, and C.D. Manning. 2007. Learning alignments and leveraging natural logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 165–170. Association for Computational Linguistics.

M. Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-02)*, Philadelphia, PA, July.

I. Dagan, O. Glickman, and B. Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Proceedings of the PASCAL Workshop*.

M.C. de Marneffe, B. MacCartney, and C.D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC 2006*.

M.C. de Marneffe, T. Grenager, B. MacCartney, D. Cer, D. Ramage, C. Kiddon, and C.D. Manning. 2007. Aligning semantic graphs for textual inference and machine reading. In *Proceedings of the AAAI Spring Symposium*. Citeseer.

Y. Freund and R. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37:277–296.

E. Gabrilovich and S. Markovitch. 2007. Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 6–12.

A.D. Haghighi, A.Y. Ng, and C.D. Manning. 2005. Robust textual inference via graph matching. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 387–394. Association for Computational Linguistics.

D. Higgins, J. Burstein, D. Marcu, and C. Gentile. 2004. Evaluating multiple aspects of coherence in student essays. In *Proceedings of the annual meeting of the North American Chapter of the Association for Computational Linguistics*, Boston, MA.

G. Hirst and D. St-Onge, 1998. *Lexical chains as representations of contexts for the detection and correction of malaproprisms*. The MIT Press.

J. Jiang and D. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics*, Taiwan.

T.K. Landauer and S.T. Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104.

C. Leacock and M. Chodorow. 1998. Combining local context and WordNet sense similarity for word sense identification. In *WordNet, An Electronic Lexical Database*. The MIT Press.

C. Leacock and M. Chodorow. 2003. C-rater: Automated Scoring of Short-Answer Questions. *Computers and the Humanities*, 37(4):389–405.

M.E. Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the SIGDOC Conference 1986*, Toronto, June.

D. Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, Madison, WI.

B. MacCartney, T. Grenager, M.C. de Marneffe, D. Cer, and C.D. Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, page 48. Association for Computational Linguistics.

K.I. Malatesta, P. Wiemer-Hastings, and J. Robertson. 2002. Beyond the Short Answer Question with Research Methods Tutor. In *Proceedings of the Intelligent Tutoring Systems Conference*.

R. Mihalcea, C. Corley, and C. Strapparava. 2006. Corpus-based and knowledge-based approaches to text semantic similarity. In *Proceedings of the American Association for Artificial Intelligence (AAAI 2006)*, Boston.

T. Mitchell, T. Russell, P. Broomhead, and N. Aldridge. 2002. Towards robust computerised marking of free-text responses. *Proceedings of the 6th International Computer Assisted Assessment (CAA) Conference*.

M. Mohler and R. Mihalcea. 2009. Text-to-text semantic similarity for automatic short answer grading. In *Proceedings of the European Association for Computational Linguistics (EACL 2009)*, Athens, Greece.

R.D. Nielsen, W. Ward, and J.H. Martin. 2009. Recognizing entailment in intelligent tutoring systems. *Natural Language Engineering*, 15(04):479–501.

T. Pedersen, S. Patwardhan, and J. Michelizzi. 2004. WordNet:: Similarity-Measuring the Relatedness of Concepts. *Proceedings of the National Conference on Artificial Intelligence*, pages 1024–1025.

S.G. Pulman and J.Z. Sukkarieh. 2005. Automatic Short Answer Marking. *ACL WS Bldg Ed Apps using NLP*.

R. Raina, A. Haghighi, C. Cox, J. Finkel, J. Michels, K. Toutanova, B. MacCartney, M.C. de Marneffe, C.D. Manning, and A.Y. Ng. 2005. Robust textual inference using diverse knowledge sources. *Recognizing Textual Entailment*, page 57.

P. Resnik. 1995. Using information content to evaluate semantic similarity. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Montreal, Canada.

V. Rus, A. Graesser, and K. Desai. 2007. Lexico-syntactic subsumption for textual entailment. *Recent Advances in Natural Language Processing IV: Selected Papers from RANLP 2005*, page 187.

J.Z. Sukkarieh, S.G. Pulman, and N. Raikes. 2004. Auto-Marking 2: An Update on the UCLES-Oxford University research into using Computational Linguistics to Score Short, Free Text Responses. *International Association of Educational Assessment, Philadephia*.

P. Wiemer-Hastings, K. Wiemer-Hastings, and A. Graesser. 1999. Improving an intelligent tutor's comprehension of students with Latent Semantic Analysis. *Artificial Intelligence in Education*, pages 535–542.

P. Wiemer-Hastings, E. Arnott, and D. Allbritton. 2005. Initial results and mixed directions for research methods tutor. In *AIED2005 - Supplementary Proceedings of the 12th International Conference on Artificial Intelligence in Education*, Amsterdam.

Z. Wu and M. Palmer. 1994. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, New Mexico.

B. Zadrozny and C. Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. Edmonton, Alberta.

Age Prediction in Blogs: A Study of Style, Content, and Online Behavior in Pre- and Post-Social Media Generations

Sara Rosenthal
Department of Computer Science
Columbia University
New York, NY 10027, USA
sara@cs.columbia.edu

Kathleen McKeown
Department of Computer Science
Columbia University
New York, NY 10027, USA
kathy@cs.columbia.edu

Abstract

We investigate whether wording, stylistic choices, and online behavior can be used to predict the age category of blog authors. Our hypothesis is that significant changes in writing style distinguish pre-social media bloggers from post-social media bloggers. Through experimentation with a range of years, we found that the birth dates of students in college at the time when social media such as AIM, SMS text messaging, MySpace and Facebook first became popular, enable accurate age prediction. We also show that internet writing characteristics are important features for age prediction, but that lexical content is also needed to produce significantly more accurate results. Our best results allow for 81.57% accuracy.

1 Introduction

The evolution of the internet has changed the way that people communicate. The introduction of instant messaging, forums, social networking and blogs has made it possible for people of every age to become authors. The users of these social media platforms have created their own form of unstructured writing that is best characterized as informal. Even *how* people communicate has dramatically changed, with multitasking increasing and responses generated immediately. We should be able to exploit those differences to automatically determine from blog posts whether an author is part of a *pre-* or *post-social media* generation. This problem is called *age prediction* and raises two main questions:

- Is there a point in time that proves to be a significantly better dividing line between pre and post-social media generations?
- What features of communication most directly reveal the generation in which a blogger was born?

We hypothesize that the dividing line(s) occur when people in *generation Y*[1], or the *millennial generation*, (born anywhere from the mid-1970s to the early 2000s) were typical college-aged students (18-22). We focus on this generation due to the rise of popular social media technologies such as messaging and online social networks sites that occurred during that time. Therefore, we experimented with binary classification into age groups using all birth dates from 1975 through 1988, thus including students from generation Y who were in college during the emergence of social media technologies. We find five years where binary classification is significantly more accurate than other years: 1977, 1979, and 1982-1984. The appearance of social media technologies such as AOL Instant Messenger (AIM), weblogs, SMS text messaging, Facebook and MySpace occurred when people with these birth dates were in college.

We explore two of these years in more detail, 1979 and 1984, and examine a wide variety of

[1] http://en.wikipedia.org/wiki/Generation_Y

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 763–772,
Portland, Oregon, June 19-24, 2011.

features that differ between the pre-social media and post-social media bloggers. We examine lexical-content features such as collocations and part-of-speech collocations, lexical-stylistic features such as internet slang and capitalization, and features representing online behavior such as time of post and number of friends. We find that both stylistic and content features have a significant impact on age prediction and show that, for unseen blogs, we are able to classify authors as born before or after 1979 with 80% accuracy and born before or after 1984 with 82% accuracy.

In the remainder of this paper, we first discuss work to date on age prediction for blogs and then present the features that we extracted, which is a larger set than previously explored. We then turn separately to three experiments. In the first, we implement a prior approach to show that we can produce a similar outcome. In the second, we show how the accuracy of age prediction changes over time and pinpoint when major changes occur. In the last experiment, we describe our age prediction experiments in more detail for the most significant years.

2 Related Work

In previous work, Mackinnon (2006) , used LiveJournal data to identify a blogger's age by examining the mean age of his peer group using his social network and not just his immediate friends. They were able to predict the correct age within +/-5 years at 98% accuracy. This approach, however, is very different from ours as it requires access to the age of each of the blogger's friends. Our approach uses only a body of text written by a person along with his blogging behavior to determine which age group he is more closely identified with.

Initial research on predicting age without using the ages of friends focuses on identifying important candidate features, including blogging characteristics (e.g., time of post), text features (e.g., length of post), and profile information (e.g., interests) (Burger and Henderson, 2006). They aimed at binary prediction of age, classifying LiveJournal bloggers as either over or under 18, but were unable to automatically predict age with more accuracy than a baseline model that always chose the majority class. In our study on determining the ideal age split we did not find 18 (bloggers born in 1986 in their dataset) to be significant.

Prior work by Schler et al. (2006) has examined metadata such as gender and age in blogger.com bloggers. In contrast to our work, they examine bloggers based on their age at the time of the experiment, whether in the 10's, 20's or 30's age bracket. They identify interesting changes in content and style features across categories, in which they include blogging words (e.g., "LOL"), all defined by the Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2007). They did not use characteristics of online behavior (e.g., friends). They can distinguish between bloggers in the 10's and in the 30's with relatively high accuracy (above 96%) but many 30s are misclassified as 20s, which results in a overall accuracy of 76.2%. We re-implement Schler et al.'s work in section 5.1 with similar findings. Their work shows that ease of classification is dependent in part on what division is made between age groups and in turn motivates our decision to study whether the creation of social media technologies can be used to find the dividing line(s). Neither Schler et al., nor we, attempt to determine how a person's writing changes over his lifespan (Pennebaker and Stone, 2003; Robins et al., 2002). Goswami et al. (2009) add to Schler et al.'s approach using the same data and have a 4% increase in accuracy. However, the paper is lacking details and it is entirely unclear how they were able to do this with fewer features than Schler et al.

In other work, Tam and Martell (2009) attempt to detect age in the NPS chat corpus between teens and other ages. They use an SVM classifier with only n-grams as features. They achieve $> 90\%$ accuracy when classifying teens vs 30s, 40s, 50s, and all adults and achieve at best 76% when using 3 character gram features in classifying teens vs 20s. This work shows that n-grams are useful features for detecting age and it is difficult to detect differences between consecutive groups such as teens and 20s, and this

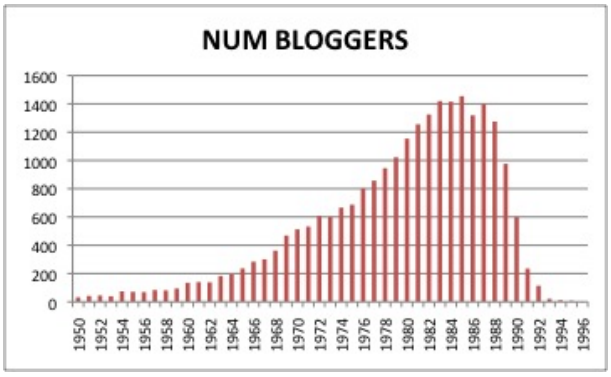


Figure 1: Number of bloggers in 2010 by year of birth from 1950-1996. A minimal amount of data occurred in years not shown.

provides evidence for the need to find a good classification split.

Other researchers have investigated weblogs for differences in writing style depending on gender identification (Herring and Paolillo, 2006; Yan and Yan, 2006; Nowson and Oberlander, 2006). Herring et al (2006) found that the typical gender related features were based on genre and independent of author gender. Yan et al (2006) used text categorization and stylistic web features, such as emoticons, to identify gender and achieved 60% F-measure. Nowson et al (2006) employed dictionary and n-gram based content analysis and achieved 91.5% accuracy using an SVM classifier. We also use a supervised machine learning approach, but classification by gender is naturally a binary classification task, while our work requires determining a natural dividing point.

3 Data Collection

Our corpus consists of blogs downloaded from the virtual community LiveJournal. We chose to use LiveJournal blogs for our corpus because the website provides an easy-to-use format in XML for downloading and crawling their site. In addition, LiveJournal gives bloggers the opportunity to post their age on their profile. We take advantage of this feature by downloading blogs where the user chooses to publicly provide this metadata.

We downloaded approximately 24,500 LiveJournal blogs containing age. We represent age as the year a person was born and not his age at the time of the experiment. Since technology has different effects in different countries, we only analyze the blogs of people who have listed US as their country. It is possible that text written in a language other than English is included in our corpus. However, in a manual check of a small portion of text from 500 blogs, we only found English words. Each blog was written by a unique individual and includes a user profile and up to 25 recent posts written between 2000-2010 with the most recent post being written in 2009-2010. The birth dates of the bloggers range in years from 1940 to 2000 and thus, their age ranges from 10 to 70 in 2010. Figure 1 shows the number of bloggers per age in our group with birth dates from 1950 to 1996. The majority of bloggers on LiveJournal were born between 1978-1989.

4 Methods

We pre-processed the data to add Part-of-Speech tags (POS) and dependencies (de Marneffe et al., 2006) between words using the Stanford Parser (Klein and Manning, 2003a; Klein and Manning, 2003b). The POS and syntactic dependencies were only found for approximately the first 90 words in each sentence. Our classification method investigates 17 different features that fall into three categories: online behavior, lexical-stylistic and lexical-content. All of the features we used are explained in Table 1 along with their trend as age decreases where applicable. Any feature that increased, decreased, or fluctuated should have some positive impact on the accuracy of predicting age.

4.1 Online Behavior and Interests

Online behavior features are blog specific, such as number of *comments* and *friends* as described in Table 1.1. The first feature, *interests*, is our only feature that is specific to LiveJournal. Interests appear in the LiveJournal user profile, but are not found on all blog sites. All other online behavior features are typically available in any blog.

	Feature	Explanation	Example	Trend as Age Decreases
1	Interests	Top[3] interests provided on the profile page[2]	disney	N/A
2	# of Friends	Number of friends the blogger has	45	fluctuates
	# of Posts	Number of downloadable posts (0-25)	23	decrease
	# of Lifetime Posts	Number of posts written in total	821	decrease
	Time	Mode hour (00-23) and day the blogger posts	11/Monday	no change
	Comments	Average number of comments per post	2.64	increase
3	Emoticons	number of emoticons[1]	:)	increase
	Acronyms	number of internet acronyms[1]	lol	increase
	Slang	number of words that are not found in the dictionary[1]	wazzup	increase
	Punctuation	number of stand-alone punctuation[1]	...	increase
	Capitalization	number of words (with length > 1) that are all CAPS[1]	YOU	increase
	Sentence Length	average sentence length	40	decrease
	Links/Images	number of url and image links[1]	www.site.com	fluctuates
4	Collocations	Top[3] Collocations in the age group.	to [] the	N/A
	Syntax Collocations	Top[3] Syntax Collocations in the age group.	best friends	N/A
	POS Collocations	Top[3] Part-of-Speech Collocations in the age group.	this [] [] VB	N/A
	Words	Top[3] words in the age group	his	N/A

Table 1: List of all features used during classification divided into three categories (1,2) online behavior and interests, (3) lexical - content, and (4) lexical - stylistic [1] normalized per sentence per entry, [2] available in LiveJournal only, [3] pruned from top 200 features to include those that do not occur within +/- 10 position in any other age group

We extracted the top 200 interests based on occurrence in the profile page from 1500 random blogs in three age groups. These age groups are used solely to illustrate the differences that occur at different ages and are not used in our classification experiments. We then pruned the list of interests by excluding any interest that occurred within a +/-10 window (based on its position in the list) in multiple age groups. We show the top interests in each age group in Table 2. For example, "disney" is the most popular unique interest in the 18-22 age group with only 39 other non-unique interests in that age group occurring more frequently. "Fanfiction" is a popular interest in all age groups, but it is significantly more popular in the 18-22 age group than in other age groups.

18-22		28-32		38-42	
disney	39	tori amos	49	polyamory	40
yaoi	40	hiking	55	sca	67
johnny depp	42	women	61	babylon 5	84
rent	44	gaming	62	leather	94
house	45	comic books	67	farscape	103
fanfiction	11	fanfiction	58	fanfiction	138
drawing	10	drawing	25	drawing	65
sci-fi	199	sci-fi	37	sci-fi	21

Table 2: Top interests for three different age groups. The top half refers to the top 5 interests that are unique to each age group. The value refers to the *position of the interest in its list*

Amongst the other online behavior features, the number of friends tends to fluctuate but seems to be higher for older bloggers. The number of *lifetime posts* (Figure 2(d)), and *posts* decreases as bloggers get younger which is as one would expect unless younger people were orders of magnitude more prolific than older people. The *mode time* (Figure 2(b)), refers to the most common hour of posting from 00-24 based on GMT time. We didn't compute time based on the time zone because city/state is often not included. We found time to not be a useful feature in this manner and it is difficult to come to any conclusions from its change as year of birth decreases.

4.2 Lexical - Stylistic

The Lexical-Stylistic features in Table 1.2, such as *slang* and *sentence length*, are computed us-

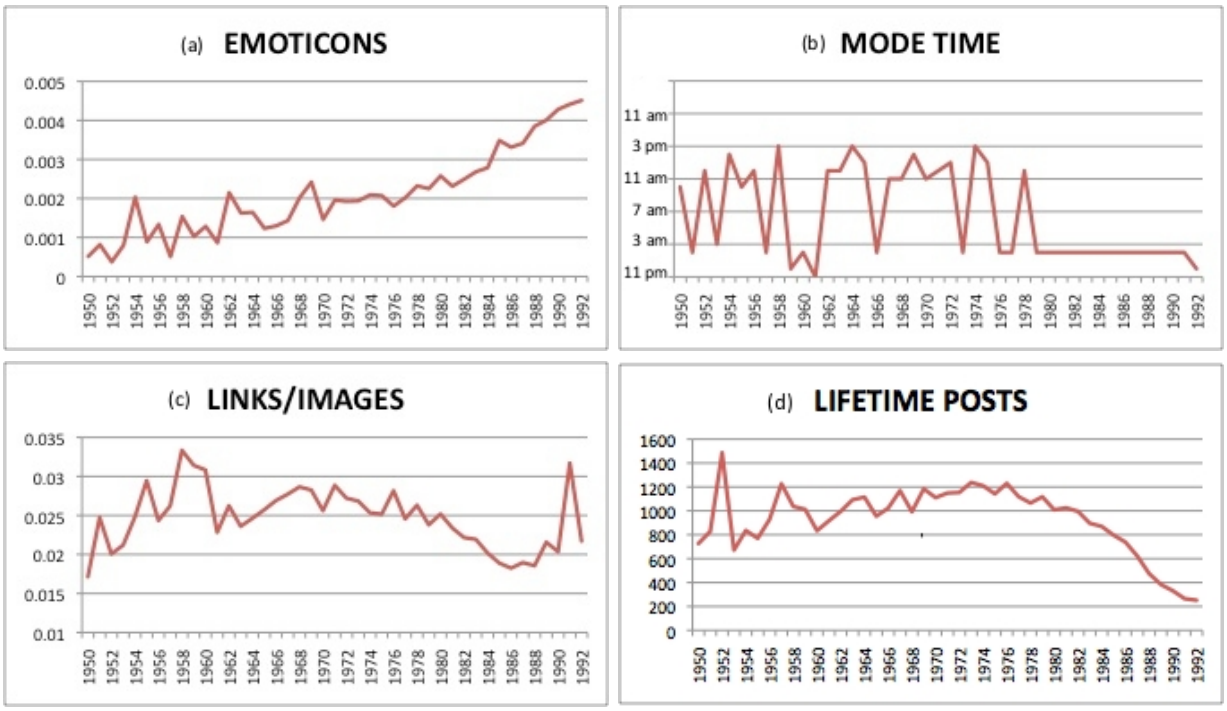


Figure 2: Examples of change to features over time (a) Average number of emoticons in a sentence increases as age decreases (b) The most common time fluctuates until 1982, where it is consistent (c) The number of links/images in a sentence fluctuates (d) The average number of lifetime posts per year decreases as age decreases

ing the text from all of the posts written by the blogger. Other than *sentence length*, they were normalized by sentence and post to keep the numbers consistent between bloggers regardless of whether the user wrote one or many posts in his/her blog. The number of *emoticons* (Figure 2(a)), *acronyms*, and *capital words* increased as bloggers got younger. *Slang* and *punctuation*, which excludes the emoticons and acronyms counted in the other features, increased as well, but not as significantly. The *length of sentences* decreased as bloggers got younger and the number of *links/images* varied across all years as shown in Figure 2(c).

4.3 Lexical - Content

The last category of features described in Table 1.3 consists of collocations and words, which are content based lexical terms. The top words are produced using a typical "bag-of-words" approach. The top collocations are computed using a system called *Xtract* (Smadja, 1993).

We use Xtract to obtain important lexical collocations, syntactic collocations, and POS collocations as features from our text. *Syntactic collocations* refer to significant word pairs that have specific syntactic dependencies such as subject/verb and verb/object. Due to the length of time it takes to run this program, we ran Xtract on 1500 random blogs from each age group and examined the first 1000 words per blog. We looked at 1.5 million words in total and found approximately 2500-2700 words that were repeated more than 50 times.

We extracted the top 200 words and collocations sorted by *post frequency* (pf), which is the number of posts the term occurred in. Then, similarly to interests, we pruned each list to include the features that did not occur within +/-10 window (based on its position in the list) within each age group. Prior to settling on these metrics, we also experimented with other metrics such as the number of times the collocation

18-22		28-32		38-42	
ldquot (')	101	great	166	may	164
t	152	find	167	old	183
school	172	many	177	house	191
x	173	years	179	world	192
anything	175	week	181	please	198
maybe	179	post	190	-	-
because	68	because	80	because	93
him	59	him	85	him	73

Table 3: Top words for three age groups. The top half refers to the top 5 words that are unique to each age group. The value refers to the *position of the interest in its list*

	blogger.com			livejournal.com		
download year	2004			2010		
# of Blogs	19320			11521		
# of Posts[1]	1.4 million			256,000		
# of words[1]	295 million			50 million		
age	13·17	23·27	33·37	18·22	28·32	38·42
size	8240	8086	2994	3518	5549	2454
majority baseline	43.8% (13-17)			48.2% (22-32)		

Table 4: Statistics for Schler et al.'s data (blogger.com) vs our data (livejournal.com) [1] is approximate amount.

occurred in total, defined as *collocation* or *term frequency* (tf), the number of blogs the collocation occurred in, defined as *blog frequency* (bf), and variations of TF*IDF (Salton and Buckley, 1988) where we tried using inverse blog frequency and inverse post frequency as the value for IDF. In addition, we also experimented with looking at a different number of important words and collocations ranging from the top 100-300 terms and experimented without pruning. None of these variations improved accuracy in our experiments, however, and thus, were dropped from further experimentation.

Table 3 shows the top words for each age group; older people tend to use words such as "house" and "old" frequently and younger people talk about "school".

In our analysis of the top collocations, we found that younger people tend to use first person singular (*I,me*) in subject position while older people tend to use first person plural (*we*) in subject position, both with a variety of verbs.

5 Experiments and Results

We ran three separate experiments to determine how well we can predict age: 1. classifying into three distinct age groups (Schler et al. (2006) experiment), 2. binary classification with the split at each birth year from 1975-1988 and 3. Detailed classification on two significant splits from the second experiment.

We ran all of our experiments in Weka (Hall et al., 2009) using logistic regression over 10 runs of 10-fold cross-validation. All values shown are the averages of the accuracies from the 10 cross-validation runs and all results were compared for statistical significance using the t-test where applicable.

We use logistic regression as our classifier because it has been shown that logistic regression typically has lower asymptotic error than naive Bayes for multiple classification tasks as well as for text classification (Ng and Jordan, 2002). We experimented with an SVM classifier and found logistic regression to do slightly better.

5.1 Age Groups

The first experiment implements a variation of the experiment done by Schler et al. (2006). The differences between the two datasets are shown in Tables 4. The experiment looks at three age groups containing a 5-year gap between each group. Intermediate years were not included to provide clear differentiation between the groups because many of the blogs have been active for several years and this will make it less common for a blogger to have posts that fall into two age groups (Schler et al., 2006).

We did not use the same age groups as Schler et al. because very few blogs on LiveJournal, in 2010, are in the 13-17 age group. Many early demographic studies (Perseus Development, 2004; Herring et al., 2004) show teens as the dominant age group in all blogs. However, more recent studies (Nowson and Oberlander, 2006; Lenhart et al., 2010) show that less teens blog. Furthermore, an early study on the LiveJournal

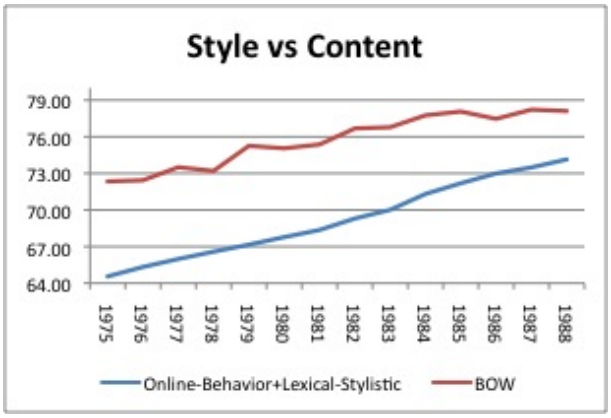


Figure 3: Style vs Content: Accuracy from 1975-1988 for Style (Online-Behavior+Lexical-Stylistic) vs Content (BOW)

demographic (Kumar et al., 2004) reported that 28.6% of blogs are written by bloggers between the ages 13-18 whereas based on the current demographic statistics, in 2010[2], only 6.96% of blogs are written by that age group and the number of bloggers in the 31-36 age group increased from 3.9% to 12.08%. We chose the later age groups because this study is based on blogs updated in 2009-10 which is 5-6 years later and thus, the 13-17 age group is now 18-22 and so on.

We use style-based (lexical-stylistic) and content-based features (BOW, interests) to mimic Schler et al.'s experiment as closely as possible and also experimented with adding online-behavior features. Our experiment with style-based and content-based features had an accuracy of 57%. However, when we added online-behavior, we increased our accuracy to 67%. A more detailed look at the better results show that our accuracies are consistently 7% lower than the original work but we have similar findings; 18-22s are distinguishable from 38-42s with accuracy of 94.5%, and 18-22s are distinguishable from 28-32s with accuracy of 80.5%. However, many 38-42s are misclassified as 28-32s with an accuracy of 72.1%, yielding overall accuracy of 67%. Due to our findings, we believe that adding online-behavior features to Schler et al.'s dataset would improve their results as well.

[2]http://www.livejournal.com/stats.bml

5.2 Social Media and Generation Y

In the first experiment we used the current age of a blogger based on when he wrote his last post. However, the age of a person changes; someone who was in one age group now will be in a different age group in 5 years. Furthermore, a blogger's posts can fall into two categories depending on his age at the time. Therefore, our second experiment looks at year of birth instead of age, as that never changes. In contrast to Schler et al.'s experiment, our division does not introduce a gap between age groups, we do binary classification, and we use significantly less data.

We approach age prediction as attempting to identify a shift in writing style over a 14 year time span from birth years 1975-1988:

For each year X = 1975-1988:

- get 1500 blogs (∼33,000 posts) balanced across years BEFORE X
- get 1500 blogs (∼33,000 posts) balanced across years IN/AFTER X
- Perform binary classification between blogs BEFORE X and IN/AFTER X

The experiment focuses on the range of birth years of bloggers from 1975-1888 to identify at what point in time, if any, shift(s) in writing style occurred amongst college-aged students in generation Y. We were motivated to examine these years due to the emergence of social media technologies during that time. Furthermore, research by Pew Internet (Zickuhr, 2010) has found that this generation (defined as 1977-1992 in their research) uses social networking, blogs, and instant messaging more than their elders. The experiment is balanced to ensure that each birth year is evenly represented. We balance the data by choosing a blogger consecutively from each birth year in the category, repeating these sweeps through the category until we have obtained 1500 blogs. We chose to use 1500 blogs from each group because of processing power, time constraints, and the amount of blogs needed to reasonably sample the age group at each split. Due to the extensive running time, we only examined variations of a combination of

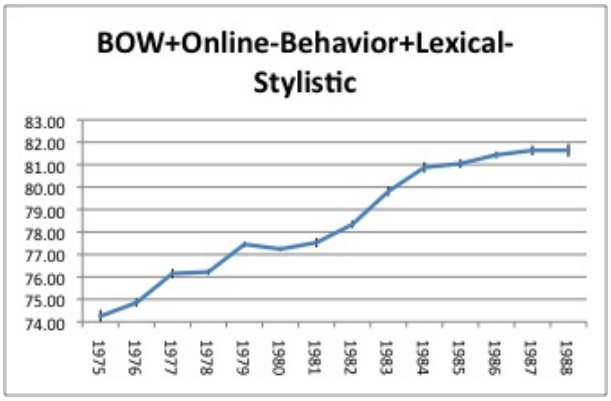


Figure 4: Style and Content: Accuracy from 1975-1988 using BOW, Online Behavior, and Lexical-Stylistic features

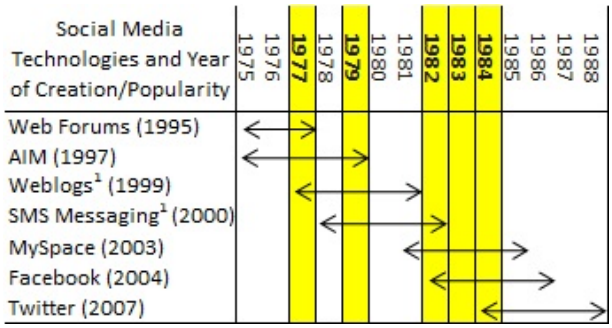


Figure 5: The impact of social media technologies: The arrows correspond to the years that generation Yers were college aged students. The highlighted years represent the significant years. [1]Year it became popular (Urmann, 2009)

online-behavior, lexical-stylistic, and BOW features.

We found accuracy to increase as year of birth increases in various feature experiments which is consistent with the trends we found while examining the distribution of features such as emoticons and lifetime posts in Figure 2. We experimented with style and content features and found that both help improve accuracy. Figure 3 shows that content helps more than style, but style helps more as age decreases. However, as shown in Figure 4, style and content combined provided the best results. We found 5 years to have significant improvement over all prior years for $p \leq .0005$: 1977, 1979, and 1982-1984.

Generation Y is considered the social media generation, so we decided to examine how the creation and/or popularity of social media technologies compared to the years that had a change in writing style. We looked at many popular social media technologies such as weblogs, messaging, and social networking sites. Figure 5 compares the significant years 1977,1979, and 1982-1984 against when each technology was created or became popular amongst college aged students. We find that all the technologies had an effect on one or more of those years. AIM and weblogs coincide with the earlier shifts at 1977 and 1979, SMS messaging coincide with both the earlier and later shifts at 1979 and 1982, and the social networking sites, MySpace and Facebook coincide with the later shifts of 1982-1984. On the other hand, web forums and Twitter each coincide with only one outlying year which suggests that either they had less of an impact on writing style or, in the case of Twitter, the change has not yet been transferred to other writing forms.

5.3 A Closer Look: 1979 and 1984

Our final experiment provides a more detailed explanation of the results using various feature combinations when splitting pre- and post- social media bloggers by year of birth at two of the significant years found in the previous section; 1979 and 1984. The results for all of the experiments described are shown in Table 5.

We experimented against two baselines, online behavior and interests. We chose these two features as baselines because they are both easy to generate and not lexical in nature. We found that we were able to exceed the baselines significantly using a simple bag-of-words (BOW) approach. This means the BOW does a better job of picking topics than interests. We found that including all 17 features did not do well, but we were able to get good results using a subset of the lexical features. We found the best results to have an accuracy of 79.96% and 81.57% for 1979 and 1984 respectively using BOW, interests, online behavior, and all lexical-stylistic features.

In addition, we show accuracy without interests since they are not always available.

Experiment	1979	1984
Online-Behavior	59.66	61.61
Interests	70.22	74.61
Lexical-Stylistic	65.38[2]	67.28[2]
Slang+Emoticons+Acronyms	60.57[2]	62.10[2]
Online-Behavior + Lexical-Stylistic	67.16[2]	71.31[2]
Collocations + Syntax Collocations	53.47[1]	73.45[2]
POS-Collocations + POS-Syntax Collocations	55.54[1]	74.00[2]
BOW	75.26	77.76
BOW+Online-Behavior	76.39	79.22
BOW + Online-Behavior + Lexical-Stylistic	**77.45**	**80.88**
BOW + Online-Behavior + Lexical-Stylistic + Syntax Collocations	74.8	80.36
BOW + Online-Behavior + Lexical-Stylistic + POS-Collocations + POS Syntax Collocations	74.73	80.54
Online-Behavior + Interests + Lexical-Stylistic	74.39	77.20
BOW + Online-Behavior + Interests + Lexical-Stylistic	**79.96**	**81.57**
All Features	71.26	74.07[2]

Table 5: Feature Accuracy. The top portion refers to the baselines. The best accuracies are shown in bold. Unless otherwise marked, all accuracies are statistically significant at p<=.0005 for both baselines. [1] not statistically significant over Online-Behavior and Interests. [2] not statistically significant over Interests.

BOW, online-behavior, and lexical-stylistic features combined did best achieving accuracy of 77.45% and 80.88% in 1979 and 1984 respectively. This indicates that our classification method could work well on blogs from any website. It is interesting to note that collocations and POS-collocations were useful, but only when we use 1984 as the split which implies that bloggers born in 1984 and later are more homogeneous.

6 Conclusion and Future Work

We have shown that it is possible to predict the age group of a person based on style, content, and online behavior features with good accuracy; these are all features that are available in any blog. While features representing writing practices that emerged with social media (e.g., capitalized words, abbreviations, slang) do not significantly impact age prediction on their own, these features have a clear change of value across time, with post-social media bloggers using them more often. We found that the birth years that had a significant change in writing style corresponded to the birth dates of college-aged students at the time of the creation/popularity of social media technologies, AIM, SMS text messaging, weblogs, Facebook and MySpace.

In the future we plan on using age and other metadata to improve results in larger tasks such as identifying opinion, persuasion and power by targeting our approach in those tasks to the identified age of the person. Another approach that we will experiment with is the use of ranking, regression, and/or clustering to create meaningful age groups.

7 Acknowledgements

This research was funded by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), through the U.S. Army Research Lab. All statements of fact, opinion or conclusions contained herein are those of the authors and should not be construed as representing the official views or policies of IARPA, the ODNI or the U.S. Government.

References

Shlomo Argamon, Moshe Koppel, Jonathan Fine, and Anat Rachel Shimoni. 2003. Gender, genre, and writing style in formal written texts. *TEXT*, 23:321–346.

John D. Burger and John C. Henderson. 2006. An exploration of observable features related to blogger age. In *AAAI Spring Symposia*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *In LREC 2006*.

Sumit Goswami, Sudeshna Sarkar, and Mayur Rustagi. 2009. Stylometric analysis of bloggers'

age and gender. In *International AAAI Conference on Weblogs and Social Media*.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: An update.

Susan C. Herring and John C. Paolillo. 2006. Gender and genre variation in weblogs. *Journal of Sociolinguistics*, 10(4):439–459.

Susan C. Herring, L.A. Scheidt, S. Bonus, and E. Wright. 2004. Bridging the gap: A genre analysis of weblogs. In *Proceedings of the 37th Hawaii International Conference on System Sciences*.

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

Dan Klein and Christopher D. Manning. 2003b. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems*, volume 15. MIT Press.

Ravi Kumar, Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. 2004. Structure and evolution of blogspace. *Commun. ACM*, 47:35–39, December.

Amanda Lenhart, Kristen Purcell, Aaron Smith, and Kathryn Zickuhr. 2010. Social media and young adults.

Ian Mackinnon. 2006. Age and geographic inferences of the livejournal social network. In *In Statistical Network Analysis Workshop*.

Andrew Y Ng and Michael I Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Neural Information Processing Systems*, 2:841–848.

Scott Nowson and Jon Oberlander. 2006. The identity of bloggers: Openness and gender in personal weblogs.

James W Pennebaker and Lori D Stone. 2003. Words of wisdom: language use over the life span. *J Pers Soc Psychol*, 85(2):291–301.

J.W. Pennebaker, R.E. Booth, and M.E. Francis. 2007. Linguistic inquiry and word count: Liwc2007 Đ operatorÕs manual. Technical report, LIWC, Austin, TX.

Perseus Development. 2004. The blogging iceberg: Of 4.12 million hosted weblogs, most little seen and quickly abandoned. Technical report, Perseus Development.

R.W. Robins, K. H. Trzesniewski, J.L. Tracy, S.D Gosling, and J Potter. 2002. Global self-esteem across the lifespan. *Psychology and Aging*, 17:423–434.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. In *Information Processing and Management*, pages 513–523.

J. Schler, M. Koppel, S. Argamon, and J. Pennebaker. 2006. Effects of age and gender on blogging. In *AAAI Spring Symposium on Computational Approaches for Analyzing Weblogs*.

Frank Smadja. 1993. Retrieving collocations from text: Xtract. *Computational Linguistics*, 19:143–177.

Jenny Tam and Craig H. Martell. 2009. Age detection in chat. In *Proceedings of the 2009 IEEE International Conference on Semantic Computing*, ICSC '09, pages 33–39, Washington, DC, USA. IEEE Computer Society.

David H. Urmann. 2009. The history of text messaging.

Xiang Yan and Ling Yan. 2006. Gender classification of weblog authors. In *AAAI Spring Symposium Series on Computation Approaches to Analyzing Weblogs*, pages 228–230.

Kathryn Zickuhr. 2010. Generations 2010.

Extracting Social Power Relationships from Natural Language

Philip Bramsen
Louisville, KY
bramsen@alum.mit.edu*

Martha Escobar-Molano
San Diego, CA
mescobar@asgard.com*

Ami Patel
Massachusetts Institute of Technology
Cambridge, MA
ampatel@mit.edu*

Rafael Alonso
SET Corporation
Arlington, VA
ralonso@setcorp.com

Abstract

Sociolinguists have long argued that social context influences language use in all manner of ways, resulting in *lects*[1]. This paper explores a text classification problem we will call *lect modeling*, an example of what has been termed computational sociolinguistics. In particular, we use machine learning techniques to identify *social power relationships* between members of a social network, based purely on the content of their interpersonal communication. We rely on statistical methods, as opposed to language-specific engineering, to extract features which represent vocabulary and grammar usage indicative of social power lect. We then apply support vector machines to model the social power lects representing superior-subordinate communication in the Enron email corpus. Our results validate the treatment of lect modeling as a text classification problem – albeit a hard one – and constitute a case for future research in computational sociolinguistics.

1 Introduction

Linguists in sociolinguistics, pragmatics and related fields have analyzed the influence of social context on language and have catalogued countless phenomena that are influenced by it, confirming many with qualitative and quantitative studies. Indeed, social context and function influence language at every level – morphologically, lexically, syntactically, and semantically, through discourse structure, and through higher-level abstractions such as pragmatics.

Considered together, the extent to which speakers modify their language for a social context amounts to an identifiable variation on language, which we call a *lect*. *Lect* is a backformation from words such as *dialect* (geographically defined language) and e*thnolect* (language defined by ethnic context).

In this paper, we describe *lect classifiers* for *social power relationships*. We refer to these lects as:

- *UpSpeak*: Communication directed to someone with greater social authority.
- *DownSpeak*: Communication directed to someone with less social authority.
- *PeerSpeak*: Communication to someone of equal social authority.

We call the problem of modeling these lects *Social Power Modeling* (SPM). The experiments reported in this paper focused primarily on modeling UpSpeak and DownSpeak.

Manually constructing tools that effectively model specific linguistic phenomena suggested by sociolinguistics would be a Herculean effort. Moreover, it would be necessary to repeat the effort in every language! Our approach first identifies statistically salient phrases of words and parts of speech – known as *n-grams* – in training texts generated in conditions where the social power

* This work was done while these authors were at SET Corporation, an SAIC Company.

[1] Fields that deal with society and language have inconsistent terminology; "lect" is chosen here because "lect" has no other English definitions and the etymology of the word gives it the sense we consider most relevant.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 773–782,
Portland, Oregon, June 19-24, 2011.

relationship is known. Then, we apply machine learning to train classifiers with groups of these n-grams as features. The classifiers assign the UpSpeak and DownSpeak labels to unseen text. This methodology is a cost-effective approach to modeling social information and requires no language- or culture-specific feature engineering, although we believe sociolinguistics-inspired features hold promise.

When applied to the corpus of emails sent and received by Enron employees (CALO Project 2009), this approach produced solid results, despite a limited number of training and test instances.

This has many implications. Since manually determining the power structure of social networks is a time-consuming process, even for an expert, effective SPM could support data driven sociocultural research and greatly aid analysts doing national intelligence work. Social network analysis (SNA) presupposes a collection of individuals, whereas a social power lect classifier, once trained, would provide useful information about individual author-recipient links. On networks where SNA already has traction, SPM could provide complementary information based on the content of communications.

If SPM were yoked with sentiment analysis, we might identify which opinions belong to respected members of online communities or lay the groundwork for understanding how respect is earned in social networks.

More broadly, computational sociolinguistics is a nascent field with significant potential to aid in modeling and understanding human relationships. The results in this paper suggest that successes to date modeling authorship, sentiment, emotion, and personality extend to social power modeling, and our approach may well be applicable to other dimensions of social meaning.

In the coming sections, we first establish the **Related Work**, primarily from Statistical NLP. We then cover our **Approach,** the **Evaluation**, and, finally, the **Conclusions and Future Research**.

2 Related Work

The feasibility of Social Power Modeling is supported by sociolinguistic research identifying specific ways in which a person's language reflects his relative power over others. Fairclough's classic work *Language and Power* explores how "sociolinguistic conventions . . . arise out of -- and give rise to – particular relations of power" (Fairclough, 1989). Brown and Levinson created a theory of politeness, articulating a set of strategies which people employ to demonstrate different levels of politeness (Brown & Levinson, 1987). Morand drew upon this theory in his analysis of emails sent within a corporate hierarchy; in it, he quantitatively showed that emails from subordinates to superiors are, in fact, perceived as more polite, and that this perceived politeness is correlated with specific linguistic tactics, including ones set out by Brown and Levinson (Morand, 2000). Similarly, Erikson et al identified measurable characteristics of the speech of witnesses in a courtroom setting which were directly associated with the witness's level of social power (Erikson, 1978). Given, then, that there are distinct differences among what we term UpSpeak and DownSpeak, we treat Social Power Modeling as an instance of *text classification* (or *categorization*): we seek to assign a class (UpSpeak or DownSpeak) to a text sample. Closely related natural language processing problems are authorship attribution, sentiment analysis, emotion detection, and personality classification: all aim to extract higher-level information from language.

Authorship attribution in computational linguistics is the task of identifying the author of a text. The earliest modern authorship attribution work was (Mosteller & Wallace, 1964), although forensic authorship analysis has been around much longer. Mosteller and Wallace used statistical language-modeling techniques to measure the similarity of disputed Federalist Papers to samples of known authorship. Since then, authorship identification has become a mature area productively exploring a broad spectrum of features (stylistic, lexical, syntactic, and semantic) and many generative and discriminative modeling approaches (Stamatatos, 2009). The generative models of authorship identification motivated our statistically extracted lexical and grammatical features, and future work should consider these language modeling (a.k.a. compression) approaches.

Sentiment analysis, which strives to determine the attitude of an author from text, has recently garnered much attention (e.g. Pang, Lee, & Vaithyanathan, 2002; Kim & Hovy, 2004; Breck, Choi

& Cardie, 2007). For example, one problem is classifying user reviews as positive, negative or neutral. Typically, polarity lexicons (each term is labeled as positive, negative or neutral) help determine attitudes in text (Hiroya & Takamura, 2005, Ravichandran 2009, Choi & Cardie 2009).

The polarity of an expression can be determined based on the polarity of its component lexical items (Choi & Cardie 2008). For example, the polarity of the expression is determined by the majority polarity of its lexical items or by rules applied to syntactic patterns of expressions on how to determine the polarity from its lexical components. McDonald et al studied models that classify sentiment on multiple levels of granularity: sentence and document-level (McDonald, 2007). Their work jointly classifies sentiment at both levels instead of using independent classifiers for each level or cascaded classifiers. Similar to our techniques, these studies determine the polarity of text based on its component lexical and grammatical sequences. Unlike their works, our text classification techniques take into account the frequency of occurrence of word n-grams and part-of-speech (POS) tag sequences, and other measures of statistical salience in training data.

Text-based emotion prediction is another instance of text classification, where the goal is to detect the emotion appropriate to a text (Alm, Roth & Sproat, 2005) or provoked by an author, for example (Strapparava & Mihalcea, 2008). Alm, Roth, and Sproat explored a broad array of lexical and syntactic features, reminiscent of those of authorship attribution, as well as features related to story structure. A Winnow-based learning algorithm trained on these features convincingly predicted an appropriate emotion for individual sentences of narrative text. Strapparava and Mihalcea try to predict the emotion the author of a headline intends to provoke by leveraging words with known affective sense and by expanding those words' synonyms. They used a Naïve Bayes classifier trained on short blogposts of known emotive sense. The knowledge engineering approaches were generally superior to the Naïve Bayes approach. Our approach is corpus-driven like the Naïve Bayes approach, but we interject statistically driven feature selection between the corpus and the machine learning classifiers.

In personality classification, a person's language is used to classify him on different personality dimensions, such as extraversion or neuroticism (Oberlander & Nowson, 2006; Mairesse & Walker; 2006). The goal is to recover the more permanent traits of a person, rather than fleeting characteristics such as sentiment or emotion. Oberlander and Nowson explore using a Naïve Bayes and an SVM classifier to perform binary classification of text on each personality dimension. For example, one classifier might determine if a person displays a high or low level of extraversion. Their attempt to classify each personality trait as either "high" or "low" echoes early sentiment analysis work that reduced sentiments to either positive or negative (Pang, Lee, & Vaithyanathan, 2002), and supports initially treating Social Power Modeling as a binary classification task. Personality classification seems to be the application of text classification which is the most relevant to Social Power Modeling. As Mairesse and Walker note, certain personality traits are indicative of leaders. Thus, the ability to model personality suggests an ability to model social power lects as well.

Apart from text classification, work from the topic modeling community is also closely related to Social Power Modeling. Andrew McCallum extended Latent Dirichlet Allocation to model the author and recipient dependencies of per-message topic distributions with an Author-Recipient-Topic (ART) model (McCallum, Wang, & Corrada-Emmanuel, 2007). This was the first significant work to model the content and relationships of communication in a social network. McCallum et al applied ART to the Enron email corpus to show that the resulting topics are strongly tied to role. They suggest that clustering these topic distributions would yield roles and argue that the person-to-person similarity matrix yielded by this approach has advantages over those of canonical social network analysis. The same authors proposed several Role-Author-Recipient-Topic (RART) models to model authors, roles and words simultaneously. With a RART modeling roles-per-word, they produced per-author distributions of generated roles that appeared reasonable (e.g. they labeled Role 10 as 'grant issues' and Role 2 as 'natural language researcher').

We have a similar emphasis on statistically modeling language and interpersonal communica-

tion. However, we model social power relationships, not roles or topics, and our approach produces discriminative classifiers, not generative models, which enables more concrete evaluation.

Namata, Getoor, and Diehl effectively applied role modeling to the Enron email corpus, allowing them to infer the social hierarchy structure of Enron (Namata et al., 2006). They applied machine learning classifiers to map individuals to their roles in the hierarchy based on features related to email traffic patterns. They also attempt to identify cases of manager-subordinate relationships within the email domain by ranking emails using traffic-based and content-based features (Diehl et al., 2007). While their task is similar to ours, our goal is to classify any case in which one person has more social power than the other, not just identify instances of direct reporting.

3 Approach

3.1 Feature Set-Up

Previous work in traditional text classification and its variants – such as sentiment analysis – has achieved successful results by using the bag-of-words representation; that is, by treating text as a collection of words with no interdependencies, training a classifier on a large feature set of word unigrams which appear in the corpus. However, our hypothesis was that this approach would not be the best for SPM. Morand's study, for instance, identified specific features that correlate with the direction of communication within a social hierarchy (Morand, 2000). Few of these tactics would be effectively encapsulated by word unigrams. Many would be better modeled by POS tag unigrams (with no word information) or by longer n-grams consisting of either words, POS tags, or a combination of the two. "Uses subjunctive" and "Uses past tense" are examples. Because considering such features would increase the size of the feature space, we suspected that including these features would also benefit from algorithmic means of selecting n-grams that are indicative of particular lects, and even from *binning* these relevant n-grams into sets to be used as features.

Therefore, we focused on an approach where each feature is associated with a set of one or more n-grams. Each n-gram is a sequence of words, POS tags or a combination of words and POS tags ("mixed" n-grams). Let S represent a set $\{n_1, ..., n_k\}$ of n-grams. The feature associated with S on text T would be:

$$f(S,T) = \sum_{i=1}^{k} freq(n_i, T)$$

where $freq(n_i, T)$ is the relative frequency (defined later) of n_i in text T. Let n_i represent the sequence $s_1 \ldots s_m$ where s_j specifies either a word or a POS tag. Let T represent the text consisting of the sequence of tagged-word tokens $t_1 \ldots t_l$. $freq(n_i, T)$ is then defined as follows:

$$freq(n_i, T) = freq(s_1 \ldots s_m, T) = \frac{\left|\left\{t_{b+1} \ldots t_{b+m} : \forall_{1 \le p \le m}(t_{b+p} = s_p)\right\}\right|}{l - m + 1}$$

where:

$$t_i = s_j \leftrightarrow \begin{cases} word(t_i) = s_j \text{ if } s_j \text{ is a word} \\ tag(t_i) = s_j \text{ if } s_j \text{ is a tag} \end{cases}$$

To illustrate, consider the following feature set, a bigram and a trigram (each term in the n-gram either has the form *word* or *^tag*):

{please ^VB, please ^'comma' ^VB}[2]

The tag "VB" denotes a verb. Suppose T consists of the following tokenized and tagged text (sentence initial and final tokens are not shown):

please^RB bring^VB the^DET report^NN to^TO our^PRP$ next^JJ weekly^JJ meeting^NN .^.

The first n-gram of the set, *please ^VB*, would match *please^RB bring^VB* from the text. The frequency of this n-gram in *T* would then be 1/9, where 1 is the number of substrings in *T* that match

[2] To distinguish a comma separating elements of a set with a comma as part of an ngram, we use 'comma' to denote the punctuation mark ',' as part of the ngram.

please ^VB and 9 is the number of bigrams in *T*, excluding sentence initial and final markers. The other n-gram, the trigram *please ^'comma' ^VB,* does not have any match, so the final value of the feature is 1/9.

Defining features in this manner allows us to both explore the bag-of-words representation as well as use groups of n-grams as features, which we believed would be a better fit for this problem.

3.2 N-Gram Selection

To identify n-grams which would be useful features, frequencies of n-grams in only the training set are considered. Different types of frequency measures were explored to capture different types of information about an n-gram's usage. These are:

- *Absolute frequency*: The total number of times a particular n-gram occurs in the text of a given class (social power lect).
- *Relative frequency*: The total number of times a particular n-gram occurs in a given class, divided by the total number of n-grams in that class. Normalization by the size of the class makes relative frequency a better metric for comparing n-gram usage across classes.

We then used the following frequency-based metrics to select n-grams:

- We set a minimum threshold for the absolute frequency of the n-gram in a class. This helps weed out extremely infrequent words and spelling errors.
- We require that the ratio of the relative frequency of the n-gram in one class to its relative frequency in the other class is also greater than a threshold. This is a simple means of selecting n-grams indicative of lect.

In experiments based on the bag-of-words model, we only consider an absolute frequency threshold, whereas in later experiments, we also take into account the relative frequency ratio threshold.

3.3 N-gram Binning

In experiments in which we bin n-grams, selected n-grams are assigned to the class in which their relative frequency is highest. For example, an n-gram whose relative frequency in UpSpeak text is twice that in DownSpeak text would be assigned to the class UpSpeak.

N-grams assigned to a class are then partitioned into sets of n-grams. Each of these sets of n-grams is associated with a feature. This partition is based on the n-gram type, the length of n-grams and the relative frequency ratio of the n-grams. While the n-grams composing a set may themselves be indicative of social power lects, this method of grouping them makes no guarantees as to how indicative the overall set is. Therefore, we experimented with filtering out sets which had a negligible information gain. Information gain is an information theoretic concept measuring how much the probability distributions for a feature differ among the different classes. A small information gain suggests that a feature may not be effective at discriminating between classes.

Although this approach to partitioning is simple and worthy of improvement, it effectively reduced the dimensionality of the feature space.

3.4 Classification

Once features are selected, a classifier is trained on these features. Many features are weak on their own; they either occur rarely or occur frequently but only hint weakly at social information. Therefore, we experimented with classifiers friendly to weak features, such as Adaboost and Logistic Regression (MaxEnt). However, we generally achieved the best results using support vector machines, a machine learning classifier which has been successfully applied to many previous text classification problems. We used Weka's optimized SVMs (SMO) (Witten 2005, Platt 1998) and default parameters, except where noted.

4 Evaluation

4.1 Data

To validate our supervised learning approach, we sought an adequately large English corpus of person-to-person communication labeled with the ground truth. For this, we used the publicly avail-

able Enron corpus. After filtering for duplicates and removing empty or otherwise unusable emails, the total number of emails is 245K, containing roughly 90 million words. However, this total includes emails to non-Enron employees, such as family members and employees of other corporations, emails to multiple people, and emails received from Enron employees without a known corporate role. Because the author-recipient relationships of these emails could not be established, they were not included in our experiments.

Building upon previous annotation done on the corpus, we were able to ascertain the corporate role (CEO, Manager, Employee, etc.) of many email authors and recipients. From this information, we determined the author-recipient relationship by applying general rules about the structure of a corporate hierarchy (an email from an Employee to a CEO, for instance, is UpSpeak). This annotation method does not take into account promotions over time, secretaries speaking on behalf of their supervisors, or other causes of relationship irregularities. However, this misinformation would, if anything, generally hurt our classifiers.

The emails were pre-processed to eliminate text not written by the author, such as forwarded text and email headers. As our approach requires text to be POS-tagged, we employed Stanford's POS tagger (http://nlp.stanford.edu/software/tagger.shtml). In addition, text was regularized by conversion to lower case and tokenized to improve counts.

To create training and test sets, we partitioned the authors of text from the corpus into two sets: A and B. Then, we used text authored by individuals in A as a training set and text authored by individuals in B as a test set. The training set is used to determine discriminating features upon which classifiers are built and applied to the test set. We found that partitioning by authors was necessary to avoid artificially inflated scores, because the classifiers pick up aspects of particular authors' language (idiolect) in addition to social power lect information. It was not necessary to account for recipients because the emails did not contain text from the recipients. Table 1 summarizes the text partitions.

	UpSpeak		**DownSpeak**	
	Links	Words	Links	Words
Training	431	136K	328	63K
Test	232	74K	148	27K

Table 1. Author-based Training and Test partitions. The number of author-recipient pairs (links) and the number of words in text labeled as UpSpeak and DownSpeak are shown.

Because preliminary experiments suggested that smaller text samples were harder to classify, the classifiers we describe in this paper were both trained and tested on a subset of the Enron corpus where at least 500 words of text was communicated from a specific author to a specific recipient. This subset contained 142 links, 40% of which were used as the test set.

Weighting for Cost-Sensitive Learning: The original corpus was not balanced: the number of UpSpeak links was greater than the number of DownSpeak links. Varying the weight given to training instances is a technique for creating a classifier that is cost-sensitive, since a classifier built on an unbalanced training set can be biased towards avoiding errors on the overrepresented class (Witten, 2005). We wanted misclassifying UpSpeak as DownSpeak to have the same cost as misclassifying DownSpeak as UpSpeak. To do this, we assigned weights to each instance in the training set. UpSpeak instances were weighted less than DownSpeak instances, creating a training set that was balanced between UpSpeak and DownSpeak. Balancing the training set generally improved results.

Weighting the test set in the same manner allowed us to evaluate the performance of the classifier in a situation in which the numbers of UpSpeak and DownSpeak instances were equal. A baseline classifier that always predicted the majority class would, on its own, achieve an accuracy of 74% on UpSpeak/DownSpeak classification of unweighted test set instances with a minimum length of 500 words. However, results on the weighted test set are properly compared to a baseline of 50%. We include both approaches to scoring in this paper.

4.2 UpSpeak/DownSpeak Classifiers

In this section, we describe experiments on classification of interpersonal email communication into UpSpeak and DownSpeak. For these experiments, only emails exchanged between two people related by a superior/subordinate power relationship were

	Features	# of features	# of n-grams	Cross-Validation		Test Set (weighted)		Test Set (unweighted)	
				Acc (%)	F-score	Acc (%)	F-score	Acc (%)	F-score
(1)	Word unigrams	3899	3899	55.4	.481	62.1	.567	78.9	.748
(2)	Word bigrams	3740	3740	54.5	.457	56.4	.498	73.7	.693
(3)	Word unigrams + word bigrams	7639	7639	51.8	.398	63.3	.576	80.7	.762
(4)	(3) + tag unigrams + tag bigrams	9014	9014	51.8	.398	58.8	.515	77.2	.719
(5)	Binned n-grams	8	106	83.0	.830	**78.1**	**.781**	77.2	.783
(6)	N-grams from (5), separated	106	106	83.0	.828	60.5	.587	70.2	.698
(7)	(5) + polite imperatives	9	108	83.9	.839	77.1	.771	**78.9**	**.797**

Table 2. Experiment Results. Accuracies/F-Scores with an SVM classifier for 10-fold cross validation on the weighted training set and evaluation against the weighted and unweighted test sets. Note that the baseline accuracy against the unweighted test set is 74%, but 50% for the weighted test set and cross-validation.

Human-Engineered Features: Before examining the data itself, we identified some features which we thought would be predictive of UpSpeak or DownSpeak, and which could be fairly accurately modeled by mixed n-grams. These features included the use of different types of imperatives.

We also thought that the type of greeting or signature used in the email might be reflective of formality, and therefore of UpSpeak and DownSpeak. For example, subordinates might be more likely to use an honorific when addressing a superior, or to sign an email with "Thanks." We preformed some preliminary experiments using these features. While the feature set was too small to produce notable results, we identified which features actually were indicative of lect. One such feature was polite imperatives (imperatives preceded by the word "please"). The polite imperative feature was represented by the n-gram set:

{please ^VB, please ^'comma' ^VB}.

Unigrams and Bigrams: As a different sort of baseline, we considered the results of a bag-of-words based classifier. Features used in these experiments consist of single words which occurred a minimum of four times in the relevant lects (UpSpeak and DownSpeak) of the training set. The results of the SVM classifier, shown in line (1) of Table 2, were fairly poor. We then performed experiments with word bigrams, selecting as features those which occurred at least seven times in the relevant lects of the training set. This threshold for bigram frequency minimized the difference in the number of features between the unigram and bigram experiments. While the bigrams on their own were less successful than the unigrams, as seen in line (2), adding them to the unigram features improved accuracy against the test set, shown in line (3).

As we had speculated that including surface-level grammar information in the form of tag n-grams would be beneficial to our problem, we performed experiments using all tag unigrams and all tag bigrams occurring in the training set as features. The results are shown in line (4) of Table 2. The results of these experiments were not particularly strong, likely owing to the increased sparsity of the feature vectors.

Binning: Next, we wished to explore longer n-grams of words or POS tags and to reduce the sparsity of the feature vectors. We therefore experimented with our method of binning the individual n-grams to be used as features. We binned features by their relative frequency ratios. In addition to binning, we also reduced the total number of n-grams by setting higher frequency thresholds and relative frequency ratio thresholds.

When selecting n-grams for this experiment, we considered only word n-grams and tag n-grams – not mixed n-grams, which are a combination of words and tags. These mixed n-grams, while useful for specifying human-defined features, largely increased the dimensionality of the feature search space and did not provide significant benefit in preliminary experiments. For the word sequences,

we set an absolute frequency threshold that depended on class. The frequency of a word n-gram in a particular class was required to be *0.18 * nrlinks / n*, where *nrlinks* is the number of links in each class (431 for UpSpeak and 328 for DownSpeak), and *n* is the number of words in the class. The relative frequency ratio was required to be at least 1.5. The tag sequences were required to meet an absolute frequency threshold of 20, but the same relative frequency ratio of 1.5.

Binning the n-grams into features was done based on both the length of the n-gram and the relative frequency ratio. For example, one feature might represent the set of all word unigrams which have a relative frequency ratio between 1.5 and 1.6.

We explored possible feature sets with cross validation. Before filtering for low information gain, we used six word n-gram bins per class (relative frequency ratios of 1.5, 1.6 ..., 1.9 and 2.0+), one tag n-gram bin for UpSpeak (2.0+), and three tag n-gram bins for DownSpeak (2.0+, 5.0+, 10.0+). Even with the weighted training set, DownSpeak instances were generally harder to identify and likely benefited from additional representation. Grouping features by length was a simple but arbitrary method for reducing dimensionality, yet sometimes produced small bins of otherwise good features. Therefore, as we explored the feature space, small bins of different n-gram lengths were merged. We then employed Weka's InfoGain feature selection tool to remove those features with a low information gain[3], which removed all but eight features. The results of this experiment are shown in line (5) of Table 2. It far outperforms the bag-of-words baselines, despite significantly fewer features.

To ascertain which feature reduction method had the greatest effect on performance – binning or setting a relative frequency ratio threshold – we performed an experiment in which all the n-grams that we used in the previous experiment were their own features. Line (6) of Table 2 shows that while this approach is an improvement over the basic bag-of-words method, grouping features still improves results.

Our goal was to have successful results using only statistically extracted features; however, we examined the effect of augmenting this feature set with the most indicative of the human-identified feature – polite imperatives. The results, in line (7), show a slight improvement in both the cross validation accuracy, and the accuracy against the unweighted test set increases to **78.9%**[4]. However, among the weighted test sets, the highest accuracy was **78.1%**, with the features in line (5).

We report the scores for cross-validation on the training set for these features; however, because the features were selected with knowledge of their per-class distribution in the training set, these cross-validation scores should not be seen as the classifier's true accuracy.

Self-Training: Besides sparse feature vectors, another factor likely to be hurting our classifier was the limited amount of training data. We attempted to increase the training set size by performing exploratory experiments with self-training, an iterative semi-supervised learning method (Zhu, 2005) with the feature set from (7). On the first iteration, we trained the classifier on the labeled training set, classified the instances of the unlabeled test set, and then added the instances of the test set along with their predicted class to the training set to be used for the next iteration. After three iterations, the accuracy of the classifier when evaluated on the weighted test set improved to **82%**, suggesting that our classifiers would benefit from more data.

Impact of Cost-Sensitive Learning: Without cost-sensitive learning, the classifiers were heavily biased towards UpSpeak, tending to classify both DownSpeak and UpSpeak test instances as UpSpeak. With cost-sensitive training, overall performance improved and classifier performance on DownSpeak instances improved dramatically. In (5) of Table 2, DownSpeak classifier accuracy even edged out the accuracy for UpSpeak. We expect that on a larger dataset behavior with unweighted training and test data would improve.

5 Conclusions and Future Research

We presented a corpus-based statistical learning approach to modeling social power relationships and experimental results for our methods. To our

[3] In Weka, features ('attributes') with a sufficiently low information gain have this value rounded down to "0"; these are the features we removed.

[4] The associated p-value is 6.56E-6.

knowledge, this is the first corpus-based approach to learning social power lects beyond those in direct reporting relationships.

Our work strongly suggests that statistically extracted features are an efficient and effective approach to modeling social information. Our methods exploit many aspects of language use and effectively model social power information while using statistical methods at every stage to tease out the information we seek, significantly reducing language-, culture-, and lect-specific engineering needs. Our feature selection method picks up on indicators suggested by sociolinguistics, and it also allows for the identification of features that are not obviously characteristic of UpSpeak or DownSpeak. Some easily recognizable features include:

Lect	Ngram	Example
UpSpeak	if you	"Let me know *if you* need anything."
		"Please call me *if you* have any questions."
DownSpeak	give me	"Read this over and *give me* a call."
		"Please *give me* your comments next week."

On the other hand, other features are less intuitive:

Lect	Ngram	Example
UpSpeak	I'll, we'll	"*I'll* let you know the final results soon"
		"Everyone is very excited […] and we're confident *we'll* be successful"
DownSpeak	that is, this is	"Neither does any other group but *that is* not my problem"
		"I think *this is* an excellent letter"

We hope to improve our methods for selecting and binning features with information theoretic selection metrics and clustering algorithms.

We also have begun work on 3-way, UpSpeak/DownSpeak/PeerSpeak classification. Training a multiclass SVM on the binned n-gram features from (5) produces **51.6%** cross-validation accuracy on training data and **44.4%** accuracy on the weighted test set (both numbers should be compared to a 33% baseline). That classifier contained no n-gram features selected from the PeerSpeak class. Preliminary experiments incorporating PeerSpeak n-grams yield slightly better numbers. However, early results also suggest that the three-way classification problem is made more tractable with cascaded two-way classifiers; feature selection was more manageable with binary problems. For example, one classifier determines whether an instance is UpSpeak; if it is not, a second classifier distinguishes between DownSpeak and PeerSpeak. Our text classification problem is similar to sentiment analysis in that there are class dependencies; for example, DownSpeak is more closely related to PeerSpeak than to UpSpeak. We might attempt to exploit these dependencies in a manner similar to Pang and Lee (2005) to improve three-way classification.

In addition, we had promising early results for classification of author-recipient links with 200 to 500 words, so we plan to explore performance improvements for links of few words.

In early, unpublished work, we had promising results with generative model-based approach to SPM, and we plan to revisit it; language models are a natural fit for lect modeling. Finally, we hope to investigate how SPM and SNA can enhance one another, and explore other lect classification problems for which the ground truth can be found.

Acknowledgments

Dr. Richard Sproat contributed time, valuable insights, and wise counsel on several occasions during the course of the research. Dr. Lillian Lee and her students in *Natural Language Processing and Social Interaction* reviewed the paper, offering valuable feedback and helpful leads.

Our colleague, Diane Bramsen, created an excellent graphical interface for probing and understanding the results. Jeff Lau guided and advised throughout the project.

We thank our anonymous reviewers for prudent advice.

This work was funded by the Army Studies Board and sponsored by Col. Timothy Hill of the United Stated Army Intelligence and Security Command (INSCOM) Futures Directorate under contract W911W4-08-D-0011.

References

Cecilia Ovesdotter Alm, Dan Roth and Richard Sproat. 2005. Emotions from text: machine learning for text-based emotion prediction. *HLT/EMNLP 2005*. October 6-8, 2005, Vancouver.

Penelope Brown and Stephen C. Levinson. 1987. *Politeness: Some universals in language usage*. Cambridge: Cambridge University Press.

Eric Breck, Yejin Choi and Claire Cardie. 2007. Identifying expressions of opinion in context. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-2007)*

CALO Project. 2009. Enron E-Mail Dataset. http://www.cs.cmu.edu/~enron/.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Honolulu, Hawaii: ACM. 793-801.

Yejin Choi and Claire Cardie. 2009. Adapting a polarity lexicon using integer linear programming for domain-specific sentiment classification. *Empirical Methods in Natural Language Processing (EMNLP)*.

Christopher P. Diehl, Galileo Namata, and Lise Getoor. 2007. Relationship identification for social network discovery. *AAAI '07: Proceedings of the 22nd National Conference on Artificial Intelligence.*

Bonnie Erickson, et al. 1978. Speech style and impression formation in a court setting: The effects of 'powerful' and 'powerless' speech. *Journal of Experimental Social Psychology 14*: 266-79.

Norman Fairclough. 1989. *Language and power*. London: Longman.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. *SIGKDD Exploration (1)*: Issue 1.

JHU Center for Imaging Science. 2005. Scan Statistics on Enron Graphs. http://cis.jhu.edu/~parky/Enron/

Soo-min Kim and Eduard Hovy. 2004. Determining the Sentiment of Opinions. *Proceedings of the COLING Conference*. Geneva, Switzerland.

Francois Mairesse and Marilyn Walker. 2006. Automatic recognition of personality in conversation. *Proceedings of HLT-NAACL.* New York City, New York.

Galileo Mark S. Namata Jr., Lise Getoor, and Christopher P. Diehl. 2006. Inferring organizational titles in online communication. *ICML 2006*, 179-181.

Andrew McCallum, Xuerui Wang, and Andres Corrada-Emmanuel. 2007. Topic and role discovery in social networks with experiments on Enron and academic e-Mail. *Journal of Artificial Intelligence Research* 29.

Ryan McDonald, Kerry Hannan, Tyler Neylon, Mike Wells, and Jeff Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. *Proceedings of the ACL.*

David Morand. 2000. Language and power: An empirical analysis of linguistic strategies used in superior/subordinate communication. *Journal of Organizational Behavior*, *21*:235-248.

Frederick Mosteller and David L. Wallace. 1964. *Inference and disputed authorship: The Federalist*. Addison-Wesley, Reading, Mass.

Jon Oberlander and Scott Nowson. 2006. Whose thumb is it anyway? Classifying author personality from weblog text. *Proceedings of CoLing/ACL*. Sydney, Australia.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. *Proceedings of EMNLP*, 79–86.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *Proceedings of the ACL.*

John Platt. 1998. Sequential minimal optimization: A fast algorithm for training support vector machines. In *Technical Report MST-TR-98-14*. Microsoft Research.

Delip Rao and Deepak Ravichandran. 2009. Semi-supervised polarity lexicon induction. *European Chapter of the Association for Computational Linguistics*.

Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. *JASIST 60(3)*: 538-556.

Carol Strapparava and Rada Mihalcea. 2008. Learning to identify emotions in text. *SAC 2008*: 1556-1560

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Semantic Orientations of Words using Spin Model. *Annual Meeting of the Association for Computational Linguistics*.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kauffman.

Xiaojin Zhu. 2005. Semi-supervised learning literature survey. *Technical Report 1530*, Department of Computer Sciences, University of Wisconsin, Madison.

Bootstrapping Coreference Resolution Using Word Associations

Hamidreza Kobdani, Hinrich Schütze, Michael Schiehlen and **Hans Kamp**
Institute for Natural Language Processing
University of Stuttgart
kobdani@ims.uni-stuttgart.de

Abstract

In this paper, we present an unsupervised framework that bootstraps a complete coreference resolution (CoRe) system from *word associations* mined from a large unlabeled corpus. We show that word associations are useful for CoRe – e.g., the strong association between *Obama* and *President* is an indicator of likely coreference. Association information has so far not been used in CoRe because it is sparse and difficult to learn from small labeled corpora. Since unlabeled text is readily available, our unsupervised approach addresses the sparseness problem. In a self-training framework, we train a decision tree on a corpus that is automatically labeled using word associations. We show that this unsupervised system has better CoRe performance than other learning approaches that do not use manually labeled data.

1 Introduction

Coreference resolution (CoRe) is the process of finding markables (noun phrases) referring to the same real world entity or concept. Until recently, most approaches tried to solve the problem by binary classification, where the probability of a pair of markables being coreferent is estimated from labeled data. Alternatively, a model that determines whether a markable is coreferent with a preceding cluster can be used. For both pair-based and cluster-based models, a well established feature model plays an important role. Typical systems use a *rich feature space* based on lexical, syntactic and semantic knowledge. Most commonly used features are described by Soon et al. (2001).

Most existing systems are supervised systems, trained on human-labeled benchmark data sets for English. These systems use linguistic features based on number, gender, person etc. It is a challenge to adapt these systems to new domains, genres and languages because a significant human labeling effort is usually necessary to get good performance.

To address this challenge, we pursue an *unsupervised self-training approach.* We train a classifier on a corpus that is automatically labeled using association information. Self-training approaches usually include the use of some manually labeled data. In contrast, our self-trained system is not trained on any manually labeled data and is therefore a completely unsupervised system. Although training on automatically labeled data can be viewed as a form of supervision, we reserve the term *supervised system* for systems that are trained on manually labeled data.

The key novelty of our approach is that we bootstrap a competitive CoRe system from association information that is mined from an unlabeled corpus in a completely unsupervised fashion. While this method is shallow, it provides valuable information for CoRe because it considers the actual identity of the words in question. Consider the pair of markables *(Obama, President)*. It is a likely coreference pair, but this information is not accessible to standard CoRe systems because they only use string-based features (often called lexical features), named entity features and semantic word class features (e.g., from WordNet) that do not distinguish,

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 783–792,
Portland, Oregon, June 19-24, 2011.

say, *Obama* from *Hawking*.

In our approach, word association information is used for *clustering markables* in unsupervised learning. Association information is calculated as *association scores between heads of markables* as described below. We view association information as an example of a *shallow feature space* which contrasts with the rich feature space that is generally used in CoRe.

Our experiments are conducted using the MCORE system ("Modular COreference REsolution").[1] MCORE can operate in three different settings: unsupervised (subsystem *A-INF*), supervised (subsystem *SUCRE* (Kobdani and Schütze, 2010)), and self-trained (subsystem *UNSEL*). The unsupervised subsystem A-INF ("Association INFormation") uses the association scores between heads as the distance measure when clustering markables. SUCRE ("SUpervised Coreference REsolution") is trained on a labeled corpus (manually or automatically labeled) similar to standard CoRe systems. Finally, the unsupervised self-trained subsystem UNSEL ("UNsupervised SELf-trained") uses the unsupervised subsystem A-INF to automatically label an unlabeled corpus that is then used as a training set for SUCRE.

Our main contributions in this paper are as follows:

1. We demonstrate that word association information can be used to develop an unsupervised model for shallow coreference resolution (subsystem A-INF).

2. We introduce an unsupervised self-trained method (UNSEL) that takes a two-learner two-feature-space approach. The two learners are A-INF and SUCRE. The feature spaces are the shallow and rich feature spaces.

3. We show that the performance of UNSEL is better than the performance of other unsupervised systems when it is self-trained on the automatically labeled corpus and uses the leveraging effect of a rich feature space.

4. MCORE is a flexible and modular framework that is able to learn from data with different quality and domain. Not only is it able to deal with shallow information spaces (A-INF), but it can also deliver competitive results for rich feature spaces (SUCRE and UNSEL).

This paper is organized as follows. Related work is discussed in Section 2. In Section 3, we present our system architecture. Section 4 describes the experiments and Section 5 presents and discusses our results. The final section presents our conclusions.

2 Related Work

There are three main approaches to CoRe: supervised, semi-supervised (or weakly supervised) and unsupervised. We use the term *semi-supervised* for approaches that use some amount of human-labeled coreference pairs.

Müller et al. (2002) used co-training for coreference resolution, a semi-supervised method. Co-training puts features into disjoint subsets when learning from labeled and unlabeled data and tries to leverage this split for better performance. Ng and Cardie (2003) use self-training in a multiple-learner framework and report performance superior to co-training. They argue that the multiple learner approach is a better choice for CoRe than the multiple view approach of co-training. Our self-trained model combines multiple learners (A-INF and SUCRE) and multiple views (shallow/rich information). A key difference to the work by Müller et al. (2002) and Ng and Cardie (2003) is that we do not use any human-labeled coreference pairs.

Our basic idea of self-training without human labels is similar to (Kehler et al., 2004), but we address the general CoRe problem, not just pronoun interpretation.

Turning to unsupervised CoRe, Haghighi and Klein (2007) proposed a generative Bayesian model with good performance. Poon and Domingos (2008) introduced an unsupervised system in the framework of Markov logic. Ng (2008) presented a generative model that views coreference as an EM clustering process. We will show that our system, which is simpler than prior work, outperforms these systems.

Haghighi and Klein (2010) present an "almost unsupervised" CoRe system. In this paper, we only compare with completely unsupervised approaches,

[1] MCORE can be downloaded from `ifnlp.org/~schuetze/mcore`.

not with approaches that make some limited use of labeled data.

Recent work by Haghighi and Klein (2009), Klenner and Ailloud (2009) and Raghunathan et al. (2010) challenges the appropriateness of machine learning methods for CoRe. These researchers show that a "deterministic" system (essentially a rule-based system) that uses a rich feature space including lexical, syntactic and semantic features can improve CoRe performance. Almost all CoRe systems, including ours, use a limited number of rules or filters, e.g., to implement binding condition A that reflexives must have a close antecedent in some sense of "close". In our view, systems that use a few basic filters are fundamentally different from carefully tuned systems with a large number of complex rules, some of which use specific lexical information. A limitation of complex rule-based systems is that they require substantial effort to encode the large number of deterministic constraints that guarantee good performance. Moreover, these systems are not adaptable (since they are not machine-learned) and may have to be rewritten for each new domain, genre and language. Consequently, we do not compare our performance with deterministic systems.

Ponzetto (2010) extracts metadata from Wikipedia for supervised CoRe. Using such additional resources in our unsupervised system should further improve CoRe performance. Elsner et al. (2009) present an unsupervised algorithm for identifying clusters of entities that belong to the same named entity (NE) class. Determining common membership in an NE class like person is an easier task than determining coreference of two NEs.

3 System Architecture

Figure 1 illustrates the system architecture of our unsupervised self-trained CoRe system (UNSEL). Oval nodes are data, box nodes are processes. We take a self-training approach to coreference resolution: We first label the corpus using the unsupervised model A-INF and then train the supervised model SUCRE on this automatically labeled training corpus. Even though we train on a labeled corpus, the labeling of the corpus is produced in a completely automatic fashion, without recourse to human labeling. Thus, it is an unsupervised approach.

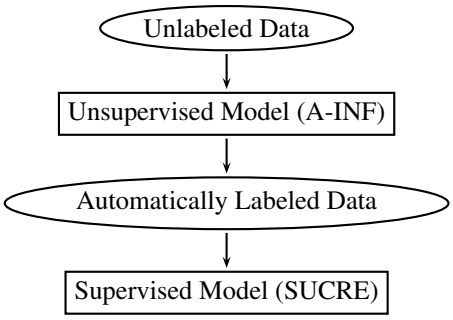


Figure 1: System Architecture of UNSEL (Unsupervised Self-Trained Model).

The MCORE architecture is very flexible; in particular, as will be explained presently, it can be easily adapted for supervised as well as unsupervised settings.

The unsupervised and supervised models have an identical top level architecture; we illustrate this in Figure 2. In *preprocessing*, tokens (words), markables and their attributes are extracted from the input text. The key difference between the unsupervised and supervised approaches is in how *pair estimation* is accomplished — see Sections 3.1 & 3.2 for details.

The main task in *chain estimation* is clustering. Figure 3 presents our clustering method, which is used for both supervised and unsupervised CoRe. We search for the best predicted antecedent (with coreference probability $p \geq 0.5$) from right to left starting from the end of the document. McEnery et al. (1997) showed that in 98.68% of cases the antecedent is within a 10-sentence window; hence we use a window of 10 sentences for search. We have found that limiting the search to a window increases both efficiency and effectiveness.

Filtering. We use a feature definition language to define the templates according to which the filters and features are calculated. These templates are hard constraints that filter out all cases that are clearly disreferent, e.g., *(he, she)* or *(he, they)*. We use the following filters: (i) the antecedent of a reflexive pronoun must be in the same sentence; (ii) the antecedent of a pronoun must occur at a distance of at most 3 sentences; (iii) a coreferent pair of a noun and a pronoun or of two pronouns must not

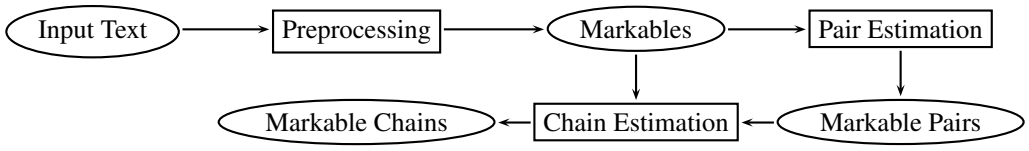


Figure 2: Common architecture of unsupervised (A-INF) and supervised (SUCRE) models.

Chain_Estimation $(M_1, M_2, \ldots, M_n)$
1. $t \leftarrow 1$
2. For each markable M_i: $C_i \leftarrow \{M_i\}$
3. Proceed through the markables from the end of the document. For each M_j, consider each preceding M_i within 10 sentences:
If Pair_Estimation(M_i, M_j)>=t: $C_i \leftarrow C_i \cup C_j$
4. $t \leftarrow t - 0.01$
5. If $t >= 0.5$: go to step 3

Pair_Estimation (M_i, M_j):
If Filtering(M_i, M_j)==FALSE then return 0; else return the probability p (or association score N) of markable pair (M_i, M_j) being coreferent.

Filtering (M_i, M_j):
return TRUE if all filters for (M_i, M_j) are TRUE else FALSE

Figure 3: MCORE chain estimation (clustering) algorithm (test). t is the clustering threshold. C_i refers to the cluster that M_i is a member of.

disagree in number; (iv) a coreferent pair of two pronouns must not disagree in gender. These four filters are used in supervised and unsupervised modes of MCORE.

3.1 Unsupervised Model (A-INF)

Figure 4 (top) shows how A-INF performs pair estimation. First, in the pair generation step, all possible pairs inside 10 sentences are generated. Other steps are separately explained for train and test as follows.

Train. In addition to the filters (i)–(iv) described above, we use the following filter: *(v) If the head of markable M_2 matches the head of the preceding markable M_1, then we ignore all other pairs for M_2 in the calculation of association scores.*

This additional filter is necessary because an approach without some kind of string matching constraint yields poor results, given the importance of string matching for CoRe. As we will show below, even the simple filters (i)–(v) are sufficient to learn high-quality association scores; this means that we do not need the complex features of "deterministic" systems. However, if such complex features are available, then we can use them to improve performance in our self-trained setting.

To learn word association information from an unlabeled corpus (see Section 4), we compute mutual information (MI) scores between heads of markables. We define MI as follows: (Cover and Thomas, 1991)

$$\text{MI}(a,b) = \sum_{i \in \{\bar{a}, a\}} \sum_{j \in \{\bar{b}, b\}} P(i,j) \log_2 \frac{P(i,j)}{P(i)P(j)}$$

E.g., $P(a, \bar{b})$ is the probability of a pair whose two elements are a and a word not equal to b.

Test. A key virtue of our approach is that in the classification of pairs as coreferent/disreferent, the coreference probability p estimated in supervised learning plays exactly the same role as the association information score N (defined below). For p, it is important that we only consider pairs with $p \geq 0.5$ as potentially coreferent (see Figure 3). To be able to impose the same constraint on N, we normalize the MI scores by the maximum values of the two words and take the average:

$$N(a,b) = \frac{1}{2}\left(\frac{\text{MI}(a,b)}{\text{argmax}_x \text{MI}(a,x)} + \frac{\text{MI}(a,b)}{\text{argmax}_x \text{MI}(x,b)}\right)$$

In the above equation, the value of N indicates how strongly two words are associated. N is normalized to ensure $0 \leq N \leq 1$. If a or b did not occur, then we set $N = 0$.

In filtering for test, we use filters (i)–(iv). We then fetch the MI values and calculate N values. The clustering algorithm described in Figure 3 uses these N values in exactly the same way as p: we search for the antecedent with the maximum association score

N greater than 0.5 from right to left starting from the end of the document.

As we will see below, using N scores acquired from an unlabeled corpus as the only source of information for CoRe performs surprising well. However, the weaknesses of this approach are (i) the failure to cover pairs that do not occur in the unlabeled corpus (negatively affecting recall) and (ii) the generation of pairs that are not plausible candidates for coreference (negatively affecting precision). To address these problems, we train a model on a corpus labeled by A-INF in a self-training approach.

3.2 Supervised Model (SUCRE)

Figure 4 (bottom) presents the architecture of pair estimation for the supervised approach (SUCRE).

In the pair generation step for train, we take each coreferent markable pair (M_i, M_j) without intervening coreferent markables and use (M_i, M_j) as a positive training instance and (M_i, M_k), $i < k < j$, as negative training instances. For test, we generate all possible pairs within 10 sentences. After filtering, we then calculate a feature vector for each generated pair that survived filters (i)–(iv).

Our basic features are similar to those described by Soon et al. (2001): string-based features, distance features, span features, part-of-speech features, grammatical features, semantic features, and agreement features. These basic features are engineered with the goal of creating a feature set that will result in good performance. For this purpose we used the relational feature engineering framework which has been presented in (Kobdani et al., 2010). It includes powerful and flexible methods for implementing and extracting new features. It allows systematic and fast search of the space of features and thereby reduces the time and effort needed for defining optimal features. We believe that the good performance of our supervised system SUCRE (tables 1 and 2) is the result of our feature engineering approach.[2]

As our classification method, we use a decision tree[3] (Quinlan, 1993) that is trained on the training set to estimate the coreference probability p for a pair and then applied to the test set. Note that, as is standard in CoRe, filtering and feature calculation are exactly the same for training and test, but that pair generation is different as described above.

4 Experimental Setup

4.1 Data Sets

For computing word association, we used a corpus of about 63,000 documents from the 2009 English *Wikipedia* (the articles that were larger than 200 bytes). This corpus consists of more than 33.8 million tokens; the average document length is 500 tokens. The corpus was parsed using the Berkeley parser (Petrov and Klein, 2007). We ignored all sentences that had no parse output. The number of detected markables (all noun phrases extracted from parse trees) is about 9 million.

We evaluate unsupervised, supervised and self-trained models on *ACE (Phase 2)* (Mitchell et al., 2003).[4] This data set is one of the most widely used CoRe benchmarks and was used by the systems that are most comparable to our approach; in particular, it was used in most prior work on unsupervised CoRe. The corpus is composed of three data sets from three different news sources. We give the number of test documents for each: (i) Broadcast News (BNEWS): 51. (ii) Newspaper (NPAPER): 17. (iii) Newswire (NWIRE): 29. We report results for *true markables* (markables extracted from the answer keys) to be able to compare with other systems that use true markables.

In addition, we use the recently published *OntoNotes* benchmark (Recasens et al., 2010). OntoNotes is an excerpt of news from the OntoNotes Corpus Release 2.0 (Pradhan et al., 2007). The advantage of OntoNotes is that it contains two parallel annotations: (i) a *gold setting*, gold standard manual annotations of the preprocessing information and (ii) an *automatic setting*, automatically predicted annotations of the preprocessing information. The automatic setting reflects the situation a CoRe system

[2] While this is not the focus of this paper, SUCRE has performance comparable to other state-of-the-art supervised systems. E.g., B^3/MUC F_1 are 75.6/72.4 on ACE-2 and 69.4/70.6 on MUC-6 compared to 78.3/66.0 on ACE-2 and 70.9/68.5 on MUC-6 for Reconcile (Stoyanov et al., 2010)

[3] We also tried support vector machines and maximum entropy models, but they did not perform better.

[4] We used two variants of ACE (Phase 2): ACE-2 and ACE2003

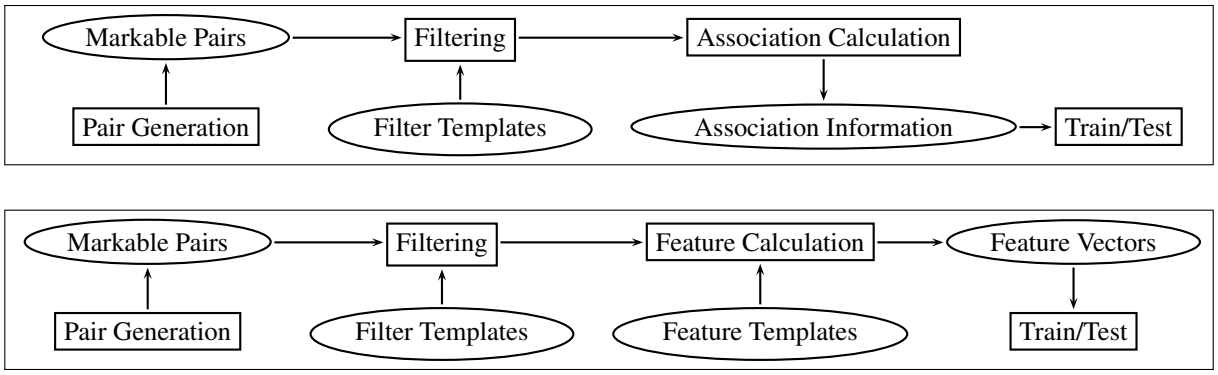


Figure 4: Pair estimation in the unsupervised model A-INF (top) and in the supervised model SUCRE (bottom).

faces in reality; in contrast, the gold setting should be considered less realistic.

The issue of gold vs. automatic setting is directly related to a second important evaluation issue: the influence of markable detection on CoRe evaluation measures. In a real application, we do not have access to true markables, so an evaluation on *system markables* (markables automatically detected by the system) reflects actual expected performance better. However, reporting only CoRe numbers (even for system markables) is not sufficient either since accuracy of markable detection is necessary to interpret CoRe scores. Thus, we need (i) measures of the quality of system markables (i.e., an evaluation of the markable detection subtask) and CoRe performance on system markables as well as (ii) a measure of CoRe performance on true markables. We use OntoNotes in this paper to perform such a, in our view, complete and realistic evaluation of CoRe. The two evaluations correspond to the two evaluations performed at SemEval-2010 (Recasens et al., 2010): the automatic setting with system markables and the gold setting with true markables. Test set size is 85 documents.

In the experiments with A-INF we use Wikipedia to compute association information and then evaluate the model on the test sets of ACE and OntoNotes. For the experiments with UNSEL, we use its unsupervised subsystem A-INF (which uses Wikipedia association scores) to automatically label the training sets of ACE and OntoNotes. Then for each data set, the supervised subsystem of UNSEL (i.e., SUCRE) is trained on its automatically labeled training set and then evaluated on its test set. Finally, for the supervised experiments, we use the manually labeled training sets and evaluate on the corresponding test sets.

4.2 Evaluation Metrics

We report recall, precision, and F_1 for MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), and CEAF (Luo, 2005). We selected these three metrics because a single metric is often misleading and because we need to use metrics that were used in previous unsupervised work.

It is well known that MUC by itself is insufficient because it gives misleadingly high scores to the "single-chain" system that puts all markables into one chain (Luo et al., 2004; Finkel and Manning, 2008). However, B^3 and CEAF have a different bias: they give high scores to the "all-singletons" system that puts each markable in a separate chain. On OntoNotes test, we get $B^3 = 83.2$ and CEAF $= 71.2$ for all-singletons, which incorrectly suggests that performance is good; but MUC F_1 is 0 in this case, demonstrating that all-singletons performs poorly. With the goal of performing a complete evaluation, one that punishes all-singletons as well as single-chain, we use one of the following two combinations: (i) MUC and B^3 or (ii) MUC and CEAF. Recasens et al. (2010) showed that B^3 and CEAF are highly correlated (Pearson's $r = 0.91$). Therefore, either combination (i) or combination (ii) fairly characterizes CoRe performance.

5 Results and Discussion

Table 1 compares our unsupervised self-trained model UNSEL and unsupervised model A-INF to

		MUC			B^3			CEAF		
	BNEWS-ACE-2	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
1	P&D	68.3	66.6	67.4	70.3	65.3	67.7	–	–	–
2	A-INF	60.8	61.4	61.1	55.5	69.0	61.5	52.6	52.0	52.3
3	UNSEL	72.5	65.6	68.9	72.5	66.4	69.3	56.7	64.8	60.5
4	SUCRE	86.6	60.3	71.0	87.6	64.6	74.4	56.1	81.6	66.5
	NWIRE-ACE-2	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
5	P&D	67.7	67.3	67.4	74.7	68.8	71.6	–	–	–
6	A-INF	62.4	57.4	59.8	59.2	62.4	60.7	46.8	52.5	49.5
7	UNSEL	76.2	61.5	68.1	81.5	67.6	73.9	61.5	77.1	68.4
8	SUCRE	82.5	65.7	73.1	85.4	72.3	78.3	63.5	80.6	71.0
	NPAPER-ACE-2	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
9	P&D	69.2	71.7	70.4	70.0	66.5	68.2	–	–	–
10	A-INF	60.6	56.0	58.2	52.4	60.3	56.0	38.9	44.0	41.3
11	UNSEL	78.6	65.7	71.6	74.0	68.0	70.9	57.6	73.2	64.5
12	SUCRE	82.5	67.0	73.9	80.7	69.5	74.6	58.8	77.1	66.7
	BNEWS-ACE2003	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
13	H&K	68.3	56.8	62.0	–	–	–	59.9	53.9	56.7
14	Ng	71.4	56.1	62.8	–	–	–	60.5	53.3	56.7
15	A-INF	60.9	64.9	62.8	50.9	72.5	59.8	53.8	49.4	51.5
16	UNSEL	69.5	65.0	67.1	70.2	65.9	68.0	58.5	64.2	61.2
17	SUCRE	73.9	68.5	71.1	75.4	69.6	72.4	60.1	66.6	63.2
	NWIRE-ACE2003	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
18	H&K	66.2	46.8	54.8	–	–	–	62.8	49.6	55.4
19	Ng	68.3	47.0	55.7	–	–	–	60.7	49.2	54.4
20	A-INF	62.7	60.5	61.6	54.8	66.1	59.9	47.7	50.2	49.0
21	UNSEL	64.8	68.6	66.6	61.5	73.6	67.0	59.8	55.1	57.3
22	SUCRE	77.6	69.3	73.2	78.8	75.2	76.9	65.1	74.4	69.5

Table 1: Scores for MCORE (A-INF, SUCRE and UNSEL) and three comparable systems on ACE-2 and ACE2003.

P&D (Poon and Domingos, 2008) on ACE-2; and to Ng (Ng, 2008) and H&K[5] (Haghighi and Klein, 2007) on ACE2003. To our knowledge, these three papers are the best and most recent evaluation results for unsupervised learning and they all report results on ACE-2 and ACE-2003. Results on SUCRE will be discussed later in this section.

A-INF scores are below some of the earlier unsupervised work reported in the literature (lines 2, 6, 10) although they are close to competitive on two of the datasets (lines 15 and 20: MUC scores are equal or better, CEAF scores are worse). Given the simplicity of A-INF, which uses nothing but associations mined from a large unannotated corpus, its performance is surprisingly good.

Turning to UNSEL, we see that F_1 is always better for UNSEL than for A-INF, for all three measures (lines 3 vs 2, 7 vs 6, 11 vs 10, 16 vs 15, 21 vs 20). This demonstrates that the self-training step of UNSEL is able to correct many of the errors that A-INF commits. Both precision and recall are improved with two exceptions: recall of B^3 decreases from line 2 to 3 and from 15 to 16.

When comparing the unsupervised system UNSEL to previous unsupervised results, we find that UNSEL's F_1 is higher in all runs (lines 3 vs 1, 7 vs 5, 11 vs 9, 16 vs 13&14, 21 vs 18&19). The differences are large (up to 11%) compared to H&K and

[5]We report numbers for the better performing Pronoun-only Salience variant of H&K proposed by Ng (2008).

Ng. The difference to P&D is smaller, ranging from 2.7% (B^3, lines 11 vs 9) to 0.7% (MUC, lines 7 vs 5). Given that MCORE is a simpler and more efficient system than this prior work on unsupervised CoRe, these results are promising.

In contrast to F_1, there is no consistent trend for precision and recall. For example, P&D is better than UNSEL on MUC recall for BNEWS-ACE-2 (lines 1 vs 3) and H&K is better than UNSEL on CEAF precision for NWIRE-ACE2003 (lines 18 vs 21). But this higher variability for precision and recall is to be expected since every system trades the two measures off differently.

These results show that the application of self-training significantly improves performance. As discussed in Section 3.1, self-training has positive effects on both recall and precision. We now present two simplified examples that illustrate this point.

Example for recall. Consider the markable pair (*Novoselov*[6]*,he*) in the test set. Its N score is 0 because our subset of 2009 Wikipedia sentences has no occurrence of *Novoselov.* However, A-INF finds many similar pairs like (*Einstein,he*) and (*Hawking,he*), pairs that have high N scores. Suppose we represent pairs using the following five features: <sentence distance, string match, type of first markable, type of second markable, number agreement>. Then (*Einstein,he*), (*Hawking,he*) and (*Novoselov,he*) will all be assigned the feature vector <1, No, Proper Noun, Personal Pronoun, Yes>. We can now automatically label Wikipedia using A-INF – this will label (*Einstein,he*) and (*Hawking,he*) as coreferent – and train SUCRE on the resulting training set. SUCRE can then resolve the coreference (*Novoselov,he*) correctly. We call this the *better recall effect*.

Example for precision. Using the same representation of pairs, suppose that for the sequence of markables *Biden, Obama, President* the markable pairs (*Biden,President*) and (*Obama,President*) are assigned the feature vectors <8, No, Proper Noun, Proper Noun, Yes> and <1, No, Proper Noun, Proper Noun, Yes>, respectively. Since both pairs have N scores > 0.5, A-INF incorrectly puts the three markables into one cluster. But as we would expect, A-INF labels many more markable pairs with the second feature vector (distance=1) as coreferent than with the first one (distance=8) in the entire automatically labeled training set. If we now train SUCRE on this training set, it can resolve such cases in the test set correctly even though they are so similar: (*Biden,President*) is classified as disreferent and (*Obama,President*) as coreferent. We call this the *better precision effect*.

[6]The 2010 physics Nobel laureate.

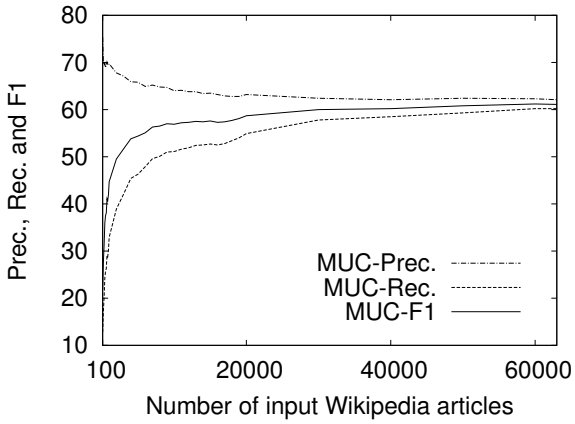


Figure 5: MUC learning curve for A-INF.

Recall that UNSEL has better recall and precision than A-INF in almost all cases (discussion of Table 1). This result shows that better precision and better recall effects do indeed benefit UNSEL.

To summarize, the advantages of our self-training approach are: (i) We cover cases that do not occur in the unlabeled corpus (better recall effect); and (ii) we use the leveraging effect of a rich feature space including distance, person, number, gender etc. to improve precision (better precision effect).

Learning curve. Figure 5 presents MUC scores of A-INF as a function of the number of Wikipedia articles used in unsupervised learning. We can see that a small number of input articles (e.g., 100) results in low recall and high precision. When we increase the number of input articles, recall rapidly increases and precision rapidly decreases up to about 10,000 articles. Increase and decrease continue more slowly after that. F_1 increases throughout because lower precision is compensated by higher recall. This learning curve demonstrates the importance of the size of the corpus for A-INF.

Comparison of UNSEL with SUCRE

Table 2 compares our unsupervised self-trained (UNSEL) and supervised (SUCRE) models with the recently published SemEval-2010 OntoNotes re-

Gold setting + True markables				
System	MD	MUC	B^3	CEAF
Relax	100	33.7	84.5	75.6
$SUCRE_{2010}$	100	60.8	82.4	74.3
SUCRE	100	64.3	87.0	80.1
UNSEL	100	63.0	86.9	79.7
Automatic setting + System markables				
System	MD	MUC	B^3	CEAF
$SUCRE_{2010}$	80.7	52.5	67.1	62.7
Tanl-1	73.9	24.6	61.3	57.3
SUCRE	80.9	55.7	69.7	66.6
UNSEL	80.9	55.0	69.8	66.3

Table 2: F_1 scores for MCORE (SUCRE and UNSEL) and the best comparable systems in SemEval-2010. MD: Markable Detection F_1 (Recasens et al., 2010).

sults (gold and automatic settings). We compare with the scores of the two best systems, Relax and $SUCRE_{2010}$[7] (for the gold setting with true markables) and $SUCRE_{2010}$ and Tanl-1 (for the automatic setting with system markables, 89.9% markable detection (MD) F_1). It is apparent from this table that our supervised and unsupervised self-trained models outperform Relax, $SUCRE_{2010}$ and Tanl-1. We should make clear that we did not use the test set for development to ensure a fair comparison with the participant systems at SemEval-2010.

Table 1 shows that the unsupervised self-trained system (UNSEL) does a lot worse than the supervised system (SUCRE) on ACE.[8] In contrast, UNSEL performs almost as well as SUCRE on OntoNotes (Table 2), for both gold and automatic settings: F_1 differences range from $+.1$ (Automatic, B^3) to -1.3 (Gold, MUC). We suspect that this is partly due to the much higher proportion of singletons in OntoNotes than in ACE-2: 85.2% (OntoNotes) vs. 60.2% (ACE-2). The low recall of the automatic labeling by A-INF introduces a bias for singletons when UNSEL is self-trained. Another reason is that the OntoNotes training set is about 4 times larger than each of BNEWS, NWIRE and NPAPER training sets. With more training data, UNSEL can correct more of its precision and recall errors. For an unsupervised approach, which only needs unlabeled data, there is little cost to creating large training sets. Thus, this comparison of ACE-2/Ontonotes results is evidence that in a realistic scenario using association information in an unsupervised self-trained system is almost as good as a system trained on manually labeled data.

It is important to note that the comparison of SUCRE to UNSEL is the most direct comparison of supervised and unsupervised CoRe learning we are aware of. The two systems are identical with the single exception that they are trained on manual vs. automatic coreference labels.

6 Conclusion

In this paper, we have demonstrated the utility of association information for coreference resolution. We first developed a simple unsupervised model for shallow CoRe that only uses association information for finding coreference chains. We then introduced an unsupervised self-trained approach where a supervised model is trained on a corpus that was automatically labeled by the unsupervised model based on the association information. The results of the experiments indicate that the performance of the unsupervised self-trained approach is better than the performance of other unsupervised learning systems. In addition, we showed that our system is a flexible and modular framework that is able to learn from data with different quality (perfect vs noisy markable detection) and domain; and is able to deliver good results for shallow information spaces and competitive results for rich feature spaces. Finally, our framework is the first CoRe system that is designed to support three major modes of machine learning equally well: supervised, self-trained and unsupervised.

Acknowledgments

This research was funded by DFG (grant SCHU 2246/4).

We thank Aoife Cahill, Alexander Fraser, Thomas Müller and the anonymous reviewers for their helpful comments.

[7] It is the first version of our supervised system that took part in SemEval-2010. We call it $SUCRE_{2010}$.

[8] A reviewer observes that SUCRE's performance is better than the supervised system of Ng (2008). This may indicate that part of our improved unsupervised performance in Table 1 is due to better feature engineering implemented in SUCRE.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *LREC Workshop on Linguistics Coreference '98*, pages 563–566.

Thomas M. Cover and Joy A. Thomas. 1991. *Elements of Information Theory*. Wiley.

Micha Elsner, Eugene Charniak, and Mark Johnson. 2009. Structured generative models for unsupervised named-entity clustering. In *HLT-NAACL '09*, pages 164–172.

Jenny Rose Finkel and Christopher D. Manning. 2008. Enforcing transitivity in coreference resolution. In *HLT '08*, pages 45–48.

Aria Haghighi and Dan Klein. 2007. Unsupervised coreference resolution in a nonparametric bayesian model. In *ACL '07*, pages 848–855.

Aria Haghighi and Dan Klein. 2009. Simple coreference resolution with rich syntactic and semantic features. In *EMNLP '09*, pages 1152–1161.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *NAACL-HLT '10*, pages 385–393.

Andrew Kehler, Douglas E. Appelt, Lara Taylor, and Aleksandr Simma. 2004. Competitive Self-Trained Pronoun Interpretation. In *HLT-NAACL '04*, pages 33–36.

Manfred Klenner and Étienne Ailloud. 2009. Optimization in coreference resolution is not needed: A nearly-optimal algorithm with intensional constraints. In *EACL*, pages 442–450.

Hamidreza Kobdani and Hinrich Schütze. 2010. Sucre: A modular system for coreference resolution. In *SemEval '10*, pages 92–95.

Hamidreza Kobdani, Hinrich Schütze, Andre Burkovski, Wiltrud Kessler, and Gunther Heidemann. 2010. Relational feature engineering of natural language processing. In *CIKM '10*. ACM.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A Mention-Synchronous Coreference Resolution Algorithm Based on the Bell Tree. In *ACL '04*, pages 135–142.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *HLT '05*, pages 25–32.

A. McEnery, I. Tanaka, and S. Botley. 1997. Corpus annotation and reference resolution. In *ANARESOLUTION '97*, pages 67–74.

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

Christoph Müller, Stefan Rapp, and Michael Strube. 2002. Applying co-training to reference resolution. In *ACL '02*, pages 352–359.

Vincent Ng and Claire Cardie. 2003. Bootstrapping coreference classifiers with multiple machine learning algorithms. In *EMNLP '03*, pages 113–120.

Vincent Ng. 2008. Unsupervised models for coreference resolution. In *EMNLP '08*, pages 640–649.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL '07*, pages 404–411.

Simone Paolo Ponzetto. 2010. *Knowledge Acquisition from a Collaboratively Generated Encyclopedia*, volume 327 of *Dissertations in Artificial Intelligence*. Amsterdam, The Netherlands: IOS Press & Heidelberg, Germany: AKA Verlag.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with markov logic. In *EMNLP '08*, pages 650–659.

Sameer S. Pradhan, Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2007. Ontonotes: A unified relational semantic representation. In *ICSC '07*, pages 517–526.

J. Ross Quinlan. 1993. *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *EMNLP '10*, pages 492–501.

Marta Recasens, Lluís Màrquez, Emili Sapena, M.Antònia Martí, Mariona Taulé, Véronique Hoste, Massimo Poesio, and Yannick Versley. 2010. SemEval-2010 Task 1: Coreference resolution in multiple languages. In *SemEval '10*, pages 70–75.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. In *CL '01*, pages 521–544.

Veselin Stoyanov, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom. 2010. Coreference resolution with reconcile. In *ACL '10*, pages 156–161.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *MUC6 '95*, pages 45–52.

Large-Scale Cross-Document Coreference Using Distributed Inference and Hierarchical Models

Sameer Singh[§] **Amarnag Subramanya**[†] **Fernando Pereira**[†] **Andrew McCallum**[§]
[§] Department of Computer Science, University of Massachusetts, Amherst MA 01002
[†] Google Research, Mountain View CA 94043
sameer@cs.umass.edu, asubram@google.com, pereira@google.com, mccallum@cs.umass.edu

Abstract

Cross-document coreference, the task of grouping all the mentions of each entity in a document collection, arises in information extraction and automated knowledge base construction. For large collections, it is clearly impractical to consider all possible groupings of mentions into distinct entities. To solve the problem we propose two ideas: (a) a *distributed* inference technique that uses parallelism to enable large scale processing, and (b) a *hierarchical* model of coreference that represents uncertainty over multiple granularities of entities to facilitate more effective approximate inference. To evaluate these ideas, we constructed a labeled corpus of 1.5 million disambiguated mentions in Web pages by selecting link anchors referring to Wikipedia entities. We show that the combination of the hierarchical model with distributed inference quickly obtains high accuracy (with error reduction of 38%) on this large dataset, demonstrating the scalability of our approach.

1 Introduction

Given a collection of mentions of entities extracted from a body of text, *coreference* or *entity resolution* consists of clustering the mentions such that two mentions belong to the same cluster if and only if they refer to the same entity. Solutions to this problem are important in semantic analysis and knowledge discovery tasks (Blume, 2005; Mayfield et al., 2009). While significant progress has been made in *within*-document coreference (Ng, 2005; Culotta et al., 2007; Haghighi and Klein, 2007; Bengston and Roth, 2008; Haghighi and Klein, 2009; Haghighi and Klein, 2010), the larger problem of *cross*-document coreference has not received as much attention.

Unlike inference in other language processing tasks that scales linearly in the size of the corpus, the hypothesis space for coreference grows super-exponentially with the number of mentions. Consequently, most of the current approaches are developed on small datasets containing a few thousand mentions. We believe that cross-document coreference resolution is most useful when applied to a very large set of documents, such as all the news articles published during the last 20 years. Such a corpus would have billions of mentions. In this paper we propose a model and inference algorithms that can scale the cross-document coreference problem to corpora of that size.

Much of the previous work in cross-document coreference (Bagga and Baldwin, 1998; Ravin and Kazi, 1999; Gooi and Allan, 2004; Pedersen et al., 2006; Rao et al., 2010) groups mentions into entities with some form of greedy clustering using a pairwise mention similarity or distance function based on mention text, context, and document-level statistics. Such methods have not been shown to scale up, and they cannot exploit cluster features that cannot be expressed in terms of mention pairs. We provide a detailed survey of related work in Section 6.

Other previous work attempts to address some of the above concerns by mapping coreference to inference on an undirected graphical model (Culotta et al., 2007; Poon et al., 2008; Wellner et al., 2004; Wick et al., 2009a). These models contain pairwise factors between all pairs of mentions capturing similarity between them. Many of these models also enforce transitivity and enable features over

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 793–803,
Portland, Oregon, June 19-24, 2011.

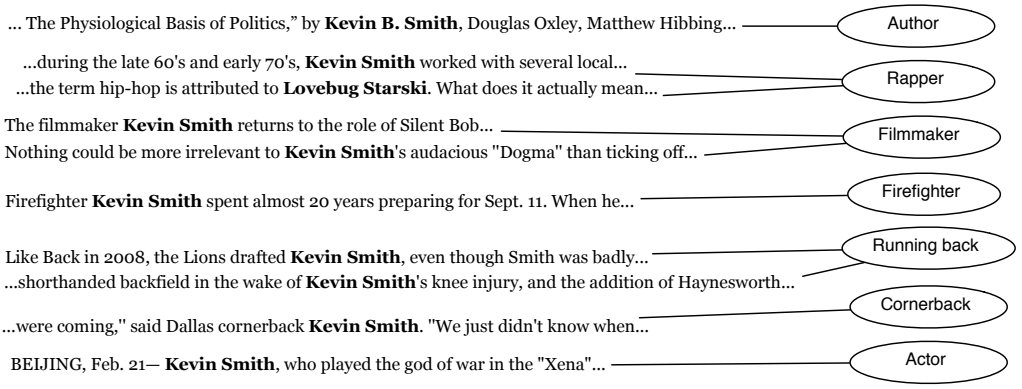


Figure 1: **Cross-Document Coreference Problem:** Example mentions of "Kevin Smith" from New York Times articles, with the true entities shown on the right.

entities by including *set-valued variables*. Exact inference in these models is intractable and a number of approximate inference schemes (McCallum et al., 2009; Rush et al., 2010; Martins et al., 2010) may be used. In particular, *Markov chain Monte Carlo* (MCMC) based inference has been found to work well in practice. However as the number of mentions grows to Web scale, as in our problem of cross-document coreference, even these inference techniques become infeasible, motivating the need for a scalable, parallelizable solution.

In this work we first distribute MCMC-based inference for the graphical model representation of coreference. Entities are distributed across the machines such that the parallel MCMC chains on the different machines use only local proposal distributions. After a fixed number of samples on each machine, we redistribute the entities among machines to enable proposals across entities that were previously on different machines. In comparison to the greedy approaches used in related work, our MCMC-based inference provides better robustness properties.

As the number of mentions becomes large, high-quality samples for MCMC become scarce. To facilitate better proposals, we present a hierarchical model. We add *sub-entity* variables that represent clusters of similar mentions that are likely to be coreferent; these are used to propose composite jumps that move multiple mentions together. We also introduce *super-entity* variables that represent clusters of similar entities; these are used to distribute entities among the machines such that similar entities are assigned to the same machine. These additional levels of hierarchy dramatically increase the probability of beneficial proposals even with a large number of entities and mentions.

To create a large corpus for evaluation, we identify pages that have hyperlinks to Wikipedia, and extract the anchor text and the context around the link. We treat the anchor text as the mention, the context as the document, and the title of the Wikipedia page as the entity label. Using this approach, 1.5 million mentions were annotated with 43k entity labels. On this dataset, our proposed model yields a B^3 (Bagga and Baldwin, 1998) F1 score of 73.7%, improving over the baseline by 16% absolute (corresponding to 38% error reduction). Our experimental results also show that our proposed hierarchical model converges much faster even though it contains many more variables.

2 Cross-document Coreference

The problem of coreference is to identify the sets of mention strings that refer to the same underlying entity. The identities and the number of the underlying entities is not known. In *within-document* coreference, the mentions occur in a single document. The number of mentions (and entities) in each document is usually in the hundreds. The difficulty of the task arises from a large hypothesis space (exponential in the number of mentions) and challenge in resolving nominal and pronominal mentions to the correct named mentions. In most cases, named mentions

are not ambiguous within a document. In *cross-document* coreference, the number of mentions and entities is in the millions, making the combinatorics even more daunting. Furthermore, naming ambiguity is much more common as the same string can refer to multiple entities in different documents, and distinct strings may refer to the same entity in different documents.

We show examples of ambiguities in Figure 1. Resolving the identity of individuals with the *same name* is a common problem in cross-document coreference. This problem is further complicated by the fact that in some situations, these individuals may belong to the same field. Another common ambiguity is that of *alternate* names, in which the same entity is referred to by different names or *aliases* (e.g. "Bill" is often used as a substitute for "William"). The figure also shows an example of the *renaming* ambiguity – "Lovebug Starski" refers to "Kevin Smith", and this is an extreme form of alternate names. Rare *singleton* entities (like the firefighter) that may appear only once in the whole corpus are also often difficult to isolate.

2.1 Pairwise Factor Model

Factor graphs are a convenient representation for a probability distribution over a vector of output variables given observed variables. The model that we use for coreference represents mentions ($\mathbf{M}$) and entities ($\mathbf{E}$) as random variables. Each mention can take an entity as its value, and each entity takes a set of mentions as its value. Each mention also has a feature vector extracted from the observed text mention and its context. More precisely, the probability of a configuration $\mathbf{E} = \mathbf{e}$ is defined by

$$p(\mathbf{e}) \propto \exp \sum_{e \in \mathbf{e}} \left\{ \sum_{m,n \in e, n \neq m} \psi_a(m,n) + \sum_{m \in e, n \notin e} \psi_r(m,n) \right\}$$

where factor ψ_a represents *affinity* between mentions that are coreferent according to $\mathbf{e}$, and factor ψ_r represents *repulsion* between mentions that are not coreferent. Different factors are instantiated for different predicted configurations. Figure 2 shows the model instantiated with five mentions over a two-entity hypothesis.

For the factor potentials, we use cosine similarity of mention context pairs (ϕ_{mn}) such that

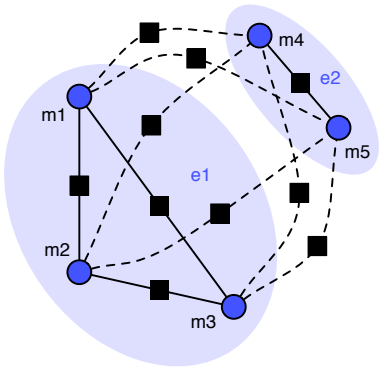


Figure 2: **Pairwise Coreference Model:** Factor graph for a 2-entity configuration of 5 mentions. Affinity factors are shown with solid lines, and repulsion factors with dashed lines.

$\psi_a(m,n) = \phi_{mn} - b$ and $\psi_r(m,n) = -(\phi_{mn} - b)$, where b is the bias. While one can certainly make use of a more sophisticated feature set, we leave this for future work as our focus is to scale up inference. However, it should be noted that this approach is agnostic to the particular set of features used. As we will note in the next section, we do not need to calculate features between all pairs of mentions (as would be prohibitively expensive for large datasets); instead we only compute the features as and when required.

2.2 MCMC-based Inference

Given the above model of coreference, we seek the *maximum a posteriori* (MAP) configuration:

$$\begin{aligned} \hat{\mathbf{e}} &= \arg\max_{\mathbf{e}} p(\mathbf{e}) \\ &= \arg\max_{\mathbf{e}} \sum_{e \in \mathbf{e}} \left\{ \sum_{m,n \in e, n \neq m} \psi_a(m,n) + \sum_{m \in e, n \notin e} \psi_r(m,n) \right\} \end{aligned}$$

Computing $\hat{\mathbf{e}}$ exactly is intractable due to the large space of possible configurations.[1] Instead, we employ MCMC-based optimization to discover the MAP configuration. A proposal function q is used to propose a change $\mathbf{e}'$ to the current configuration $\mathbf{e}$. This jump is accepted with the following Metropolis-Hastings acceptance probability:

$$\alpha(\mathbf{e}, \mathbf{e}') = \min\left(1, \left(\frac{p(\mathbf{e}')}{p(\mathbf{e})}\right)^{1/t} \frac{q(\mathbf{e})}{q(\mathbf{e}')}\right) \quad (1)$$

[1] Number of possible entities is Bell(n) in the number of mentions, i.e. number of partitions of n items

where t is the annealing temperature parameter.

MCMC chains efficiently explore the high-density regions of the probability distribution. By slowly reducing the temperature, we can decrease the entropy of the distribution to encourage convergence to the MAP configuration. MCMC has been used for optimization in a number of related work (McCallum et al., 2009; Goldwater and Griffiths, 2007; Changhe et al., 2004).

The proposal function moves a randomly chosen mention l from its current entity e_s to a randomly chosen entity e_t. For such a proposal, the log-model ratio is:

$$\log \frac{p(\mathbf{e}')}{p(\mathbf{e})} = \sum_{m \in e_t} \psi_a(l, m) + \sum_{n \in e_s} \psi_r(l, n) - \sum_{n \in e_s} \psi_a(l, n) - \sum_{m \in e_t} \psi_r(l, m) \quad (2)$$

Note that since only the factors between mention l and mentions in e_s and e_t are involved in this computation, the acceptance probability of each proposal is calculated efficiently.

In general, the model may contain arbitrarily complex set of features over pairs of mentions, with parameters associated with them. Given labeled data, these parameters can be *learned* by Perceptron (Collins, 2002), which uses the MAP configuration according to the model ($\hat{\mathbf{e}}$). There also exist more efficient training algorithms such as SampleRank (McCallum et al., 2009; Wick et al., 2009b) that update parameters *during* inference. However, we only focus on inference in this work, and the only parameter that we set manually is the bias b, which indirectly influences the number of entities in $\hat{\mathbf{e}}$. Unless specified otherwise, in this work the initial configuration for MCMC is the *singleton* configuration, i.e. all entities have a size of 1.

This MCMC inference technique, which has been used in McCallum and Wellner (2004), offers several advantages over other inference techniques: (a) unlike message-passing-methods, it does not require the full ground graph, (b) we only have to examine the factors that lie within the changed entities to evaluate a proposal, and (c) inference may be stopped at any point to obtain the current best configuration. However, the super exponential nature of the hypothesis space in cross-doc coreference renders this algorithm computationally unsuitable for large scale coreference tasks. In particular, fruitful proposals (that increase the model score) are extremely rare, resulting in a large number of proposals that are not accepted. We describe methods to speed up inference by 1) evaluating multiple proposal simultaneously (Section 3), and 2) by augmenting our model with hierarchical variables that enable better proposal distributions (Section 4).

3 Distributed MAP Inference

The key observation that enables distribution is that the acceptance probability computation of a proposal only examines a few factors that are **not** common to the previous and next configurations (Eq. 2). Consider a pair of proposals, one that moves mention l from entity e_s to entity e_t, and the other that moves mention l' from entity e'_s to entity e'_t. The set of factors to compute acceptance of the first proposal are factors between l and mentions in e_s and e_t, while the set of factors required to compute acceptance of the second proposal lie between l' and mentions in e'_s and e'_t. Since these set of factors are completely disjoint from each other, and the resulting configurations do not depend on each other, these two proposals are *mutually-exclusive*. Different orders of evaluating such proposals are equivalent, and in fact, these proposals can be proposed and evaluated concurrently. This mutual-exclusivity is not restricted only to pairs of proposals; a set of proposals are mutually-exclusive if no two proposals require the same factor for evaluation.

Using this insight, we introduce the following approach to distributed cross-document coreference. We divide the mentions and entities among multiple machines, and propose moves of mentions between entities assigned to the same machine. These jumps are evaluated exactly and accepted without communication between machines. Since acceptance of a mention's move requires examining factors that lie between other mentions in its entity, we ensure that all mentions of an entity are assigned the same machine. Unless specified otherwise, the distribution is performed randomly. To enable exploration of the complete configuration space, rounds of sampling are interleaved by *redistribution* stages, in which the entities are redistributed among the machines (see Figure 3). We use MapReduce (Dean and Ghe-

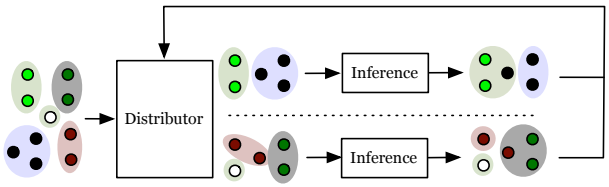


Figure 3: **Distributed MCMC-based Inference**: Distributor divides the entities among the machines, and the machines run inference. The process is repeated by the redistributing the entities.

mawat, 2004) to manage the distributed computation.

This approach to distribution is equivalent to inference with all mentions and entities on a single machine with a restricted proposer, but is faster since it exploits independencies to propose multiple jumps simultaneously. By restricting the jumps as described above, the acceptance probability calculation is exact. Partitioning the entities and proposing local jumps are restrictions to the single-machine proposal distribution; redistribution stages ensure the equivalent Markov chains are still irreducible. See Singh et al. (2010) for more details.

4 Hierarchical Coreference Model

The proposal function for MCMC-based MAP inference presents changes to the current entities. Since we use MCMC to reach high-scoring regions of the hypothesis space, we are interested in the changes that improve the current configuration. But as the number of mentions and entities increases, these *fruitful* samples become extremely rare due to the blowup in the possible space of configurations, resulting in rejection of a large number of proposals. By distributing as described in the previous section, we propose samples in parallel, improving chances of finding changes that result in better configurations. However, due to random redistribution and a naive proposal function within each machine, a large fraction of proposals are still wasted. We address these concerns by adding *hierarchy* to the model.

4.1 Sub-Entities

Consider the task of proposing moves of mentions (within a machine). Given the large number of mentions and entities, the probability that a randomly picked mention that is moved to a random entity results in a better configuration is extremely small. If such a move is accepted, this gives us evidence that the mention did not belong to the previous entity, and we should also move similar mentions from the previous entity simultaneously to the same entity. Since the proposer moves only a single mention at a time, a large number of samples may be required to discover these fruitful moves. To enable *block* proposals that move similar mentions simultaneously, we introduce latent *sub-entity* variables that represent groups of similar mentions within an entity, where the similarity is defined by the model. For inference, we have stages of sampling sub-entities (moving individual mentions) interleaved with stages of entity sampling (moving all mentions within a sub-entity). Even though our configuration space has become larger due to these extra variables, the proposal distribution has also improved since it proposes composite moves.

4.2 Super-Entities

Another issue faced during distributed inference is that random redistribution is often wasteful. For example, if dissimilar entities are assigned to a machine, none of the proposals may be accepted. For a large number of entities and machines, the probability that similar entities will be assigned to the same machine is extremely small, leading to a larger number of wasted proposals. To alleviate this problem, we introduce *super-entities* that represent groups of similar entities. During redistribution, we ensure all entities in the same super-entity are assigned to the same machine. As for sub-entities above, inference switches between regular sampling of entities and sampling of super-entities (by moving entities). Although these extra variables have made the configuration space larger, they also allow more efficient distribution of entities, leading to useful proposals.

4.3 Combined Hierarchical Model

Each of the described levels of the hierarchy are similar to the initial model (Section 2.1): mentions/sub-entities have the same structure as the entities/super-entities, and are modeled using similar factors. To represent the "context" of a sub-entity we take the union of the bags-of-words of the constituent mention contexts. Similarly, we take the union of sub-

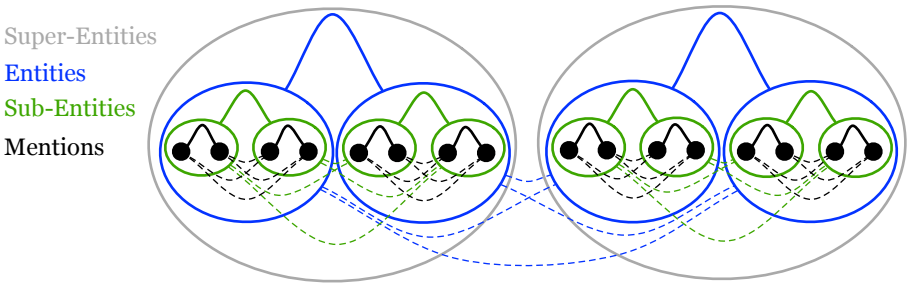


Figure 4: **Combined Hierarchical Model** with factors instantiated for a hypothesis containing 2 super-entities, 4 entities, and 8 sub-entities, shown as colored circles, over 16 mentions. Dotted lines represent repulsion factors and solid lines represent affinity factors (the color denotes the type of variable that the factor touches). The boxes on factors were excluded for clarity.

entity contexts to represent the context of an entity. The factors are instantiated in the same manner as Section 2.1 except that we change the bias factor b for each level (increasing it for sub-entities, and decreasing it for super-entities). The exact values of these biases indirectly determines the number of predicted sub-entities and super-entities.

Since these two levels of hierarchy operate at separate granularities from each other, we combine them into a single hierarchical model that contains both sub- and super-entities. We illustrate this hierarchical structure in Figure 4. Inference for this model takes a round-robin approach by fixing two of the levels of the hierarchy and sampling the third, cycling through these three levels. Unless specified otherwise, the initial configuration is the *singleton* configuration, in which all sub-entities, entities, and super-entities are of size 1.

5 Experiments

We evaluate our models and algorithms on a number of datasets. First, we compare performance on the small, publicly-available "John Smith" dataset. Second, we run the automated *Person-X* evaluation to obtain thousands of mentions that we use to demonstrate accuracy and scalability improvements. Most importantly, we create a large labeled corpus using links to Wikipedia to explore the performance in the large-scale setting.

5.1 John Smith Corpus

To compare with related work, we run an evaluation on the "John Smith" corpus (Bagga and Baldwin, 1998), containing 197 mentions of the name "John Smith" from New York Times articles (labeled to obtain 35 true entities). The bias b for our approach is set to result in the correct number of entities. Our model achieves B^3 F1 accuracy of 66.4% on this dataset. In comparison, Rao et al. (2010) obtains 61.8% using the model most similar to ours, while their best model (which uses sophisticated topic-model features that do not scale easily) achieves 69.7%. It is encouraging to note that our approach, using only a subset of the features, performs competitively with related work. However, due to the small size of the dataset, we require further evaluation before reaching any conclusions.

5.2 Person-X Evaluation

There is a severe lack of labeled corpora for cross-document coreference due to the effort required to evaluate the coreference decisions. Related approaches have used automated *Person-X* evaluation (Gooi and Allan, 2004), in which unique person-name strings are treated as the true entity labels for the mentions. Every mention string is replaced with an "X" for the coreference system. We use this evaluation methodology on 25k person-name mentions from the New York Times corpus (Sandhaus, 2008) each with one of 50 unique strings. As before, we set the bias b to achieve the same number of entities. We use 1 million samples in each round of inference, followed by random redistribution in the flat model, and super-entities in the hierarchical model. Results are averaged over five runs.

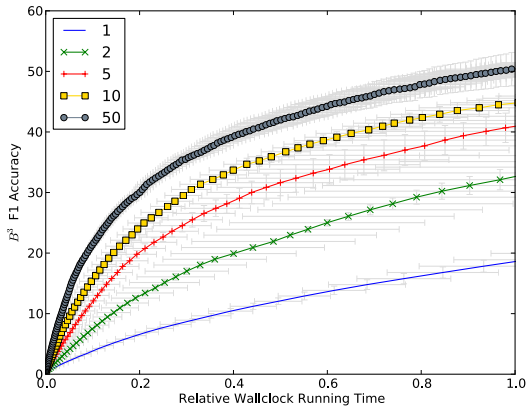


Figure 5: **Person-X Evaluation of Pairwise model**: Performance as number of machines is varied, averaged over 5 runs.

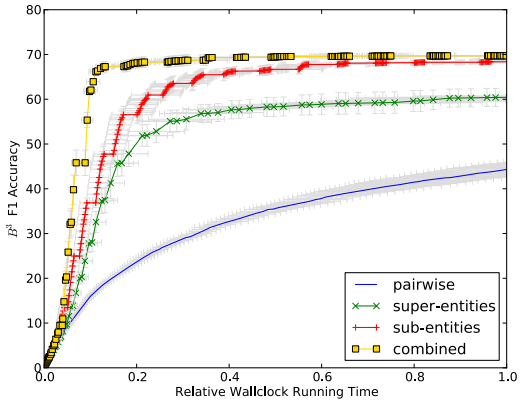


Figure 6: **Person-X Evaluation of Hierarchical Models**: Performance of inference on hierarchical models compared to the pairwise model. Experiments were run using 50 machines.

Number of Entities	43,928
Number of Mentions	1,567,028
Size of Largest Entity	6,096
Average Mentions per Entity	35.7
Variance of Mentions per Entity	5191.7

Table 1: **Wikipedia Link Corpus Statistics**. Size of an entity is the number of mentions of that entity.

Figure 5 shows accuracy compared to relative wallclock running time for distributed inference on the flat, pairwise model. Speed and accuracy improve as additional machines are added, but larger number of machines lead to diminishing returns for this small dataset. Distributed inference on our hierarchical model is evaluated in Figure 6 against inference on the pairwise model from Figure 5. We see that the individual hierarchical models perform much better than the pairwise model; they achieve the same accuracy as the pairwise model in approximately 10% of the time. Moreover, distributed inference on the combined hierarchical model is both faster and more accurate than the individual hierarchical models.

5.3 Wikipedia Link Corpus

To explore the application of the proposed approach to a larger, realistic dataset, we construct a corpus based on the insight that links to Wikipedia that appear on webpages can be treated as mentions, and since the links were added manually by the page author, we use the destination Wikipedia page as the entity the link refers to.

The dataset is created as follows: First, we crawl the web and select hyperlinks on webpages that link to an English Wikipedia page.[2] The anchors of these links form our set of *mentions*, with the surrounding block of clean text (obtained after removing markup, etc.) around each link being its context. We assign the title of the linked Wikipedia page as the *entity* label of that link. Since this set of mentions and labels can be noisy, we use the following filtering steps. All links that have less than 36 words in their block, or whose anchor text has a large string edit distance from the title of the Wikipedia page, are discarded. While this results in cases in which "President" is discarded when linked to the "Barack Obama" Wikipedia page, it was necessary to reduce noise. Further, we also discard links to Wikipedia pages that are concepts (such as "`public_domain`") rather than entities. All entities with less than 6 links to them are also discarded.

Table 1 shows some statistics about our automatically generated data set. We randomly sampled 5% of the entities to create a development set, treating the remaining entities as the test set. Unlike the John Smith and Person-X evaluation, this data set also contains non-person entities such as organizations and locations.

For our models, we augment the factor potentials with mention-string similarity:

[2] e.g. `http://en.wikipedia.org/Hillary_Clinton`

$$\psi_{a/r}(m,n) = \pm\left(\phi_{mn} - b + w\text{STREQ}(m,n)\right)$$

where STREQ is 1 if mentions m and n are string identical (0 otherwise), and w is the weight to this feature.[3] In our experiments we found that setting $w = 0.8$ and $b = 1e-4$ gave the best results on the development set.

Due to the large size of the corpus, existing cross-document coreference approaches could not be applied to this dataset. However, since a majority of related work consists of using clustering after defining a similarity function (Section 6), we provide a baseline evaluation of clustering with *Subsquare* (Bshouty and Long, 2010), a scalable, distributed clustering method. Subsquare takes as input a weighted graph with mentions as nodes and similarity between mentions used as edge weights. Subsquare works by stochastically assigning a vertex to the cluster of one its neighbors if they have significant neighborhood overlap. This algorithm is an efficient form of approximate spectral clustering (Bshouty and Long, 2010), and since it is given the same distances between mentions as our models, we expect it to get similar accuracy. We also generate another baseline clustering by assigning mentions with identical strings to the same entity. This mention-string clustering is also used as the initial configuration of our inference.

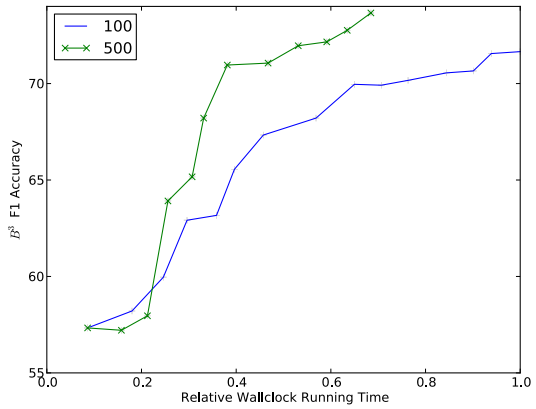


Figure 7: **Wikipedia Link Evaluation**: Performance of inference for different number of machines ($N = 100, 500$). Mention-string match clustering is used as the initial configuration.

[3]Note that we do not use mention-string similarity for John Smith or Person-X as the mention strings are all identical.

Method	Pairwise		B^3 Score	
	P/ R	F1	P/ R	F1
String-Match	30.0 / 66.7	41.5	82.7 / 43.8	57.3
Subsquare	38.2 / 49.1	43.0	87.6 / 51.4	64.8
Our Model	44.2 / 61.4	**51.4**	89.4 / 62.5	**73.7**

Table 2: **F1 Scores on the Wikipedia Link Data**. The results are significant at the 0.0001 level over Subsquare according to the difference of proportions significance test.

Inference is run for 20 rounds of 10 million samples each, distributed over N machines. We use $N = 100, 500$ and the B^3 F1 score results obtained set for each case are shown in Figure 7. It can be seen that $N = 500$ converges to a better solution faster, showing effective use of parallelism. Table 2 compares the results of our approach (at convergence for $N = 500$), the baseline mention-string match and the Subsquare algorithm. Our approach significantly outperforms the competitors.

6 Related Work

Although the cross-document coreference problem is challenging and lacks large labeled datasets, its ubiquitous role as a key component of many knowledge discovery tasks has inspired several efforts.

A number of previous techniques use scoring functions between pairs of contexts, which are then used for clustering. One of the first approaches to cross-document coreference (Bagga and Baldwin, 1998) uses an idf-based cosine-distance scoring function for pairs of contexts, similar to the one we use. Ravin and Kazi (1999) extend this work to be somewhat scalable by comparing pairs of contexts only if the mentions are deemed "ambiguous" using a heuristic. Others have explored multiple methods of context similarity, and concluded that agglomerative clustering provides effective means of inference (Gooi and Allan, 2004). Pedersen et al. (2006) and Purandare and Pedersen (2004) integrate second-order co-occurrence of words into the similarity function. Mann and Yarowsky (2003) use biographical facts from the Web as features for clustering. Niu et al. (2004) incorporate information extraction into the context similarity model, and annotate a small dataset to learn the parameters. A number of other approaches include various forms of

hand-tuned weights, dictionaries, and heuristics to define similarity for name disambiguation (Blume, 2005; Baron and Freedman, 2008; Popescu et al., 2008). These approaches are greedy and differ in the choice of the distance function and the clustering algorithm used. Daumé III and Marcu (2005) propose a generative approach to supervised clustering, and Haghighi and Klein (2010) use entity profiles to assist within-document coreference.

Since many related methods use clustering, there are a number of distributed clustering algorithms that may help scale these approaches. Datta et al. (2006) propose an algorithm for distributed k-means. Chen et al. (2010) describe a parallel spectral clustering algorithm. We use the Subsquare algorithm (Bshouty and Long, 2010) as baseline because it works well in practice. Mocian (2009) presents a survey of distributed clustering algorithms.

Rao et al. (2010) have proposed an online deterministic method that uses a stream of input mentions and assigns them greedily to entities. Although it can resolve mentions from non-trivial sized datasets, the method is restricted to a single machine, which is not scalable to the very large number of mentions that are encountered in practice.

Our representation of the problem as an undirected graphical model, and performing distributed inference on it, provides a combination of advantages not available in any of these approaches. First, most of the methods will not scale to the hundreds of millions of mentions that are present in real-world applications. By utilizing parallelism across machines, our method can run on very large datasets simply by increasing the number of machines used. Second, approaches that use clustering are limited to using pairwise distance functions for which additional supervision and features are difficult to incorporate. In addition to representing features from all of the related work, graphical models can also use more complex entity-wide features (Culotta et al., 2007; Wick et al., 2009a), and parameters can be learned using supervised (Collins, 2002) or semi-supervised techniques (Mann and McCallum, 2008). Finally, the inference for most of the related approaches is *greedy*, and earlier decisions are not revisited. Our technique is based on MCMC inference and simulated annealing, which are able to escape local maxima.

7 Conclusions

Motivated by the problem of solving the coreference problem on billions of mentions from all of the newswire documents from the past few decades, we make the following contributions. First, we introduce distributed version of MCMC-based inference technique that can utilize parallelism to enable scalability. Second, we augment the model with hierarchical variables that facilitate fruitful proposal distributions. As an additional contribution, we use links to Wikipedia pages to obtain a high-quality cross-document corpus. Scalability and accuracy gains of our method are evaluated on multiple datasets.

There are a number of avenues for future work. Although we demonstrate scalability to more than a million mentions, we plan to explore performance on datasets in the billions. We also plan to examine inference on complex coreference models (such as with entity-wide factors). Another possible avenue for future work is that of learning the factors. Since our approach supports parameter estimation, we expect significant accuracy gains with additional features and supervised data. Our work enables cross-document coreference on very large corpora, and we would like to explore the downstream applications that can benefit from it.

Acknowledgments

This work was done when the first author was an intern at Google Research. The authors would like to thank Mark Dredze, Sebastian Riedel, and anonymous reviewers for their valuable feedback. This work was supported in part by the Center for Intelligent Information Retrieval, the University of Massachusetts gratefully acknowledges the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181., in part by an award from Google, in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0326249, in part by NSF grant #CNS-0958392, and in part by UPenn NSF medium IIS-0803847. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Amit Bagga and Breck Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *International Conference on Computational Linguistics*, pages 79–85.

A. Baron and M. Freedman. 2008. Who is who and what is what: experiments in cross-document co-reference. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 274–283.

Eric Bengston and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Matthias Blume. 2005. Automatic entity disambiguation: Benefits to NER, relation extraction, link analysis, and inference. In *International Conference on Intelligence Analysis (ICIA)*.

Nader H. Bshouty and Philip M. Long. 2010. Finding planted partitions in nearly linear time using arrested spectral clustering. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 135–142, Haifa, Israel, June. Omnipress.

Yuan Changhe, Lu Tsai-Ching, and Druzdzel Marek. 2004. Annealed MAP. In *Uncertainty in Artificial Intelligence (UAI)*, pages 628–635, Arlington , Virginia. AUAI Press.

Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y. Chang. 2010. Parallel spectral clustering in distributed systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithm. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Aron Culotta, Michael Wick, and Andrew McCallum. 2007. First-order probabilistic models for coreference resolution. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*.

S. Datta, C. Giannella, and H. Kargupta. 2006. K-Means Clustering over a Large, Dynamic Network. In *SIAM Data Mining Conference (SDM)*.

Hal Daumé III and Daniel Marcu. 2005. A Bayesian model for supervised clustering with the Dirichlet process prior. *Journal of Machine Learning Research (JMLR)*, 6:1551–1577.

Jeffrey Dean and Sanjay Ghemawat. 2004. Mapreduce: Simplified data processing on large clusters. *Symposium on Operating Systems Design & Implementation (OSDI)*.

Sharon Goldwater and Tom Griffiths. 2007. A fully bayesian approach to unsupervised part-of-speech tagging. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 744–751.

Chung Heong Gooi and James Allan. 2004. Cross-document coreference on a large scale corpus. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, pages 9–16.

Aria Haghighi and Dan Klein. 2007. Unsupervised coreference resolution in a nonparametric bayesian model. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 848–855.

Aria Haghighi and Dan Klein. 2009. Simple coreference resolution with rich syntactic and semantic features. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1152–1161.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, pages 385–393.

Gideon S. Mann and Andrew McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 870–878.

Gideon S. Mann and David Yarowsky. 2003. Unsupervised personal name disambiguation. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, pages 33–40.

Andre Martins, Noah Smith, Eric Xing, Pedro Aguiar, and Mario Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 34–44, Cambridge, MA, October. Association for Computational Linguistics.

J. Mayfield, D. Alexander, B. Dorr, J. Eisner, T. Elsayed, T. Finin, C. Fink, M. Freedman, N. Garera, P. McNamee, et al. 2009. Cross-document coreference resolution: A key technology for learning by reading. In *AAAI Spring Symposium on Learning by Reading and Learning to Read*.

Andrew McCallum and Ben Wellner. 2004. Conditional models of identity uncertainty with application to noun coreference. In *Neural Information Processing Systems (NIPS)*.

Andrew McCallum, Karl Schultz, and Sameer Singh. 2009. FACTORIE: Probabilistic programming via imperatively defined factor graphs. In *Neural Information Processing Systems (NIPS)*.

Horatiu Mocian. 2009. *Survey of Distributed Clustering Techniques*. Ph.D. thesis, Imperial College of London.

Vincent Ng. 2005. Machine learning for coreference resolution: From local classification to global ranking. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Cheng Niu, Wei Li, and Rohini K. Srihari. 2004. Weakly supervised learning for cross-document person name disambiguation supported by information extraction. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, page 597.

Ted Pedersen, Anagha Kulkarni, Roxana Angheluta, Zornitsa Kozareva, and Thamar Solorio. 2006. An unsupervised language independent method of name discrimination using second order co-occurrence features. In *International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, pages 208–222.

Hoifung Poon, Pedro Domingos, and Marc Sumner. 2008. A general method for reducing the complexity of relational inference and its application to MCMC. In *AAAI Conference on Artificial Intelligence*.

Octavian Popescu, Christian Girardi, Emanuele Pianta, and Bernardo Magnini. 2008. Improving cross-document coreference. *Journées Internationales d'Analyse statistique des Données Textuelles*, 9:961–969.

A. Purandare and T. Pedersen. 2004. Word sense discrimination by clustering contexts in vector and similarity spaces. In *Conference on Computational Natural Language Learning (CoNLL)*, pages 41–48.

Delip Rao, Paul McNamee, and Mark Dredze. 2010. Streaming cross document entity coreference resolution. In *International Conference on Computational Linguistics (COLING)*, pages 1050–1058, Beijing, China, August. Coling 2010 Organizing Committee.

Yael Ravin and Zunaid Kazi. 1999. Is Hillary Rodham Clinton the president? disambiguating names across documents. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 9–16.

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

Evan Sandhaus. 2008. The New York Times annotated corpus. *Linguistic Data Consortium*.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2010. Distributed map inference for undirected graphical models. In *Neural Information Processing Systems (NIPS), Workshop on Learning on Cores, Clusters and Clouds*.

Ben Wellner, Andrew McCallum, Fuchun Peng, and Michael Hay. 2004. An integrated, conditional model of information extraction and coreference with application to citation matching. In *Uncertainty in Artificial Intelligence (UAI)*, pages 593–601.

Michael Wick, Aron Culotta, Khashayar Rohanimanesh, and Andrew McCallum. 2009a. An entity-based model for coreference resolution. In *SIAM International Conference on Data Mining (SDM)*.

Michael Wick, Khashayar Rohanimanesh, Aron Culotta, and Andrew McCallum. 2009b. Samplerank: Learning preferences from atomic gradients. In *Neural Information Processing Systems (NIPS), Workshop on Advances in Ranking*.

A Cross-Lingual ILP Solution to Zero Anaphora Resolution

Ryu Iida
Tokyo Institute of Technology
2-12-1, Ôokayama, Meguro,
Tokyo 152-8552, Japan
ryu-i@cl.cs.titech.ac.jp

Massimo Poesio
Università di Trento,
Center for Mind / Brain Sciences
University of Essex,
Language and Computation Group
massimo.poesio@unitn.it

Abstract

We present an ILP-based model of zero anaphora detection and resolution that builds on the joint determination of anaphoricity and coreference model proposed by Denis and Baldridge (2007), but revises it and extends it into a three-way ILP problem also incorporating subject detection. We show that this new model outperforms several baselines and competing models, as well as a direct translation of the Denis / Baldridge model, for both Italian and Japanese zero anaphora. We incorporate our model in complete anaphoric resolvers for both Italian and Japanese, showing that our approach leads to improved performance also when not used in isolation, provided that separate classifiers are used for zeros and for explicitly realized anaphors.

1 Introduction

In so-called 'pro-drop' languages such as Japanese and many romance languages including Italian, phonetic realization is not required for anaphoric references in contexts in which in English non-contrastive pronouns are used: e.g., the subjects of Italian and Japanese translations of *buy* in (1b) and (1c) are not explicitly realized. We call these non-realized mandatory arguments **zero anaphors**.

(1) a. [EN] [John]$_i$ went to visit some friends. On the way, [he]$_i$ bought some wine.
b. [IT] [Giovanni]$_i$ andò a far visita a degli amici. Per via, ϕ_i comprò del vino.
c. [JA] [John]$_i$-wa yujin-o houmon-sita. Tochu-de ϕ_i wain-o ka-tta.

The felicitousness of zero anaphoric reference depends on the referred entity being sufficiently salient, hence this type of data–particularly in Japanese and Italian–played a key role in early work in coreference resolution, e.g., in the development of Centering (Kameyama, 1985; Walker et al., 1994; Di Eugenio, 1998). This research highlighted both commonalities and differences between the phenomenon in such languages. Zero anaphora resolution has remained a very active area of study for researchers working on Japanese, because of the prevalence of zeros in such languages[1] (Seki et al., 2002; Isozaki and Hirao, 2003; Iida et al., 2007a; Taira et al., 2008; Imamura et al., 2009; Sasano et al., 2009; Taira et al., 2010). But now the availability of corpora annotated to study anaphora, including zero anaphora, in languages such as Italian (e.g., Rodriguez et al. (2010)), and their use in competitions such as SEMEVAL 2010 Task 1 on Multilingual Coreference (Recasens et al., 2010), is leading to a renewed interest in zero anaphora resolution, particularly at the light of the mediocre results obtained on zero anaphors by most systems participating in SEMEVAL.

Resolving zero anaphora requires the simultaneous decision that one of the arguments of a verb is phonetically unrealized (and which argument exactly–in this paper, we will only be concerned with subject zeros as these are the only type to occur in Italian) and that a particular entity is its antecedent. It is therefore natural to view zero anaphora resolution as a joint inference

[1] As shown in Table 1, 64.3% of anaphors in the NAIST Text Corpus of Anaphora are zeros.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 804–813,
Portland, Oregon, June 19-24, 2011.

task, for which Integer Linear Programming (ILP)– introduced to NLP by Roth and Yih (2004) and successfully applied by Denis and Baldridge (2007) to the task of jointly inferring anaphoricity and determining the antecedent–would be appropriate.

In this work we developed, starting from the ILP system proposed by Denis and Baldridge, an ILP approach to zero anaphora detection and resolution that integrates (revised) versions of Denis and Baldridge's constraints with additional constraints between the values of three distinct classifiers, one of which is a novel one for subject prediction. We demonstrate that treating zero anaphora resolution as a three-way inference problem is successful for both Italian and Japanese. We integrate the zero anaphora resolver with a coreference resolver and demonstrate that the approach leads to improved results for both Italian and Japanese.

The rest of the paper is organized as follows. Section 2 briefly summarizes the approach proposed by Denis and Baldridge (2007). We next present our new ILP formulation in Section 3. In Section 4 we show the experimental results with zero anaphora only. In Section 5 we discuss experiments testing that adding our zero anaphora detector and resolver to a full coreference resolver would result in overall increase in performance. We conclude and discuss future work in Section 7.

2 Using ILP for joint anaphoricity and coreference determination

Integer Linear Programming (ILP) is a method for constraint-based inference aimed at finding the values for a set of variables that maximize a (linear) **objective function** while satisfying a number of constraints. Roth and Yih (2004) advocated ILP as a general solution for a number of NLP tasks that require combining multiple classifiers and which the traditional pipeline architecture is not appropriate, such as entity disambiguation and relation extraction.

Denis and Baldridge (2007) defined the following object function for the joint anaphoricity and coreference determination problem.

$$\min \sum_{\langle i,j\rangle \in P} c^{C}_{\langle i,j\rangle} \cdot x_{\langle i,j\rangle} + c^{-C}_{\langle i,j\rangle} \cdot (1 - x_{\langle i,j\rangle}) + \sum_{j \in M} c^{A}_{j} \cdot y_j + c^{-A}_{j} \cdot (1 - y_j) \quad (2)$$

subject to

$$x_{\langle i,j\rangle} \in \{0,1\} \qquad \forall \langle i,j\rangle \in P$$
$$y_j \in \{0,1\} \qquad \forall j \in M$$

M stands for the set of mentions in the document, and P the set of possible coreference links over these mentions. $x_{\langle i,j\rangle}$ is an indicator variable that is set to 1 if mentions i and j are coreferent, and 0 otherwise. y_j is an indicator variable that is set to 1 if mention j is anaphoric, and 0 otherwise. The costs $c^{C}_{\langle i,j\rangle} = -log(P(\mathrm{COREF}|i,j))$ are (logs of) probabilities produced by an antecedent identification classifier with $-log$, whereas $c^{A}_{j} = -log(P(\mathrm{ANAPH}|j))$, are the probabilities produced by an anaphoricity determination classifier with $-log$. In the Denis & Baldridge model, the search for a solution to antecedent identification and anaphoricity determination is guided by the following three constraints.

Resolve only anaphors: if a pair of mentions $\langle i,j\rangle$ is coreferent ($x_{\langle i,j\rangle} = 1$), then mention j must be anaphoric ($y_j = 1$).

$$x_{\langle i,j\rangle} \leq y_j \qquad \forall \langle i,j\rangle \in P \quad (3)$$

Resolve anaphors: if a mention is anaphoric ($y_j = 1$), it must be coreferent with at least one antecedent.

$$y_j \leq \sum_{i \in M_j} x_{\langle i,j\rangle} \qquad \forall j \in M \quad (4)$$

Do not resolve non-anaphors: if a mention is non-anaphoric ($y_j = 0$), it should have no antecedents.

$$y_j \geq \frac{1}{|M_j|} \sum_{i \in M_j} x_{\langle i,j\rangle} \qquad \forall j \in M \quad (5)$$

3 An ILP-based account of zero anaphora detection and resolution

In the corpora used in our experiments, zero anaphora is annotated using as markable the first verbal form (not necessarily the head) following the position where the argument would have been realized, as in the following example.

(6) [Pahor]$_i$ è nato a Trieste, allora porto principale dell'Impero Austro-Ungarico.
A sette anni [vide]$_i$ l'incendio del Narodni dom,

The proposal of Denis and Baldridge (2007) can be easily turned into a proposal for the task of detecting and resolving zero anaphora in this type of data by reinterpreting the indicator variables as follows:

- y_j is 1 if markable j (a verbal form) initiates a verbal complex whose subject is unrealized, 0 otherwise;
- $x_{\langle i,j \rangle}$ is 1 if the empty mention realizing the subject argument of markable j and markable i are mentions of the same entity, 0 otherwise.

There are however a number of ways in which this direct adaptation can be modified and extended. We discuss them in turn.

3.1 Best First

In the context of zero anaphora resolution, the 'Do not resolve non-anaphors' constraint (5) is too weak, as it allows the redundant choice of more than one candidate antecedent. We developed therefore the following alternative, that blocks selection of more than one antecedent.

Best First (BF):

$$y_j \geq \sum_{i \in M_j} x_{\langle i,j \rangle} \qquad \forall j \in M \qquad (7)$$

3.2 A subject detection model

The greatest difficulty in zero anaphora resolution in comparison to, say, pronoun resolution, is zero anaphora detection. Simply relying for this on the parser is not enough: most dependency parsers are not very accurate at identifying cases in which the verb does not have a subject on syntactic grounds only. Again, it seems reasonable to suppose this is because zero anaphora detection requires a combination of syntactic information and information about the current context. Within the ILP framework, this hypothesis can be implemented by turning the zero anaphora resolution optimization problem into one with *three* indicator variables, with the objective function in (8). The third variable, z_j, encodes the information provided by the parser: it is 1 with cost $c_j^S = -log(P(SUBJ|j))$ if the parser thinks that verb j has an explicit subject with probability $P(SUBJ|j)$, otherwise it is 0.

$$\begin{aligned} \min \sum_{\langle i,j \rangle \in P} & c^C_{\langle i,j \rangle} \cdot x_{\langle i,j \rangle} + c^{-C}_{\langle i,j \rangle} \cdot (1 - x_{\langle i,j \rangle}) \\ & + \sum_{j \in M} c^A_j \cdot y_j + c^{-A}_j \cdot (1 - y_j) \\ & + \sum_{j \in M} c^S_j \cdot z_j + c^{-S}_j \cdot (1 - z_j) \quad (8) \end{aligned}$$

$$\begin{aligned} \text{subject to} \\ x_{\langle i,j \rangle} &\in \{0,1\} && \forall \langle i,j \rangle \in P \\ y_j &\in \{0,1\} && \forall j \in M \\ z_j &\in \{0,1\} && \forall j \in M \end{aligned}$$

The crucial fact about the relation between z_j and y_j is that a verb has either a syntactically realized NP or a zero pronoun as a subject, but not both. This is encoded by the following constraint.

Resolve only non-subjects: if a predicate j syntactically depends on a subject ($z_j = 1$), then the predicate j should have no antecedents of its subject zero pronoun.

$$y_j + z_j \leq 1 \qquad \forall j \in M \qquad (9)$$

4 Experiment 1: zero anaphora resolution

In a first round of experiments, we evaluated the performance of the model proposed in Section 3 on zero anaphora only (i.e., not attempting to resolve other types of anaphoric expressions).

4.1 Data sets

We use the two data sets summarized in Table 1. The table shows that NP anaphora occurs more frequently than zero-anaphora in Italian, whereas in Japanese the frequency of anaphoric zero-anaphors[2] is almost double the frequency of the remaining anaphoric expressions.

Italian For Italian coreference, we used the annotated data set presented in Rodriguez et al. (2010) and developed for the Semeval 2010 task 'Coreference Resolution in Multiple Languages' (Recasens et al., 2010), where both zero-anaphora and NP

[2] In Japanese, like in Italian, zero anaphors are often used non-anaphorically, to refer to situationally introduced entities, as in *I went to John's office, but they told me that he had left.*

					#instances (anaphoric/total)		
language	type	#docs	#sentences	#words	zero-anaphors	others	all
Italian	train	97	3,294	98,304	1,093 / 1,160	6,747 / 27,187	7,840 / 28,347
	test	46	1,478	41,587	792 / 837	3,058 / 11,880	3,850 / 12,717
Japanese	train	1,753	24,263	651,986	18,526 / 29,544	10,206 / 161,124	28,732 / 190,668
	test	696	9,287	250,901	7,877 / 11,205	4,396 / 61,652	12,273 / 72,857

In the 6th column we use the term 'anaphoric' to indicate the number of zero anaphors that have an antecedent in the text, whereas the total figure is the sum of anaphoric and *exophoric* zero-anaphors - zeros with a vague / generic reference.

Table 1: Italian and Japanese Data Sets

coreference are annotated. This dataset consists of articles from Italian Wikipedia, tokenized, POS-tagged and morphologically analyzed using TextPro, a freely available Italian pipeline (Pianta et al., 2008). We parsed the corpus using the Italian version of the DESR dependency parser (Attardi et al., 2007).

In Italian, zero pronouns may only occur as omitted subjects of verbs. Therefore, in the task of zero-anaphora resolution all verbs appearing in a text are considered candidates for zero pronouns, and all gold mentions or system mentions preceding a candidate zero pronoun are considered as candidate antecedents. (In contrast, in the experiments on coreference resolution discussed in the following section, all mentions are considered as both candidate anaphors and candidate antecedents. To compare the results with gold mentions and with system detected mentions, we carried out an evaluation using the mentions automatically detected by the Italian version of the BART system (I-BART) (Poesio et al., 2010), which is freely downloadable.[3]

Japanese For Japanese coreference we used the NAIST Text Corpus (Iida et al., 2007b) version 1.4β, which contains the annotated data about NP coreference and zero-anaphoric relations. We also used the Kyoto University Text Corpus[4] that provides dependency relations information for the same articles as the NAIST Text Corpus. In addition, we also used a Japanese named entity tagger, *CaboCha*[5] for automatically tagging named entity labels. In the NAIST Text Corpus mention boundaries are not annotated, only the heads. Thus, we considered as pseudo-mentions all *bunsetsu* chunks (i.e. base phrases in Japanese) whose head part-of-speech was automatically tagged by the Japanese morphological analyser *Chasen*[6] as either 'noun' or 'unknown word' according to the NAIST-jdic dictionary.[7]

For evaluation, articles published from January 1st to January 11th and the editorials from January to August were used for training and articles dated January 14th to 17th and editorials dated October to December are used for testing as done by Taira et al. (2008) and Imamura et al. (2009). Furthermore, in the experiments we only considered *subject* zero pronouns for a fair comparison to Italian zero-anaphora.

4.2 Models

In these first experiments we compared the three ILP-based models discussed in Section 3: the direct reimplementation of the Denis and Baldridge proposal (i.e., using the same constrains), a version replacing Do-Not-Resolve-Not-Anaphors with Best-First, and a version with Subject Detection as well.

As discussed by Iida et al. (2007a) and Imamura et al. (2009), useful features in intra-sentential zero-anaphora are different from ones in inter-sentential zero-anaphora because in the former problem syntactic information between a zero pronoun and its candidate antecedent is essential, while the latter needs to capture the significance of saliency based on Centering Theory (Grosz et al., 1995). To directly reflect this difference, we created two antecedent identification models; one for intra-sentential zero-anaphora, induced using the training instances which a zero pronoun and its candidate antecedent appear in the same sentences, the other for

[3] http://www.bart-coref.org/

[4] http://www-lab25.kuee.kyoto-u.ac.jp/nl-resource/corpus.html

[5] http://chasen.org˜taku/software/cabocha/

[6] http://chasen-legacy.sourceforge.jp/

[7] http://sourceforge.jp/projects/naist-jdic/

inter-sentential cases, induced from the remaining training instances.

To estimate the feature weights of each classifier, we used MEGAM[8], an implementation of the Maximum Entropy model, with default parameter settings. The ILP-based models were compared with the following baselines.

PAIRWISE: as in the work by Soon et al. (2001), antecedent identification and anaphoricity determination are simultaneously executed by a single classifier.

DS-CASCADE: the model first filters out non-anaphoric candidate anaphors using an anaphoricity determination model, then selects an antecedent from a set of candidate antecedents of anaphoric candidate anaphors using an antecedent identification model.

4.3 Features

The feature sets for antecedent identification and anaphoricity determination are briefly summarized in Table 2 and Table 3, respectively. The agreement features such as NUM_AGREE and GEN_AGREE are automatically derived using TextPro. Such agreement features are not available in Japanese because Japanese words do not contain such information.

4.4 Creating subject detection models

To create a subject detection model for Italian, we used the TUT corpus[9] (Bosco et al., 2010), which contains manually annotated dependency relations and their labels, consisting of 80,878 tokens in CoNLL format. We induced an maximum entropy classifier by using as items all arcs of dependency relations, each of which is used as a positive instance if its label is subject; otherwise it is used as a negative instance.

To train the Japanese subject detection model we used 1,753 articles contained both in the NAIST Text Corpus and the Kyoto University Text Corpus. By merging these two corpora, we can obtain the annotated data including which dependency arc is subject[10]. To create the training instances, any pair of a predicate and its dependent are extracted, each of which is judged as positive if its relation is subject; as negative otherwise.

feature	description
SUBJ_PRE	1 if subject is included in the preceding words of *ZERO* in a sentence; otherwise 0.
TOPIC_PRE*	1 if topic case marker appears in the preceding words of *ZERO* in a sentence; otherwise 0.
NUM_PRE (GEN_PRE)	1 if a candidate which agrees with *ZERO* with regards to number (gender) is included in the set of *NP*; otherwise 0.
FIRST_SENT	1 if ZERO appears in the first sentence of a text; otherwise 0.
FIRST_WORD	1 if the predicate which has *ZERO* is the first word in a sentence; otherwise 0.
POS / LEMMA / DEP_LABEL	part-of-speech / dependency label / lemma of the predicate which has *ZERO*.
D_POS / D_LEMMA / D_DEP_LABEL	part-of-speech / dependency label / lemma of the dependents of the predicate which has *ZERO*.
PATH*	dependency labels (functional words) of words intervening between a *ZERO* and the sentence head

The features marked with '*' used only in Japanese.

Table 3: Features for anaphoricity determination

As features for Italian, we used lemmas, PoS tag of a predicate and its dependents as well as their morphological information (i.e. gender and number) automatically computed by TextPro (Pianta et al., 2008). For Japanese, the head lemmas of predicate and dependent chunks as well as the functional words involved with these two chunks were used as features. One case specially treated is when a dependent is placed as an adnominal constituent of a predicate, as in this case relation estimation of dependency arcs is difficult. In such case we instead use the features shown in Table 2 for accurate estimation.

4.5 Results with zero anaphora only

In zero anaphora resolution, we need to find all predicates that have *anaphoric* unrealized subjects (i.e. zero pronouns which have an antecedent in a text), and then identify an antecedent for each such argument.

The Italian and Japanese test data sets contain 4,065 and 25,467 verbal predicates respectively. The performance of each model at zero-anaphora detection and resolution is shown in Table 4, using recall

[8] http://www.cs.utah.edu/~hal/megam/

[9] http://www.di.unito.it/~tutreeb/

[10] Note that Iida et al. (2007b) referred to this relation as 'nominative'.

feature	description
HEAD_LEMMA	characters of the head lemma in *NP*.
POS	part-of-speech of *NP*.
DEFINITE	1 if *NP* contains the article corresponding to DEFINITE 'the'; otherwise 0.
DEMONSTRATIVE	1 if *NP* contains the article corresponding to DEMONSTRATIVE such as 'that' and 'this'; otherwise 0.
POSSESSIVE	1 if *NP* contains the article corresponding to POSSESSIVE such as 'his' and 'their'; otherwise 0.
CASE_MARKER**	case marker followed by *NP*, such as '*wa* (topic)', '*ga* (subject)', '*o* (object)'.
DEP_LABEL*	dependency label of *NP*.
COOC_MI**	the score of well-formedness model estimated from a large number of triplets ⟨ *NP*, Case, Predicate⟩.
FIRST_SENT	1 if *NP* appears in the first sentence of a text; otherwise 0.
FIRST_MENTION	1 if *NP* first appears in the set of candidate antecedents; otherwise 0.
CL_RANK**	a rank of *NP* in forward looking-center list based on Centering Theory (Grosz et al., 1995)
CL_ORDER**	a order of *NP* in forward looking-center list based on Centering Theory (Grosz et al., 1995)
PATH	dependency labels (functional words) of words intervening between a *ZERO* and *NP*
NUM_(DIS)AGREE	1 if *NP* (dis)agrees with *ZERO* with regards to number; otherwise 0.
GEN_(DIS)AGREE	1 if *NP* (dis)agrees with *ZERO* with regards to gender; otherwise 0.
HEAD_MATCH	1 if *ANA* and *NP* have the same head lemma; otherwise 0.
REGEX_MATCH	1 if the string of NP subsumes the string of ANA; otherwise 0.
COMP_MATCH	1 if *ANA* and *NP* have the same string; otherwise 0.

NP, *ANA* and *ZERO* stand for a candidate antecedent, a candidate anaphor and a candidate zero pronoun respectively. The features marked with '*' are only used in Italian, while the features marked with '**' are only used in Japanese.

Table 2: Features used for antecedent identification

	Italian						Japanese		
	system mentions			gold mentions					
model	R	P	F	R	P	F	R	P	F
PAIRWISE	0.864	0.172	0.287	0.864	0.172	0.287	0.286	0.308	0.296
DS-CASCADE	0.396	0.684	0.502	0.404	0.697	0.511	0.345	0.194	0.248
ILP	0.905	0.034	0.065	0.929	0.028	0.055	0.379	0.238	0.293
ILP +BF	0.803	0.375	0.511	0.834	0.369	0.511	0.353	0.256	0.297
ILP +SUBJ	0.900	0.034	0.066	0.927	0.028	0.055	0.371	0.315	0.341
ILP +BF +SUBJ	0.777	0.398	0.526	0.815	0.398	**0.534**	0.345	0.348	**0.346**

Table 4: Results on zero pronouns

/ precision / F over link detection as a metric (model theoretic metrics do not apply for this task as only subsets of coreference chains are considered). As can be seen from Table 4, the ILP version with Do-Not-Resolve-Non-Anaphors performs no better than the baselines for either languages, but in both languages replacing that constraint with Best-First results in a performance above the baselines; adding Subject Detection results in further improvement for both languages. Notice also that the performance of the models on Italian is quite a bit higher than for Japanese although the dataset is much smaller, possibly meaning that the task is easier in Italian.

5 Experiment 2: coreference resolution for all anaphors

In a second series of experiments we evaluated the performance of our models together with a full coreference system resolving all anaphors, not just zeros.

5.1 Separating vs combining classifiers

Different types of nominal expressions display very different anaphoric behavior: e.g., pronoun resolution involves very different types of information from nominal expression resolution, depending more on syntactic information and on the local context and less on commonsense knowledge. But the most common approach to coreference resolu-

tion (Soon et al., 2001; Ng and Cardie, 2002, etc.) is to use a single classifier to identify antecedents of all anaphoric expressions, relying on the ability of the machine learning algorithm to learn these differences. These models, however, often fail to capture the differences in anaphoric behavior between different types of expressions–one of the reasons being that the amount of training instances is often too small to learn such differences.[11] Using different models would appear to be key in the case of zero-anaphora resolution, which differs even more from the rest of anaphora resolution, e.g., in being particularly sensitive to local salience, as amply discussed in the literature on Centering discussed earlier.

To test the hypothesis that using what we will call *separated models* for zero anaphora and everything else would work better than *combined models* induced from all the learning instances, we manually split the training instances in terms of these two anaphora types and then created two classifiers for antecedent identification: one for zero-anaphora, the other for NP-anaphora, separately induced from the corresponding training instances. Likewise, anaphoricity determination models were separately induced with regards to these two anaphora types.

5.2 Results with all anaphors

In Table 5 and Table 6 we show the (MUC scorer) results obtained by adding the zero anaphoric resolution models proposed in this paper to both a combined and a separated classifier. For the separated classifier, we use the ILP+BF model for explicitly realized NPs, and different ILP models for zeros.

The results show that the separated classifier works systematically better than a combined classifier. For both Italian and Japanese the ILP+BF+SUBJ model works clearly better than the baselines, whereas simply applying the original Denis and Baldridge model unchanged to this case we obtain worse results than the baselines. For Italian we could also compare our results with those obtained on the same dataset by one of the two systems that participated to the Italian section of SEMEVAL, I-BART. I-BART's results are clearly better than those with both baselines, but also clearly inferior to the results obtained with our models. In particular, the effect of introducing the separated model with ILP+BF+SUBJ is more significant when using the system detected mentions; it obtained performance more than 13 points better than I-BART when the model referred to the system detected mentions.

[11] E.g., the entire MUC-6 corpus contains a grand total of 3 reflexive pronouns.

	Japanese					
	combined			separated		
model	R	P	F	R	P	F
PAIRWISE	0.345	0.236	0.280	0.427	0.240	0.308
DS-CASCADE	0.207	0.592	0.307	0.291	0.488	0.365
ILP	0.381	0.330	0.353	0.490	0.304	0.375
ILP +BF	0.349	0.390	0.368	0.446	0.340	0.386
ILP +SUBJ	0.376	0.366	0.371	0.484	0.353	0.408
ILP +BF +SUBJ	0.344	0.450	0.390	0.441	0.415	**0.427**

Table 6: Results for overall coreference: Japanese (MUC score)

6 Related work

We are not aware of any previous machine learning model for zero anaphora in Italian, but there has been quite a lot of work on Japanese zero-anaphora (Iida et al., 2007a; Taira et al., 2008; Imamura et al., 2009; Taira et al., 2010; Sasano et al., 2009). In work such as Taira et al. (2008) and Imamura et al. (2009), zero-anaphora resolution is considered as a sub-task of predicate argument structure analysis, taking the NAIST text corpus as a target data set. Taira et al. (2008) and Taira et al. (2010) applied decision lists and transformation-based learning respectively in order to manually analyze which clues are important for each argument assignment. Imamura et al. (2009) also tackled to the same problem setting by applying a pairwise classifier for each argument. In their approach, a 'null' argument is explicitly added into the set of candidate argument to learn the situation where an argument of a predicate is 'exophoric'. They reported their model achieved better performance than the work by Taira et al. (2008).

Iida et al. (2007a) also used the NAIST text corpus. They adopted the BACT learning algorithm (Kudo and Matsumoto, 2004) to effectively learn subtrees useful for both antecedent identification and zero pronoun detection. Their model drastically outperformed a simple pairwise model, but it is still performed as a cascaded process. Incorporating

	Italian											
	system mentions						gold mentions					
	combined			separated			combined			separated		
model	R	P	F	R	P	F	R	P	F	R	P	F
PAIRWISE	0.508	0.208	0.295	0.472	0.241	0.319	0.582	0.261	0.361	0.566	0.314	0.404
DS-CASCADE	0.225	0.553	0.320	0.217	0.574	0.315	0.245	0.609	0.349	0.246	0.686	0.362
I-BART	0.324	0.294	0.308	–	–	–	0.532	0.441	0.482	–	–	–
ILP	0.539	0.321	0.403	0.535	0.316	0.397	0.614	0.369	0.461	0.607	0.384	0.470
ILP +BF	0.471	0.404	0.435	0.483	0.409	0.443	0.545	0.517	0.530	0.563	0.519	0.540
ILP +SUBJ	0.537	0.325	0.405	0.534	0.318	0.399	0.611	0.372	0.463	0.606	0.387	0.473
ILP +BF +SUBJ	0.464	0.410	0.435	0.478	0.418	**0.446**	0.538	0.527	0.533	0.559	0.536	**0.547**

R: Recall, P: Precision, F: f-score, BF: best first constraint, SUBJ: subject detection model.

Table 5: Results for overall coreference: Italian (MUC score)

their model into the ILP formulation proposed here looks like a promising further extension.

Sasano et al. (2009) obtained interesting experimental results about the relationship between zero-anaphora resolution and the scale of automatically acquired case frames. In their work, their case frames were acquired from a very large corpus consisting of 100 billion words. They also proposed a probabilistic model to Japanese zero-anaphora in which an argument assignment score is estimated based on the automatically acquired case frames. They concluded that case frames acquired from larger corpora lead to better f-score on zero-anaphora resolution.

In contrast to these approaches in Japanese, the participants to Semeval 2010 task 1 (especially the Italian coreference task) simply solved the problems using one coreference classifier, not distinguishing zero-anaphora from the other types of anaphora (Kobdani and Schütze, 2010; Poesio et al., 2010). On the other hand, our approach shows separating problems contributes to improving performance in Italian zero-anaphora. Although we used gold mentions in our evaluations, mention detection is also essential. As a next step, we also need to take into account ways of incorporating a mention detection model into the ILP formulation.

7 Conclusion

In this paper, we developed a new ILP-based model of zero anaphora detection and resolution that extends the coreference resolution model proposed by Denis and Baldridge (2007) by introducing modified constraints and a subject detection model. We evaluated this model both individually and as part of the overall coreference task for both Italian and Japanese zero anaphora, obtaining clear improvements in performance.

One avenue for future research is motivated by the observation that whereas introducing the subject detection model and the best-first constraint results in higher precision maintaining the recall compared to the baselines, that precision is still low. One of the major source of the errors is that zero pronouns are frequently used in Italian and Japanese in contexts in which in English as so-called *generic they* would be used: "*I walked into the hotel and (they) said ..*". In such case, the zero pronoun detection model is often incorrect. We are considering adding a generic they detection component.

We also intend to experiment with introducing more sophisticated antecedent identification models in the ILP framework. In this paper, we used a very basic pairwise classifier; however Yang et al. (2008) and Iida et al. (2003) showed that the relative comparison of two candidate antecedents leads to obtaining better accuracy than the pairwise model. However, these approaches do not output absolute probabilities, but relative significance between two candidates, and therefore cannot be directly integrated with the ILP-framework. We plan to examine ways of appropriately estimating an absolute score from a set of relative scores for further refinement.

Finally, we would like to test our model with English constructions which closely resemble zero anaphora. One example were studied in the Semeval 2010 'Linking Events and their Participants in Discourse' task, which provides data about *null instan-*

tiation, omitted arguments of predicates like "*We arrived ϕ^{goal} at 8pm.*". (Unfortunately the dataset available for SEMEVAL was very small.) Another interesting area of application of these techniques would be VP ellipsis.

Acknowledgments

Ryu Iida's stay in Trento was supported by the Excellent Young Researcher Overseas Visit Program of the Japan Society for the Promotion of Science (JSPS). Massimo Poesio was supported in part by the Provincia di Trento Grande Progetto LiveMemories, which also funded the creation of the Italian corpus used in this study. We also wish to thank Francesca Delogu, Kepa Rodriguez, Olga Uryupina and Yannick Versley for much help with the corpus and BART.

References

G. Attardi, F. Dell'Orletta, M. Simi, A. Chanev, and M. Ciaramita. 2007. Multilingual dependency parsing and domain adaptation using desr. In *Proc. of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, Prague.

C. Bosco, S. Montemagni, A. Mazzei, V. Lombardo, F. Dell'Orletta, A. Lenci, L. Lesmo, G. Attardi, M. Simi, A. Lavelli, J. Hall, J. Nilsson, and J. Nivre. 2010. Comparing the influence of different treebank annotations on dependency parsing. In *Proceedings of LREC*, pages 1794–1801.

P. Denis and J. Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *Proc. of HLT/NAACL*, pages 236–243.

B. Di Eugenio. 1998. Centering in Italian. In M. A. Walker, A. K. Joshi, and E. F. Prince, editors, *Centering Theory in Discourse*, chapter 7, pages 115–138. Oxford.

B. J. Grosz, A. K. Joshi, and S. Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–226.

R. Iida, K. Inui, H. Takamura, and Y. Matsumoto. 2003. Incorporating contextual cues in trainable models for coreference resolution. In *Proceedings of the 10th EACL Workshop on The Computational Treatment of Anaphora*, pages 23–30.

R. Iida, K. Inui, and Y. Matsumoto. 2007a. Zero-anaphora resolution by learning rich syntactic pattern features. *ACM Transactions on Asian Language Information Processing* (*TALIP*), 6(4).

R. Iida, M. Komachi, K. Inui, and Y. Matsumoto. 2007b. Annotating a Japanese text corpus with predicate-argument and coreference relations. In *Proceeding of the ACL Workshop 'Linguistic Annotation Workshop'*, pages 132–139.

K. Imamura, K. Saito, and T. Izumi. 2009. Discriminative approach to predicate-argument structure analysis with zero-anaphora resolution. In *Proceedings of ACL-IJCNLP, Short Papers*, pages 85–88.

H. Isozaki and T. Hirao. 2003. Japanese zero pronoun resolution based on ranking rules and machine learning. In *Proceedings of EMNLP*, pages 184–191.

M. Kameyama. 1985. *Zero Anaphora: The case of Japanese*. Ph.D. thesis, Stanford University.

H. Kobdani and H. Schütze. 2010. Sucre: A modular system for coreference resolution. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 92–95.

T. Kudo and Y. Matsumoto. 2004. A boosting algorithm for classification of semi-structured text. In *Proceedings of EMNLP*, pages 301–308.

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th ACL*, pages 104–111.

E. Pianta, C. Girardi, and R. Zanoli. 2008. The TextPro tool suite. In *In Proceedings of LREC*, pages 28–30.

M. Poesio, O. Uryupina, and Y. Versley. 2010. Creating a coreference resolution system for Italian. In *Proceedings of LREC*.

M. Recasens, L. Màrquez, E. Sapena, M. A. Martí, M. Taulé, V. Hoste, M. Poesio, and Y. Versley. 2010. Semeval-2010 task 1: Coreference resolution in multiple languages. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 1–8.

K-J. Rodriguez, F. Delogu, Y. Versley, E. Stemle, and M. Poesio. 2010. Anaphoric annotation of wikipedia and blogs in the live memories corpus. In *Proc. LREC*.

D. Roth and W.-T. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. of CONLL*.

R. Sasano, D. Kawahara, and S. Kurohashi. 2009. The effect of corpus size on case frame acquisition for discourse analysis. In *Proceedings of HLT/NAACL*, pages 521–529.

K. Seki, A. Fujii, and T. Ishikawa. 2002. A probabilistic method for analyzing Japanese anaphora integrating zero pronoun detection and resolution. In *Proceedings of the 19th COLING*, pages 911–917.

W. M. Soon, H. T. Ng, and D. C. Y. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

H. Taira, S. Fujita, and M. Nagata. 2008. A Japanese predicate argument structure analysis using decision lists. In *Proceedings of EMNLP*, pages 523–532.

H. Taira, S. Fujita, and M. Nagata. 2010. Predicate argument structure analysis using transformation based learning. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 162–167.

M. A. Walker, M. Iida, and S. Cote. 1994. Japanese discourse and the process of centering. *Computational Linguistics*, 20(2):193–232.

X. Yang, J. Su, and C. L. Tan. 2008. Twin-candidate model for learning-based anaphora resolution. *Computational Linguistics*, 34(3):327–356.

Coreference Resolution with World Knowledge

Altaf Rahman and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{altaf,vince}@hlt.utdallas.edu

Abstract

While world knowledge has been shown to improve learning-based coreference resolvers, the improvements were typically obtained by incorporating world knowledge into a fairly weak baseline resolver. Hence, it is not clear whether these benefits can carry over to a stronger baseline. Moreover, since there has been no attempt to apply different sources of world knowledge in combination to coreference resolution, it is not clear whether they offer complementary benefits to a resolver. We systematically compare commonly-used and under-investigated sources of world knowledge for coreference resolution by applying them to two learning-based coreference models and evaluating them on documents annotated with two different annotation schemes.

1 Introduction

Noun phrase (NP) coreference resolution is the task of determining which NPs in a text or dialogue refer to the same real-world entity. The difficulty of the task stems in part from its reliance on world knowledge (Charniak, 1972). To exemplify, consider the following text fragment.

Martha Stewart is hoping people don't run out on her. The celebrity indicted on charges stemming from . . .

Having the (world) knowledge that *Martha Stewart* is a *celebrity* would be helpful for establishing the coreference relation between the two NPs. One may argue that employing heuristics such as subject preference or syntactic parallelism (which prefers resolving an NP to a candidate antecedent that has the same grammatical role) in this example would also allow us to correctly resolve *the celebrity* (Mitkov, 2002), thereby obviating the need for world knowledge. However, since these heuristics are not perfect, complementing them with world knowledge would be an important step towards bringing coreference systems to the next level of performance.

Despite the usefulness of world knowledge for coreference resolution, early learning-based coreference resolvers have relied mostly on morpho-syntactic features (e.g., Soon et al. (2001), Ng and Cardie (2002), Yang et al. (2003)). With recent advances in lexical semantics research and the development of large-scale knowledge bases, researchers have begun to employ world knowledge for coreference resolution. World knowledge is extracted primarily from three data sources, web-based encyclopedia (e.g., Ponzetto and Strube (2006), Uryupina et al. (2011)), unannotated data (e.g., Daumé III and Marcu (2005), Ng (2007)), and coreference-annotated data (e.g., Bengtson and Roth (2008)).

While each of these three sources of world knowledge has been shown to improve coreference resolution, the improvements were typically obtained by incorporating world knowledge (as features) into a baseline resolver composed of a rather weak coreference model (i.e., the mention-pair model) and a small set of features (i.e., the 12 features adopted by Soon et al.'s (2001) knowledge-lean approach). As a result, some questions naturally arise. First, can world knowledge still offer benefits when used in combination with a richer set of features? Second, since automatically extracted world knowledge is typically noisy (Ponzetto and Poesio, 2009), are recently-developed coreference models more noise-tolerant than the mention-pair model, and if so, can they profit more from the noisily extracted world knowledge? Finally, while different world knowl-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 814–824,
Portland, Oregon, June 19-24, 2011.

edge sources have been shown to be useful when applied in isolation to a coreference system, do they offer complementary benefits and therefore can further improve a resolver when applied in combination?

We seek answers to these questions by conducting a systematic evaluation of different world knowledge sources for learning-based coreference resolution. Specifically, we (1) derive world knowledge from encyclopedic sources that are under-investigated for coreference resolution, including FrameNet (Baker et al., 1998) and YAGO (Suchanek et al., 2007), in addition to coreference-annotated data and unannotated data; (2) incorporate such knowledge as features into a richer baseline feature set that we previously employed (Rahman and Ng, 2009); and (3) evaluate their utility using two coreference models, the traditional mention-pair model (Soon et al., 2001) and the recently developed cluster-ranking model (Rahman and Ng, 2009).

Our evaluation corpus contains 410 documents, which are coreference-annotated using the ACE annotation scheme as well as the OntoNotes annotation scheme (Hovy et al., 2006). By evaluating on two sets of coreference annotations for the same set of documents, we can determine whether the usefulness of world knowledge sources for coreference resolution is dependent on the underlying annotation scheme used to annotate the documents.

2 Preliminaries

In this section, we describe the corpus, the NP extraction methods, the coreference models, and the evaluation measures we will use in our evaluation.

2.1 Data Set

We evaluate on documents that are coreference-annotated using both the ACE annotation scheme and the OntoNotes annotation scheme, so that we can examine whether the usefulness of our world knowledge sources is dependent on the underlying coreference annotation scheme. Specifically, our data set is composed of the 410 English newswire articles that appear in both OntoNotes-2 and ACE 2004/2005. We partition the documents into a training set and a test set following a 80/20 ratio.

ACE and OntoNotes employ different guidelines to annotate coreference chains. A major difference between the two annotation schemes is that ACE only concerns establishing coreference chains among NPs that belong to the ACE entity types, whereas OntoNotes does not have this restriction. Hence, the OntoNotes annotation scheme should produce more coreference chains (i.e., non-singleton coreference clusters) than the ACE annotation scheme for a given set of documents. For our data set, the OntoNotes scheme yielded 4500 chains, whereas the ACE scheme yielded only 3637 chains.

Another difference between the two annotation schemes is that singleton clusters are annotated in ACE but not OntoNotes. As discussed below, the presence of singleton clusters may have an impact on NP extraction and coreference evaluation.

2.2 NP Extraction

Following common practice, we employ different methods to extract NPs from the documents annotated with the two annotation schemes.

To extract NPs from the ACE-annotated documents, we train a mention extractor on the training texts (see Section 5.1 of Rahman and Ng (2009) for details), which recalls 83.6% of the NPs in the test set. On the other hand, to extract NPs from the OntoNotes-annotated documents, the same method should not be applied. To see the reason, recall that only the NPs in non-singleton clusters are annotated in these documents. Training a mention extractor on these NPs implies that we are learning to extract *non-singleton NPs*, which are typically much smaller in number than the entire set of NPs. In other words, doing so could substantially simplify the coreference task. Consequently, we follow the approach adopted by traditional learning-based resolvers and employ an NP chunker to extract NPs. Specifically, we use the markable identification system in the Reconcile resolver (Stoyanov et al., 2010) to extract NPs from the training and test texts. This identifier recalls 77.4% of the NPs in the test set.

2.3 Coreference Models

We evaluate the utility of world knowledge using the mention-pair model and the cluster-ranking model.

2.3.1 Mention-Pair Model

The mention-pair (MP) model is a classifier that determines whether two NPs are coreferent or not.

Each instance $\boldsymbol{i}(\text{NP}_j, \text{NP}_k)$ corresponds to NP_j and NP_k, and is represented by a Baseline feature set consisting of 39 features. Linguistically, these features can be divided into four categories: string-matching, grammatical, semantic, and positional. These features can also be categorized based on whether they are relational or not. Relational features capture the relationship between NP_j and NP_k, whereas non-relational features capture the linguistic property of one of these two NPs. Since space limitations preclude a description of these features, we refer the reader to Rahman and Ng (2009) for details.

We follow Soon et al.'s (2001) method for creating training instances: we create (1) a positive instance for each anaphoric NP, NP_k, and its closest antecedent, NP_j; and (2) a negative instance for NP_k paired with each of the intervening NPs, NP_{j+1}, NP_{j+2}, ..., NP_{k-1}. The classification of a training instance is either positive or negative, depending on whether the two NPs are coreferent in the associated text. To train the MP model, we use the SVM learning algorithm from SVM^{light} (Joachims, 2002).[1]

After training, the classifier is used to identify an antecedent for an NP in a test text. Specifically, each NP, NP_k, is compared in turn to each preceding NP, NP_j, from right to left, and NP_j is selected as its antecedent if the pair is classified as coreferent. The process terminates as soon as an antecedent is found for NP_k or the beginning of the text is reached.

Despite its popularity, the MP model has two major weaknesses. First, since each candidate antecedent for an NP to be resolved (henceforth an *active NP*) is considered independently of the others, this model only determines how good a candidate antecedent is relative to the active NP, but not how good a candidate antecedent is relative to other candidates. So, it fails to answer the critical question of which candidate antecedent is most probable. Second, it has limitations in its expressiveness: the information extracted from the two NPs alone may not be sufficient for making a coreference decision.

2.3.2 Cluster-Ranking Model

The cluster-ranking (CR) model addresses the two weaknesses of the MP model by combining the strengths of the *entity-mention* model (e.g., Luo et al. (2004), Yang et al. (2008)) and the *mention-ranking* model (e.g., Denis and Baldridge (2008)). Specifically, the CR model ranks the preceding clusters for an active NP so that the highest-ranked cluster is the one to which the active NP should be linked. Employing a ranker addresses the first weakness, as a ranker allows all candidates to be compared *simultaneously*. Considering preceding clusters rather than antecedents as candidates addresses the second weakness, as *cluster-level* features (i.e., features that are defined over any subset of NPs in a preceding cluster) can be employed. Details of the CR model can be found in Rahman and Ng (2009).

Since the CR model ranks preceding clusters, a training instance $\boldsymbol{i}(c_j, \text{NP}_k)$ represents a preceding cluster, c_j, and an anaphoric NP, NP_k. Each instance consists of features that are computed based solely on NP_k as well as cluster-level features, which describe the relationship between c_j and NP_k. Motivated in part by Culotta et al. (2007), we create cluster-level features from the *relational* features in our feature set using four predicates: NONE, MOST-FALSE, MOST-TRUE, and ALL. Specifically, for each relational feature X, we first convert X into an equivalent set of binary-valued features if it is multi-valued. Then, for each resulting binary-valued feature X_b, we create four binary-valued cluster-level features: (1) NONE-X_b is true when X_b is false between NP_k and each NP in c_j; (2) MOST-FALSE-X_b is true when X_b is true between NP_k and less than half (but at least one) of the NPs in c_j; (3) MOST-TRUE-X_b is true when X_b is true between NP_k and at least half (but not all) of the NPs in c_j; and (4) ALL-X_b is true when X_b is true between NP_k and each NP in c_j.

We train a cluster ranker to jointly learn anaphoricity determination and coreference resolution using SVM^{light}'s ranker-learning algorithm. Specifically, for each NP, NP_k, we create a training instance between NP_k and *each* preceding cluster c_j using the features described above. Since we are learning a joint model, we need to provide the ranker with the option to start a new cluster by creating an additional training instance that contains the non-relational features describing NP_k. The rank value of a training instance $\boldsymbol{i}(c_j, \text{NP}_k)$ created for NP_k is the rank of c_j among the competing clusters. If NP_k is anaphoric, its rank is HIGH if NP_k belongs to c_j, and LOW otherwise. If NP_k is non-anaphoric, its rank is

[1] For this and subsequent uses of the SVM learner in our experiments, we set all parameters to their default values.

LOW unless it is the additional training instance described above, which has rank HIGH.

After training, the cluster ranker processes the NPs in a test text in a left-to-right manner. For each active NP, NP_k, we create test instances for it by pairing it with each of its preceding clusters. To allow for the possibility that NP_k is non-anaphoric, we create an additional test instance as during training. All these test instances are then presented to the ranker. If the additional test instance is assigned the highest rank value, then we create a new cluster containing NP_k. Otherwise, NP_k is linked to the cluster that has the highest rank. Note that the partial clusters preceding NP_k are formed incrementally based on the predictions of the ranker for the first $k-1$ NPs.

2.4 Evaluation Measures

We employ two commonly-used scoring programs, B^3 (Bagga and Baldwin, 1998) and CEAF (Luo, 2005), both of which report results in terms of recall (R), precision (P), and F-measure (F) by comparing the gold-standard (i.e., key) partition, KP, against the system-generated (i.e., response) partition, RP.

Briefly, B^3 computes the R and P values of each NP and averages these values at the end. Specifically, for each NP, NP_j, B^3 first computes the number of common NPs in KP_j and RP_j, the clusters containing NP_j in KP and RP, respectively, and then divides this number by $|KP_j|$ and $|RP_j|$ to obtain the R and P values of NP_j, respectively. On the other hand, CEAF finds the best one-to-one alignment between the key clusters and the response clusters.

A complication arises when B^3 is used to score a response partition containing automatically extracted NPs. Recall that B^3 constructs a mapping between the NPs in the response and those in the key. Hence, if the response is generated using gold-standard NPs, then every NP in the response is mapped to some NP in the key and vice versa. In other words, there are no *twinless* (i.e., unmapped) NPs (Stoyanov et al., 2009). This is not the case when automatically extracted NPs are used, but the original description of B^3 does not specify how twinless NPs should be scored (Bagga and Baldwin, 1998). To address this problem, we set the recall and precision of a twinless NP to zero, regardless of whether the NP appears in the key or the response. Note that CEAF can compare partitions with twinless NPs without any modification, since it operates by finding the best alignment between the clusters in the two partitions.

Additionally, in order not to over-penalize a response partition, we remove all the twinless NPs in the response that are singletons. The rationale is simple: since the resolver has successfully identified these NPs as singletons, it should not be penalized, and removing them avoids such penalty.

Since B^3 and CEAF align NPs/clusters, the lack of singleton clusters in the OntoNotes annotations implies that the resulting scores reflect solely how well a resolver identifies coreference links and do not take into account how well it identifies singleton clusters.

3 Extracting World Knowledge

In this section, we describe how we extract world knowledge for coreference resolution from three different sources: large-scale knowledge bases, coreference-annotated data and unannotated data.

3.1 World Knowledge from Knowledge Bases

We extract world knowledge from two large-scale knowledge bases, YAGO and FrameNet.

3.1.1 Extracting Knowledge from YAGO

We choose to employ YAGO rather than the more popularly-used Wikipedia due to its potentially richer knowledge, which comprises 5 million facts extracted from Wikipedia and WordNet. Each fact is represented as a triple (NP_j, rel, NP_k), where rel is one of the 90 YAGO relation types defined on two NPs, NP_j and NP_k. Motivated in part by previous work (Bryl et al., 2010; Uryupina et al., 2011), we employ the two relation types that we believe are most useful for coreference resolution, TYPE and MEANS. TYPE is essentially an IS-A relation. For instance, the triple (`AlbertEinstein`, TYPE, `physicist`) denotes the fact that *Albert Einstein* is a physicist. MEANS provides different ways of expressing an entity, and therefore allows us to deal with synonymy and ambiguity. For instance, the two triples (`Einstein`, MEANS, `AlbertEinstein`) and (`Einstein`, MEANS, `AlfredEinstein`) denote the facts that *Einstein* may refer to the physicist *Albert Einstein* and the musicologist *Alfred Einstein*, respectively. Hence, the presence of one or

both of these relations between two NPs provides strong evidence that the two NPs are coreferent.

YAGO's unification of the information in Wikipedia and WordNet enables it to extract facts that cannot be extracted with Wikipedia or WordNet alone, such as (`MarthaStewart`, TYPE, `celebrity`). To better appreciate YAGO's strengths, let us see how this fact was extracted. YAGO first heuristically maps each of the Wiki categories in the Wiki page for *Martha Stewart* to its semantically closest WordNet synset. For instance, the Wiki category AMERICAN TELEVISION PERSONALITIES is mapped to the synset corresponding to sense #2 of the word *personality*. Then, given that *personality* is a direct hyponym of *celebrity* in WordNet, YAGO extracts the desired fact. This enables YAGO to extract facts that cannot be extracted with Wikipedia or WordNet alone.

We incorporate the world knowledge from YAGO into our coreference models as a binary-valued feature. If the MP model is used, the YAGO feature for an instance will have the value 1 if and only if the two NPs involved are in a TYPE or MEANS relation. On the other hand, if the CR model is used, the YAGO feature for an instance involving NP_k and preceding cluster c will have the value 1 if and only if NP_k has a TYPE or MEANS relation with any of the NPs in c. Since knowledge extraction from web-based encyclopedia is typically noisy (Ponzetto and Poesio, 2009), we use YAGO to determine whether two NPs have a relation only if one NP is a named entity (NE) of type person, organization, or location according to the Stanford NE recognizer (Finkel et al., 2005) and the other NP is a common noun.

3.1.2 Extracting Knowledge from FrameNet

FrameNet is a lexico-semantic resource focused on semantic frames (Baker et al., 1998). As a schematic representation of a situation, a frame contains the *lexical predicates* that can invoke it as well as the *frame elements* (i.e., semantic roles). For example, the JUDGMENT_COMMUNICATION frame describes situations in which a COMMUNICATOR communicates a judgment of an EVALUEE to an ADDRESSEE. This frame has COMMUNICATOR and EVALUEE as its core frame elements and ADDRESSEE as its non-core frame elements, and can be invoked by more than 40 predicates, such as *acclaim*, *accuse*, *commend*, *decry*, *denounce*, *praise*, and *slam*.

To better understand why FrameNet contains potentially useful knowledge for coreference resolution, consider the following text segment:

Peter Anthony decries program trading as "limiting the game to a few," but he is not sure whether he wants to denounce it because ...

To establish the coreference relation between *it* and *program trading*, it may be helpful to know that *decry* and *denounce* appear in the same frame and the two NPs have the same semantic role.

This example suggests that features encoding both the semantic roles of the two NPs under consideration and whether the associated predicates are "related" to each other in FrameNet (i.e., whether they appear in the same frame) could be useful for identifying coreference relations. Two points regarding our implementation of these features deserve mention. First, since we do not employ verb sense disambiguation, we consider two predicates *related* as long as there is at least one semantic frame in which they both appear. Second, since FrameNet-style semantic role labelers are not publicly available, we use ASSERT (Pradhan et al., 2004), a semantic role labeler that provides PropBank-style semantic roles such as ARG0 (the PROTOAGENT, which is typically the subject of a transitive verb) and ARG1 (the PROTOPATIENT, which is typically its direct object).

Now, assuming that NP_j and NP_k are the arguments of two stemmed predicates, $pred_j$ and $pred_k$, we create 15 features using the knowledge extracted from FrameNet and ASSERT as follows. First, we encode the knowledge extracted from FrameNet as one of three possible values: (1) $pred_j$ and $pred_k$ are in the same frame; (2) they are both predicates in FrameNet but never appear in the same frame; and (3) one or both predicates do not appear in FrameNet. Second, we encode the semantic roles of NP_j and NP_k as one of five possible values: ARG0-ARG0, ARG1-ARG1, ARG0-ARG1, ARG1-ARG0, and OTHERS (the default case).[2] Finally, we create 15 binary-valued features by pairing the 3 possible values extracted from FrameNet and the 5 possible values provided by ASSERT. Since these features

[2] We focus primarily on ARG0 and ARG1 because they are the most important core arguments of a predicate and may provide more useful information than other semantic roles.

are computed over two NPs, we can employ them directly for the MP model. Note that by construction, exactly one of these features will have a non-zero value. For the CR model, we extend their definitions so that they can be computed between an NP, NP_k, and a preceding cluster, c. Specifically, the value of a feature is 1 if and only if its value between NP_k and one of the NPs in c is 1 under its original definition.

The above discussion assumes that the two NPs under consideration serve as predicate arguments. If this assumption fails, we will not create any features based on FrameNet for these two NPs.

To our knowledge, FrameNet has not been exploited for coreference resolution. However, the use of related verbs is similar in spirit to Bean and Riloff's (2004) use of patterns for inducing contextual role knowledge, and the use of semantic roles is also discussed in Ponzetto and Strube (2006).

3.2 World Knowledge from Annotated Data

Since world knowledge is needed for coreference resolution, a human annotator must have employed world knowledge when coreference-annotating a document. We aim to design features that can "recover" such world knowledge from annotated data.

3.2.1 Features Based on Noun Pairs

A natural question is: what kind of world knowledge can we extract from annotated data? We may gather the knowledge that *Barack Obama* is a *U.S. president* if we see these two NPs appearing in the same coreference chain. Equally importantly, we may gather the commonsense knowledge needed for determining *non-coreference*. For instance, we may discover that a *lion* and a *tiger* are unlikely to refer to the same real-world entity after realizing that they never appear in the same chain in a large number of annotated documents. Note that any features computed based on WordNet distance or distributional similarity are likely to incorrectly suggest that *lion* and *tiger* are coreferent, since the two nouns are similar distributionally and according to WordNet.

Given these observations, one may collect the noun pairs from the (coreference-annotated) training data and use them as features to train a resolver. However, for these features to be effective, we need to address *data sparseness*, as many noun pairs in the training data may not appear in the test data.

To improve generalization, we instead create different kinds of *noun-pair-based* features given an annotated text. To begin with, we preprocess each document. A *training* text is preprocessed by randomly replacing 10% of its common nouns with the label UNSEEN. If an NP, NP_k, is replaced with UNSEEN, all NPs that have the same string as NP_k will also be replaced with UNSEEN. A *test* text is preprocessed differently: we simply replace all NPs whose strings are not seen in the training data with UNSEEN. Hence, artificially creating UNSEEN labels from a training text will allow a learner to learn how to handle unseen words in a test text.

Next, we create *noun-pair-based features* for the MP model, which will be used to augment the Baseline feature set. Here, each instance corresponds to two NPs, NP_j and NP_k, and is represented by three groups of *binary-valued* features.

Unseen features are applicable when both NP_j and NP_k are UNSEEN. Either an UNSEEN-SAME feature or an UNSEEN-DIFF feature is created, depending on whether the two NPs are the same string before being replaced with the UNSEEN token.

Lexical features are applicable when neither NP_j nor NP_k is UNSEEN. A lexical feature is an ordered pair consisting of the heads of the NPs. For a pronoun or a common noun, the head is the last word of the NP; for a proper name, the head is the entire NP.

Semi-lexical features aim to improve generalization, and are applicable when neither NP_j nor NP_k is UNSEEN. If exactly one of NP_j and NP_k is tagged as a NE by the Stanford NE recognizer, we create a semi-lexical feature that is identical to the lexical feature described above, except that the NE is replaced with its NE label. On the other hand, if both NPs are NEs, we check whether they are the same string. If so, we create a *NE*-SAME feature, where *NE* is replaced with the corresponding NE label. Otherwise, we check whether they have the same NE tag *and* a word-subset match (i.e., whether the word tokens in one NP appears in the other's list of word tokens). If so, we create a *NE*-SUBSAME feature, where *NE* is replaced with their NE label. Otherwise, we create a feature that is the concatenation of the NE labels of the two NPs.

The noun-pair-based features for the CR model can be generated using essentially the same method. Specifically, since each instance now corresponds to

an NP, NP_k, and a preceding cluster, c, we can generate a noun-pair-based feature by applying the above method to NP_k and each of the NPs in c, and its value is the number of times it is applicable to NP_k and c.

3.2.2 Features Based on Verb Pairs

As discussed above, features encoding the semantic roles of two NPs and the relatedness of the associated verbs could be useful for coreference resolution. Rather than encoding verb relatedness, we may replace verb relatedness with the verbs themselves in these features, and have the learner learn directly from coreference-annotated data whether two NPs serving as the objects of *decry* and *denounce* are likely to be coreferent or not, for instance.

Specifically, assuming that NP_j and NP_k are the arguments of two stemmed predicates, $pred_j$ and $pred_k$, in the training data, we create five features as follows. First, we encode the semantic roles of NP_j and NP_k as one of five possible values: ARG0-ARG0, ARG1-ARG1, ARG0-ARG1, ARG1-ARG0, and OTHERS (the default case). Second, we create five binary-valued features by pairing each of these five values with the two stemmed predicates. Since these features are computed over two NPs, we can employ them directly for the MP model. Note that by construction, exactly one of these features will have a non-zero value. For the CR model, we extend their definitions so that they can be computed between an NP, NP_k, and a preceding cluster, c. Specifically, the value of a feature is 1 if and only if its value between NP_k and one of the NPs in c is 1 under its original definition.

The above discussion assumes that the two NPs under consideration serve as predicate arguments. If this assumption fails, we will not create any features based on verb pairs for these two NPs.

3.3 World Knowledge from Unannotated Data

Previous work has shown that syntactic appositions, which can be extracted using heuristics from unannotated documents or parse trees, are a useful source of world knowledge for coreference resolution (e.g., Daumé III and Marcu (2005), Ng (2007), Haghighi and Klein (2009)). Each extraction is an NP pair such as <*Barack Obama*, *the president*> and <*Eastern Airlines*, *the carrier*>, where the first NP in the pair is a proper name and the second NP is a common NP. Low-frequency extractions are typically assumed to be noisy and discarded.

We combine the extractions produced by Fleischman et al. (2003) and Ng (2007) to form a database consisting of 1.057 million NP pairs, and create a binary-valued feature for our coreference models using this database. If the MP model is used, this feature will have the value 1 if and only if the two NPs appear as a pair in the database. On the other hand, if the CR model is used, the feature for an instance involving NP_k and preceding cluster c will have the value 1 if and only if NP_k and at least one of the NPs in c appears as a pair in the database.

4 Evaluation

4.1 Experimental Setup

As described in Section 2, we use as our evaluation corpus the 411 documents that are coreference-annotated using the ACE and OntoNotes annotation schemes. Specifically, we divide these documents into five (disjoint) folds of roughly the same size, training the MP model and the CR model using SVM^{light} on four folds and evaluate their performance on the remaining fold. The linguistic features, as well as the NPs used to create the training and test instances, are computed automatically. We employ B^3 and CEAF as described in Section 2.3 to score the output of a coreference system.

4.2 Results and Discussion

4.2.1 Baseline Models

Since our goal is to evaluate the effectiveness of the features encoding world knowledge for learning-based coreference resolution, we employ as our baselines the MR model and the CR model trained on the Baseline feature set, which does not contain any features encoding world knowledge. For the MP model, the Baseline feature set consists of the 39 features described in Section 2.3.1; for the CR model, the Baseline feature set consists of the cluster-level features derived from the 39 features used in the Baseline MP model (see Section 2.3.2).

Results of the MP model and the CR model employing the Baseline feature set are shown in rows 1 and 8 of Table 1, respectively. Each row contains the B^3 and CEAF results of the corresponding coreference model when it is evaluated using the ACE and

		ACE						OntoNotes					
		B^3			CEAF			B^3			CEAF		
	Feature Set	R	P	F	R	P	F	R	P	F	R	P	F
	Results for the Mention-Pair Model												
1	Base	56.5	69.7	62.4	54.9	66.3	60.0	50.4	56.7	53.3	48.9	54.5	51.5
2	Base+YAGO Types (YT)	57.3	70.3	63.1	58.7	67.5	**62.8**	51.7	57.9	**54.6**	50.3	55.6	**52.8**
3	Base+YAGO Means (YM)	56.7	70.0	62.7	55.3	66.5	60.4	50.6	57.0	53.6	49.3	54.9	51.9
4	Base+Noun Pairs (WP)	57.5	70.6	**63.4**	55.8	67.4	61.1	51.6	57.6	54.4	49.7	55.4	52.4
5	Base+FrameNet (FN)	56.4	70.9	62.8	54.9	67.5	60.5	50.5	57.5	53.8	48.8	55.1	51.8
6	Base+Verb Pairs (VP)	56.9	71.3	63.3	55.2	67.6	60.8	50.7	57.9	54.0	49.0	55.4	52.0
7	Base+Appositives (AP)	56.9	70.0	62.7	55.6	66.9	60.7	50.3	57.1	53.5	49.1	55.1	51.9
	Results for the Cluster-Ranking Model												
8	Base	61.7	71.2	66.1	59.6	68.8	63.8	53.4	59.2	56.2	51.1	57.3	54.0
9	Base+YAGO Types (YT)	63.5	72.4	67.6	61.7	70.0	**65.5**	54.8	60.6	57.6	52.4	58.9	55.4
10	Base+YAGO Means (YM)	62.0	71.4	66.4	59.9	69.1	64.1	53.9	59.5	56.6	51.4	57.5	54.3
11	Base+Noun Pairs (WP)	64.1	73.4	**68.4**	61.3	70.1	65.4	55.9	62.1	**58.8**	53.5	59.1	**56.2**
12	Base+FrameNet (FN)	61.8	71.9	66.5	59.8	69.3	64.2	53.5	60.0	56.6	51.1	57.9	54.3
13	Base+Verb Pairs (VP)	62.1	72.2	66.8	60.1	69.3	64.4	54.4	60.1	57.1	51.9	58.2	54.9
14	Base+Appositives (AP)	63.1	71.7	67.1	60.5	69.4	64.6	54.1	60.1	56.9	51.9	57.8	54.7

Table 1: Results obtained by applying different types of features in isolation to the Baseline system.

		ACE						OntoNotes					
		B^3			CEAF			B^3			CEAF		
	Feature Set	R	P	F	R	P	F	R	P	F	R	P	F
	Results for the Mention-Pair Model												
1	Base	56.5	69.7	62.4	54.9	66.3	60.0	50.4	56.7	53.3	48.9	54.5	51.5
2	Base+YT	57.3	70.3	63.1	58.7	67.5	62.8	51.7	57.9	54.6	50.3	55.6	52.8
3	Base+YT+YM	57.8	70.9	63.6	59.1	67.9	63.2	52.1	58.3	55.0	50.8	56.0	53.3
4	Base+YT+YM+WP	59.5	71.9	65.1	57.5	69.4	62.9	53.1	59.2	56.0	51.5	57.1	54.1
5	Base+YT+YM+WP+FN	59.6	72.1	65.3	57.2	69.7	62.8	53.1	59.5	56.2	51.3	57.4	54.2
6	Base+YT+YM+WP+FN+VP	59.9	72.5	**65.6**	57.8	70.0	**63.3**	53.4	59.8	**56.4**	51.8	57.7	**54.6**
7	Base+YT+YM+WP+FN+VP+AP	59.7	72.4	65.4	57.6	69.8	63.1	53.2	59.8	56.3	51.5	57.6	54.4
	Results for the Cluster-Ranking Model												
8	Base	61.7	71.2	66.1	59.6	68.8	63.8	53.4	59.2	56.2	51.1	57.3	54.0
9	Base+YT	63.5	72.4	67.6	61.7	70.0	65.5	54.8	60.6	57.6	52.4	58.9	55.4
10	Base+YT+YM	63.9	72.6	68.0	62.1	70.4	66.0	55.2	61.0	57.9	52.8	59.1	55.8
11	Base+YT+YM+WP	66.1	75.4	70.4	62.9	72.4	67.3	57.7	64.4	**60.8**	55.1	61.6	**58.2**
12	Base+YT+YM+WP+FN	66.3	75.1	70.4	63.1	72.3	67.4	57.3	64.1	60.5	54.7	61.2	57.8
13	Base+YT+YM+WP+FN+VP	66.6	75.9	**70.9**	63.5	72.9	**67.9**	57.7	64.4	**60.8**	55.1	61.6	**58.2**
14	Base+YT+YM+WP+FN+VP+AP	66.4	75.7	70.7	63.3	72.9	67.8	57.6	64.3	**60.8**	55.0	61.5	58.1

Table 2: Results obtained by adding different types of features incrementally to the Baseline system.

OntoNotes annotations as the gold standard. As we can see, the MP model achieves F-measure scores of 62.4 (B^3) and 60.0 (CEAF) on ACE and 53.3 (B^3) and 51.5 (CEAF) on OntoNotes, and the CR model achieves F-measure scores of 66.1 (B^3) and 63.8 (CEAF) on ACE and 56.2 (B^3) and 54.0 (CEAF) on OntoNotes. Also, the results show that the CR model is stronger than the MP model, corroborating previous empirical findings (Rahman and Ng, 2009).

4.2.2 Incorporating World Knowledge

Next, we examine the usefulness of world knowledge for coreference resolution. The remaining rows in Table 1 show the results obtained when different types of features encoding world knowledge are applied to the Baseline system in isolation. The best result for each combination of data set, evaluation measure, and coreference model is boldfaced.

Two points deserve mention. First, each type of features improves the Baseline, regardless of the coreference model, the evaluation measure, and the annotation scheme used. This suggests that all these feature types are indeed useful for coreference resolution. It is worth noting that in all but a few cases involving the FrameNet-based and appositive-based features, the rise in F-measure is accompanied by a

1.	The Bush White House is breeding non-duck ducks the same way the **Nixon** White House did: It hops on an issue that is unopposable – cleaner air, better treatment of the disabled, better child care. **The President** came up with a good bill, but now may end up signing the awful bureaucratic creature hatched on Capitol Hill.
2.	**The tumor**, he suggested, developed when the second, normal copy also was damaged. He believed **colon cancer** might also arise from multiple "hits" on cancer suppressor genes, as it often seems to develop in stages.

Table 3: Examples errors introduced by YAGO and FrameNet.

simultaneous rise in recall and precision. This is perhaps not surprising: as the use of world knowledge helps discover coreference links, recall increases; and as more (relevant) knowledge is available to make coreference decisions, precision increases.

Second, the feature types that yield the best improvement over the Baseline are YAGO TYPE and Noun Pairs. When the MP model is used, the best coreference system improves the Baseline by 1–1.3% (B^3) and 1.3–2.8% (CEAF) in F-measure. On the other hand, when the CR model is used, the best system improves the Baseline by 2.3–2.6% (B^3) and 1.7–2.2% (CEAF) in F-measure.

Table 2 shows the results obtained when the different types of features are added to the Baseline one after the other. Specifically, we add the feature types in this order: YAGO TYPE, YAGO MEANS, Noun Pairs, FrameNet, Verb Pairs, and Appositives. In comparison to the results in Table 1, we can see that better results are obtained when the different types of features are applied to the Baseline in combination than in isolation, regardless of the coreference model, the evaluation measure, and the annotation scheme used. The best-performing system, which employs all but the Appositive features, outperforms the Baseline by 3.1–3.3% in F-measure when the MR model is used and by 4.1–4.8% in F-measure when the CR model is used. In both cases, the gains in F-measure are accompanied by a simultaneous rise in recall and precision. Overall, these results seem to suggest that the CR model is making more effective use of the available knowledge than the MR model, and that the different feature types are providing complementary information for the two coreference models.

4.3 Example Errors

While the different types of features we considered improve the performance of the Baseline primarily via the establishment of coreference links, some of these links are spurious. Sentences 1 and 2 of Table 3 show the spurious coreference links introduced by the CR model when YAGO and FrameNet are used, respectively. In sentence 1, while *The President* and *Bush* are coreferent, YAGO caused the CR model to establish the spurious link between *The President* and *Nixon* owing to the proximity of the two NPs and the presence of this NP pair in the YAGO TYPE relation. In sentence 2, FrameNet caused the CR model to establish the spurious link between *The tumor* and *colon cancer* because these two NPs are the ARG0 arguments of *develop* and *arise*, which appear in the same semantic frame in FrameNet.

5 Conclusions

We have examined the utility of three major sources of world knowledge for coreference resolution, namely, large-scale knowledge bases (YAGO, FrameNet), coreference-annotated data (Noun Pairs, Verb Pairs), and unannotated data (Appositives), by applying them to two learning-based coreference models, the mention-pair model and the cluster-ranking model, and evaluating them on documents annotated with the ACE and OntoNotes annotation schemes. When applying the different types of features in isolation to a Baseline system that does not employ world knowledge, we found that all of them improved the Baseline regardless of the underlying coreference model, the evaluation measure, and the annotation scheme, with YAGO TYPE and Noun Pairs yielding the largest performance gains. Nevertheless, the best results were obtained when they were applied in combination to the Baseline system. We conclude from these results that the different feature types we considered are providing complementary world knowledge to the coreference resolvers, and while each of them provides fairly small gains, their cumulative benefits can be substantial.

Acknowledgments

We thank the three reviewers for their invaluable comments on an earlier draft of the paper. This work was supported in part by NSF Grant IIS-0812261.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the Linguistic Coreference Workshop at The First International Conference on Language Resources and Evaluation*, pages 563–566.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90.

David Bean and Ellen Riloff. 2004. Unsupervised learning of contextual role knowledge for coreference resolution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 297–304.

Eric Bengtson and Dan Roth. 2008. Understanding the values of features for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 294–303.

Volha Bryl, Claudio Giuliano, Luciano Serafini, and Kateryna Tymoshenko. 2010. Using background knowledge to support coreference resolution. In *Proceedings of the 19th European Conference on Artificial Intelligence*, pages 759–764.

Eugene Charniak. 1972. *Towards a Model of Children's Story Comphrension*. AI-TR 266, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Aron Culotta, Michael Wick, and Andrew McCallum. 2007. First-order probabilistic models for coreference resolution. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 81–88.

Hal Daumé III and Daniel Marcu. 2005. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 97–104.

Pascal Denis and Jason Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 660–669.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 363–370.

Michael Fleischman, Eduard Hovy, and Abdessamad Echihabi. 2003. Offline strategies for online question answering: Answering questions before they are asked. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 1–7.

Aria Haghighi and Dan Klein. 2009. Simple coreference resolution with rich syntactic and semantic features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1152–1161.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the Bell tree. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 135–142.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32.

Ruslan Mitkov. 2002. *Anaphora Resolution*. Longman.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

Vincent Ng. 2007. Shallow semantics for coreference resolution. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 1689–1694.

Simone Paolo Ponzetto and Massimo Poesio. 2009. State-of-the-art NLP approaches to coreference resolution: Theory and practical recipes. In *Tutorial Abstracts of ACL-IJCNLP 2009*, page 6.

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proceedings*

of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, pages 192–199.

Sameer S. Pradhan, Wayne H. Ward, Kadri Hacioglu, James H. Martin, and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 233–240.

Altaf Rahman and Vincent Ng. 2009. Supervised models for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 968–977.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Veselin Stoyanov, Nathan Gilbert, Claire Cardie, and Ellen Riloff. 2009. Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 656–664.

Veselin Stoyanov, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom. 2010. Coreference resolution with Reconcile. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 156–161.

Fabian Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A core of semantic knowledge unifying wordnet and wikipedia. In *Proceedings of the World Wide Web Conference*, pages 697–706.

Olga Uryupina, Massimo Poesio, Claudio Giuliano, and Kateryna Tymoshenko. 2011. Disambiguation and filtering methods in using web knowledge for coreference resolution. In *Proceedings of the 24th International Florida Artificial Intelligence Research Society Conference*.

Xiaofeng Yang, Guodong Zhou, Jian Su, and Chew Lim Tan. 2003. Coreference resolution using competitive learning approach. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 176–183.

Xiaofeng Yang, Jian Su, Jun Lang, Chew Lim Tan, and Sheng Li. 2008. An entity-mention model for coreference resolution with inductive logic programming. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 843–851.

How to train your multi bottom-up tree transducer

Andreas Maletti
Universität Stuttgart, Institute for Natural Language Processing
Azenbergstraße 12, 70174 Stuttgart, Germany
andreas.maletti@ims.uni-stuttgart.de

Abstract

The local multi bottom-up tree transducer is introduced and related to the (non-contiguous) synchronous tree sequence substitution grammar. It is then shown how to obtain a weighted local multi bottom-up tree transducer from a bilingual and biparsed corpus. Finally, the problem of non-preservation of regularity is addressed. Three properties that ensure preservation are introduced, and it is discussed how to adjust the rule extraction process such that they are automatically fulfilled.

1 Introduction

A (formal) translation model is at the core of every machine translation system. Predominantly, statistical processes are used to instantiate the formal model and derive a specific translation device. Brown et al. (1990) discuss automatically trainable translation models in their seminal paper. However, the IBM models of Brown et al. (1993) are string-based in the sense that they base the translation decision on the words and their surrounding context. Contrary, in the field of syntax-based machine translation, the translation models have full access to the syntax of the sentences and can base their decision on it. A good exposition to both fields is presented in (Knight, 2007).

In this paper, we deal exclusively with syntax-based translation models such as synchronous tree substitution grammars (STSG), multi bottom-up tree transducers (MBOT), and synchronous tree-sequence substitution grammars (STSSG). Chiang (2006) gives a good introduction to STSG, which originate from the syntax-directed translation schemes of Aho and Ullman (1972). Roughly speaking, an STSG has rules in which two linked nonterminals are replaced (at the same time) by two corresponding trees containing terminal and nonterminal symbols. In addition, the nonterminals in the two replacement trees are linked, which creates new linked nonterminals to which further rules can be applied. Henceforth, we refer to these two trees as input and output tree. MBOT have been introduced in (Arnold and Dauchet, 1982; Lilin, 1981) and are slightly more expressive than STSG. Roughly speaking, they allow one replacement input tree and several output trees in a single rule. This change and the presence of states yields many algorithmically advantageous properties such as closure under composition, efficient binarization, and efficient input and output restriction [see (Maletti, 2010)]. Finally, STSSG, which have been derived from rational tree relations (Raoult, 1997), have been discussed by Zhang et al. (2008a), Zhang et al. (2008b), and Sun et al. (2009). They are even more expressive than the local variant of the multi bottom-up tree transducer (LMBOT) that we introduce here and can have several input and output trees in a single rule.

In this contribution, we restrict MBOT to a form that is particularly relevant in machine translation. We drop the general state behavior of MBOT and replace it by the common locality tests that are also present in STSG, STSSG, and STAG (Shieber and Schabes, 1990; Shieber, 2007). The obtained device is the local MBOT (LMBOT).

Maletti (2010) argued the algorithmical advantages of MBOT over STSG and proposed MBOT as an implementation alternative for STSG. In particular, the training procedure would train STSG; i.e., it would not utilize the additional expressive power

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 825–834,
Portland, Oregon, June 19-24, 2011.

of MBOT. However, Zhang et al. (2008b) and Sun et al. (2009) demonstrate that the additional expressivity gained from non-contiguous rules greatly improves the translation quality. In this contribution we address this separation and investigate a training procedure for LMBOT that allows non-contiguous fragments while preserving the algorithmic advantages of MBOT. To this end, we introduce a rule extraction and weight training method for LMBOT that is based on the corresponding procedures for STSG and STSSG. However, general LMBOT can be too expressive in the sense that they allow translations that do not preserve regularity. Preservation of regularity is an important property for efficient representations and efficient algorithms [see (May et al., 2010)]. Consequently, we present 3 properties that ensure that an LMBOT preserves regularity. In addition, we shortly discuss how these properties could be enforced in the rule extraction procedure.

2 Notation

The set of nonnegative integers is $\mathbb{N}$. We write $[k]$ for the set $\{i \mid 1 \leq i \leq k\}$. We treat functions as special relations. For every relation $R \subseteq A \times B$ and $S \subseteq A$, we write

$$R(S) = \{b \in B \mid \exists a \in S\colon (a,b) \in R\}$$
$$R^{-1} = \{(b,a) \mid (a,b) \in R\} \ ,$$

where R^{-1} is called the *inverse* of R.

Given an alphabet Σ, the set of all words (or sequences) over Σ is Σ^*, of which the empty word is ε. The concatenation of two words u and w is simply denoted by the juxtaposition uw. The length of a word $w = \sigma_1 \cdots \sigma_k$ with $\sigma_i \in \Sigma$ for all $i \in [k]$ is $|w| = k$. Given $1 \leq i \leq j \leq k$, the (i,j)-*span* $w[i,j]$ of w is $\sigma_i \sigma_{i+1} \cdots \sigma_j$.

The set T_Σ of all Σ-*trees* is the smallest set T such that $\sigma(\mathbf{t}) \in T$ for all $\sigma \in \Sigma$ and $\mathbf{t} \in T^*$. We generally use bold-face characters (like $\mathbf{t}$) for sequences, and we refer to their elements using subscripts (like t_i). Consequently, a tree t consists of a labeled root node σ followed by a sequence $\mathbf{t}$ of its children. To improve readability we sometimes write a sequence $t_1 \cdots t_k$ as $t_1, \ldots, t_k$.

The *positions* $\mathrm{pos}(t) \subseteq \mathbb{N}^*$ of a tree $t = \sigma(\mathbf{t})$ are inductively defined by $\mathrm{pos}(t) = \{\varepsilon\} \cup \mathrm{pos}(\mathbf{t})$, where

$$\mathrm{pos}(\mathbf{t}) = \bigcup_{1 \leq i \leq |\mathbf{t}|} \{ip \mid p \in \mathrm{pos}(t_i)\} \ .$$

Note that this yields an undesirable difference between $\mathrm{pos}(t)$ and $\mathrm{pos}(\mathbf{t})$, but it will always be clear from the context whether we refer to a single tree or a sequence. Note that positions are ordered via the (standard) lexicographic ordering. Let $t \in T_\Sigma$ and $p \in \mathrm{pos}(t)$. The label of t at position p is $t(p)$, and the subtree rooted at position p is $t|_p$. Formally, they are defined by

$$t(p) = \begin{cases} \sigma & \text{if } p = \varepsilon \\ \mathbf{t}(p) & \text{otherwise} \end{cases} \qquad \mathbf{t}(ip) = t_i(p)$$

$$t|_p = \begin{cases} t & \text{if } p = \varepsilon \\ \mathbf{t}|_p & \text{otherwise} \end{cases} \qquad \mathbf{t}|_{ip} = t_i|_p$$

for all $t = \sigma(\mathbf{t})$ and $1 \leq i \leq |\mathbf{t}|$. As demonstrated, these notions are also used for sequences. A position $p \in \mathrm{pos}(t)$ is a *leaf* (in t) if $p1 \notin \mathrm{pos}(t)$. Given a subset $\mathrm{NT} \subseteq \Sigma$, we let

$$\downarrow_{\mathrm{NT}}(t) = \{p \in \mathrm{pos}(t) \mid t(p) \in \mathrm{NT},\ p \text{ leaf in } t\} \ .$$

Later NT will be the set of nonterminals, so that the elements of $\downarrow_{\mathrm{NT}}(t)$ will be the *leaf nonterminals* of t. We extend the notion to sequences $\mathbf{t}$ by

$$\downarrow_{\mathrm{NT}}(\mathbf{t}) = \bigcup_{1 \leq i \leq |\mathbf{t}|} \{ip \mid p \in \downarrow_{\mathrm{NT}}(t_i)\} \ .$$

We also need a substitution that replaces subtrees. Let $p_1, \ldots, p_n \in \mathrm{pos}(t)$ be pairwise incomparable positions and $t_1, \ldots, t_n \in T_\Sigma$. Then $t[p_i \leftarrow t_i \mid 1 \leq i \leq n]$ denotes the tree that is obtained from t by replacing (in parallel) the subtrees at p_i by t_i for every $i \in [k]$.

Finally, let us recall regular tree languages. A *finite tree automaton* M is a tuple (Q, Σ, δ, F) such that Q is a finite set, $\delta \subseteq Q^* \times \Sigma \times Q$ is a finite relation, and $F \subseteq Q$. We extend δ to a mapping $\underline{\delta}\colon T_\Sigma \to 2^Q$ by

$$\underline{\delta}(\sigma(\mathbf{t})) = \{q \mid (\mathbf{q}, \sigma, q) \in \delta, \forall i \in [\,|\mathbf{t}|\,]\colon q_i \in \underline{\delta}(t_i)\}$$

for every $\sigma \in \Sigma$ and $\mathbf{t} \in T_\Sigma^*$. The finite tree automaton M recognizes the tree language

$$L(M) = \{t \in T_\Sigma \mid \underline{\delta}(t) \cap F \neq \emptyset\} \ .$$

A tree language $L \subseteq T_\Sigma$ is *regular* if there exists a finite tree automaton M such that $L = L(M)$.

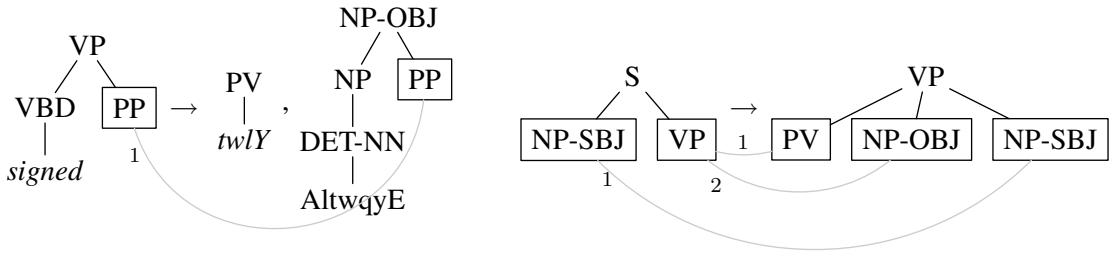


Figure 1: Sample LMBOT rules.

3 The model

In this section, we recall particular multi bottom-up tree transducers, which have been introduced by Arnold and Dauchet (1982) and Lilin (1981). A detailed (and English) presentation of the general model can be found in Engelfriet et al. (2009) and Maletti (2010). Using the nomenclature of Engelfriet et al. (2009), we recall a variant of *linear and nondeleting extended multi bottom-up tree transducers* (MBOT) here. Occasionally, we will refer to general MBOT, which differ from the local variant discussed here because they have explicit states.

Throughout the article, we assume sets Σ and Δ of input and output symbols, respectively. Moreover, let $\mathrm{NT} \subseteq \Sigma \cup \Delta$ be the set of designated nonterminal symbols. Finally, we avoid weights in the formal development to keep it simple. It is straightforward to add weights to our model.

Essentially, the model works on pairs $\langle t, \mathbf{u} \rangle$ consisting of an input tree $t \in T_\Sigma$ and a sequence $\mathbf{u} \in T_\Delta^*$ of output trees. Each such pair is called a *pre-translation* and the rank $\mathrm{rk}(\langle t, \mathbf{u} \rangle)$ the pre-translation $\langle t, \mathbf{u} \rangle$ is $|\mathbf{u}|$. In other words, the rank of a pre-translation equals the number of output trees stored in it. Given a pre-translation $\langle t, \mathbf{u} \rangle \in T_\Sigma \times T_\Delta^k$ and $i \in [k]$, we call u_i the i^{th} translation of t. An alignment for the pre-translation $\langle t, \mathbf{u} \rangle$ is an injective mapping $\psi \colon \downarrow_{\mathrm{NT}}(\mathbf{u}) \to \downarrow_{\mathrm{NT}}(t) \times \mathbb{N}$ such that $(p, j) \in \psi(\downarrow_{\mathrm{NT}}(\mathbf{u}))$ for every $(p, i) \in \psi(\downarrow_{\mathrm{NT}}(\mathbf{u}))$ and $j \in [i]$. In other words, an alignment should request each translation of a particular subtree at most once and if it requests the i^{th} translation, then it should also request all previous translations.

Definition 1 *A* local multi bottom-up tree transducer *(*LMBOT*) is a finite set R of* rules *such that every rule, written $l \to_\psi \mathbf{r}$, contains a pre-translation $\langle l, \mathbf{r} \rangle$ and an alignment ψ for it.*

The component l is the *left-hand side*, $\mathbf{r}$ is the *right-hand side*, and ψ is the *alignment* of a rule $l \to_\psi \mathbf{r} \in R$. The rules of an LMBOT are similar to the rules of an STSG (synchronous tree substitution grammar) of Eisner (2003) and Shieber (2004), but right-hand sides of LMBOT contain a sequence of trees instead of just a single tree as in an STSG. In addition, the alignments in an STSG rule are bijective between leaf nonterminals, whereas our model permits multiple alignments to a single leaf nonterminal in the left-hand side. A model that is even more powerful than LMBOT is the non-contiguous version of STSSG (synchronous tree-sequence substitution grammar) of Zhang et al. (2008a), Zhang et al. (2008b), and Sun et al. (2009), which allows sequences of trees on both sides of rules [see also (Raoult, 1997)]. Figure 1 displays sample rules of an LMBOT using a graphical representation of the trees and the alignment.

Next, we define the semantics of an LMBOT R. To avoid difficulties[1], we explicitly exclude rules like $l \to_\psi \mathbf{r}$ where $l \in \mathrm{NT}$ or $\mathbf{r} \in \mathrm{NT}^*$; i.e., rules where the left- or right-hand side are only leaf nonterminals. We first define the traditional bottom-up semantics. Let $\rho = l \to_\psi \mathbf{r} \in R$ be a rule and $p \in \downarrow_{\mathrm{NT}}(l)$. The p-rank $\mathrm{rk}(\rho, p)$ of ρ is $\mathrm{rk}(\rho, p) = |\{i \in \mathbb{N} \mid (p, i) \in \psi(\downarrow_{\mathrm{NT}}(\mathbf{r}))\}|$.

Definition 2 *The set $\tau(R)$ of pre-translations of an* LMBOT *R is inductively defined to be the smallest set such that: If $\rho = l \to_\psi \mathbf{r} \in R$ is a rule, $\langle t_p, \mathbf{u}_p \rangle \in \tau(R)$ is a pre-translation of R for every $p \in \downarrow_{\mathrm{NT}}(l)$, and*

- $\mathrm{rk}(\rho, p) = \mathrm{rk}(\langle t_p, \mathbf{u}_p \rangle)$,
- $l(p) = t_p(\varepsilon)$, *and*

[1] Actually, difficulties arise only in the weighted setting.

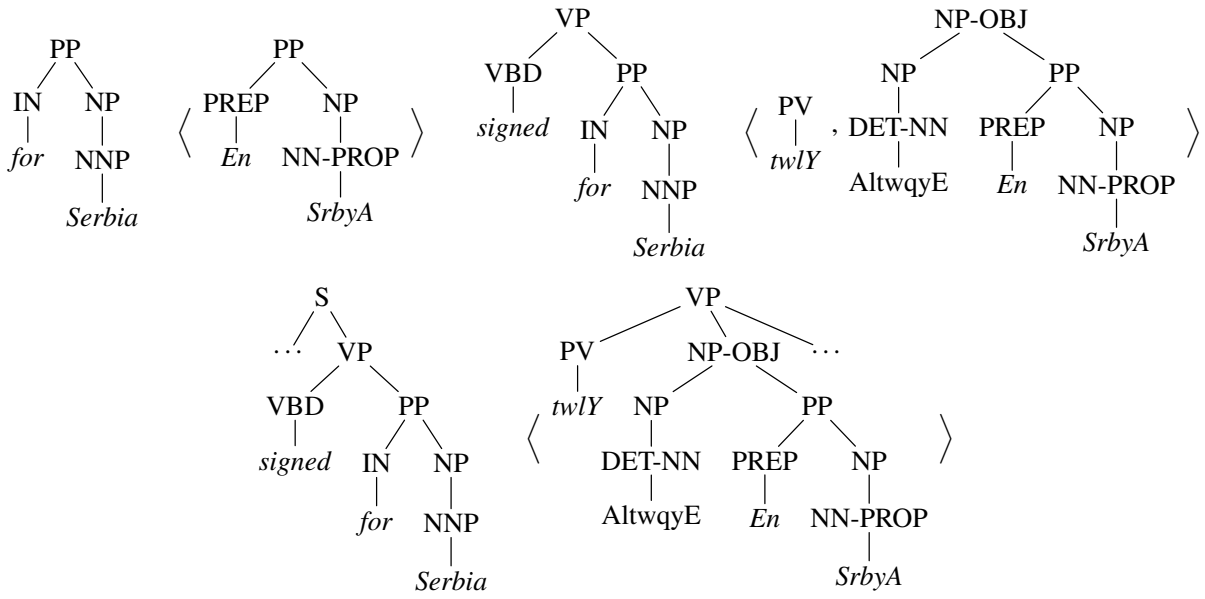


Figure 2: Top left: (a) Initial pre-translation; Top right: (b) Pre-translation obtained from the left rule of Fig. 1 and (a); Bottom: (c) Pre-translation obtained from the right rule of Fig. 1 and (b).

- $\mathbf{r}(p') = \mathbf{u}_{p''}(i)$ *with* $\psi(p') = (p'', i)$

for every $p' \in \downarrow_{\mathrm{NT}}(\mathbf{r})$*, then* $\langle t, \mathbf{u}\rangle \in \tau(R)$ *where*

- $t = l[p \leftarrow t_p \mid p \in \downarrow_{\mathrm{NT}}(l)]$ *and*
- $\mathbf{u} = \mathbf{r}[p' \leftarrow (\mathbf{u}_{p''})_i \mid p' \in \psi^{-1}(p'', i)]$.

In plain words, each nonterminal leaf p in the left-hand side of a rule ρ can be replaced by the input tree t of a pre-translation $\langle t, \mathbf{u}\rangle$ whose root is labeled by the same nonterminal. In addition, the rank $\mathrm{rk}(\rho, p)$ of the replaced nonterminal should match the rank $\mathrm{rk}(\langle t, \mathbf{u}\rangle)$ of the pre-translation and the nonterminals in the right-hand side that are aligned to p should be replaced by the translation that the alignment requests, provided that the nonterminal matches with the root symbol of the requested translation. The main benefit of the bottom-up semantics is that it works exclusively on pre-translations. The process is illustrated in Figure 2.

Using the classical bottom-up semantics, we simply obtain the following theorem by Maletti (2010) because the MBOT constructed there is in fact an LMBOT.

Theorem 3 *For every* STSG*, an equivalent* LMBOT *can be constructed in linear time, which in turn yields a particular* MBOT *in linear time.*

Finally, we want to relate LMBOT to the STSSG of Sun et al. (2009). To this end, we also introduce the top-down semantics for LMBOT. As expected, both semantics coincide. The top-down semantics is introduced using rule compositions, which will play an important rule later on.

Definition 4 *The set* R^k *of* k*-fold composed rules is inductively defined as follows:*

- $R^1 = R$ *and*
- $\ell \to_\varphi \mathbf{s} \in R^{k+1}$ *for all* $\rho = l \to_\psi \mathbf{r} \in R$ *and* $\rho_p = l_p \to_{\psi_p} \mathbf{r}_p \in R^k$ *such that*
  - $\mathrm{rk}(\rho, p) = \mathrm{rk}(\langle l_p, \mathbf{r}_p\rangle)$,
  - $l(p) = l_p(\varepsilon)$*, and*
  - $\mathbf{r}(p') = \mathbf{r}_{p''}(i)$ *with* $\psi(p') = (p'', i)$

  for every $p \in \downarrow_{\mathrm{NT}}(l)$ *and* $p' \in \downarrow_{\mathrm{NT}}(\mathbf{r})$ *where*
  - $\ell = l[p \leftarrow l_p \mid p \in \downarrow_{\mathrm{NT}}(l)]$,
  - $\mathbf{s} = \mathbf{r}[p' \leftarrow (\mathbf{r}_{p''})_i \mid p' \in \psi^{-1}(p'', i)]$*, and*
  - $\varphi(p'p) = p''\psi_{p''}(ip)$ *for all positions* $p' \in \psi^{-1}(p'', i)$ *and* $ip \in \downarrow_{\mathrm{NT}}(\mathbf{r}_{p''})$.

The rule closure $R^{\leq\infty}$ *of* R *is* $R^{\leq\infty} = \bigcup_{i\geq 1} R^i$*. The top-down pre-translation of* R *is*

$$\tau_{\mathrm{t}}(R) = \{\langle l, \mathbf{r}\rangle \mid l \to_\psi \mathbf{r} \in R^{\leq\infty}, \downarrow_{\mathrm{NT}}(l) = \emptyset\} \ .$$

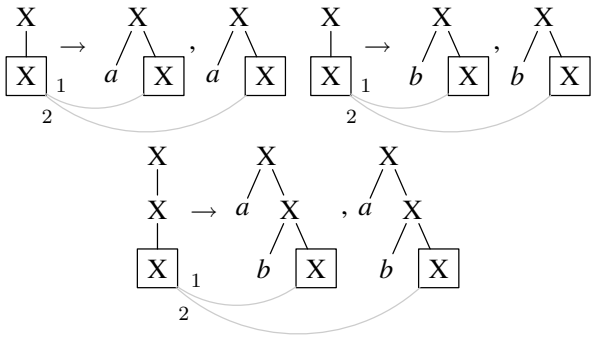


Figure 3: Composed rule.

The composition of the rules, which is illustrated in Figure 3, in the second item of Definition 4 could also be represented as $\rho(\rho_1, \ldots, \rho_k)$ where $\rho_1, \ldots, \rho_k$ is an enumeration of the rules $\{\rho_p \mid p \in \downarrow_{\mathrm{NT}}(l)\}$ used in the item. The following theorem is easy to prove.

Theorem 5 *The bottom-up and top-down semantics coincide; i.e.,* $\tau(R) = \tau_{\mathrm{t}}(R)$.

Chiang (2005) and Graehl et al. (2008) argue that STSG have sufficient expressive power for syntax-based machine translation, but Zhang et al. (2008a) show that the additional expressive power of tree-sequences helps the translation process. This is mostly due to the fact that smaller (and less specific) rules can be extracted from bi-parsed word-aligned training data. A detailed overview that focusses on STSG is presented by Knight (2007).

Theorem 6 *For every* LMBOT*, an equivalent* STSSG *can be constructed in linear time.*

4 Rule extraction and training

In this section, we will show how to automatically obtain an LMBOT from a bi-parsed, word-aligned parallel corpus. Essentially, the process has two steps: *rule extraction* and *training*. In the rule extraction step, an (unweighted) LMBOT is extracted from the corpus. The rule weights are then set in the training procedure.

The two main inspirations for our rule extraction are the corresponding procedures for STSG (Galley et al., 2004; Graehl et al., 2008) and for STSSG (Sun et al., 2009). STSG are always contiguous in both the left- and right-hand side, which means that they (completely) cover a single span of input or output words. On the contrary, STSSG rules can be non-contiguous on both sides, but the extraction procedure of Sun et al. (2009) only extracts rules that are contiguous on the left- or right-hand side. We can adjust its 1^{st} phase that extracts rules with (potentially) non-contiguous right-hand sides. The adjustment is necessary because LMBOT rules cannot have (contiguous) tree sequences in their left-hand sides. Overall, the rule extraction process is sketched in Algorithm 1.

Algorithm 1 Rule extraction for LMBOT

Require: word-aligned tree pair (t, u)
Return: LMBOT rules R such that $(t, u) \in \tau(R)$
  while there exists a maximal non-leaf node $p \in \mathrm{pos}(t)$ and minimal $p_1, \ldots, p_k \in \mathrm{pos}(u)$ such that $t|_p$ and $(u|_{p_1}, \ldots, u|_{p_k})$ have a consistent alignment (i.e., no alignments from within $t|_p$ to a leaf outside $(u|_{p_1}, \ldots, u|_{p_k})$ and vice versa)
  do
2: add rule $\rho = t|_p \to_\psi (u_{p_1}, \ldots, u_{p_k})$ to R with the nonterminal alignments ψ
      `// excise rule` ρ `from` (t, u)
4: $t \leftarrow t[p \leftarrow t(p)]$
    $u \leftarrow u[p_i \leftarrow u(p_i) \mid i \in \{1, \ldots, k\}]$
6: establish alignments according to position
  end while

The requirement that we can only have one input tree in LMBOT rules indeed might cause the extraction of bigger and less useful rules (when compared to the corresponding STSSG rules) as demonstrated in (Sun et al., 2009). However, the stricter rule shape preserves the good algorithmic properties of LMBOT. The more powerful STSSG rules can cause nonclosure under composition (Raoult, 1997; Radmacher, 2008) and parsing to be less efficient.

Figure 4 shows an example of biparsed aligned parallel text. According to the method of Galley et al. (2004) we can extract the (minimal) STSG rule displayed in Figure 5. Using the more liberal format of LMBOT rules, we can decompose the STSG rule of Figure 5 further into the rules displayed in Figure 1. The method of Sun et al. (2009) would also extract the rule displayed in Figure 6.

Let us reconsider Figures 1 and 2. Let ρ_1 be the top left rule of Figure 2 and ρ_2 and ρ_3 be the

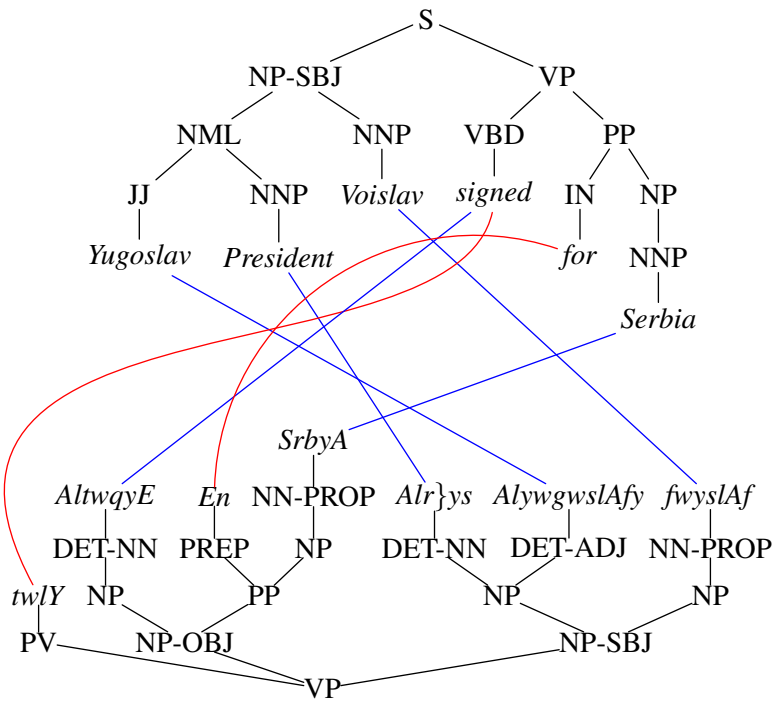


Figure 4: Biparsed aligned parallel text.

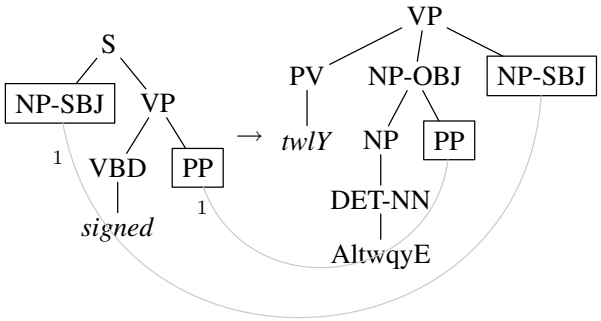


Figure 5: Minimal STSG rule.

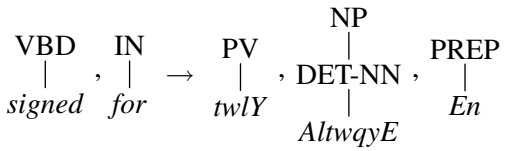


Figure 6: Sample STSSG rule.

left and right rule of Figure 1, respectively. We can represent the lower pre-translation of Figure 2 by $\rho_3(\cdots, \rho_2(\rho_1))$, where $\rho_2(\rho_1)$ represents the upper right pre-translation of Figure 2. If we name all rules of R, then we can represent each pre-translation of $\tau(R)$ symbolically by a tree containing rule names. Such trees containing rule names are often called *derivation trees*. Overall, we obtain the following result, for which details can be found in (Arnold and Dauchet, 1982).

Theorem 7 *The set $D(R)$ is a regular tree language for every* LMBOT *R, and the set of derivations is also regular for every* MBOT.

Moreover, using the input and output product constructions of Maletti (2010) we obtain that even the set $D_{t,u}(R)$ of derivations for a specific input tree t and output tree u is regular. Since $D_{t,u}(R)$ is regular, we can compute the inside and outside weight of each (weighted) rule of R following the method of Graehl et al. (2008). Similarly, we can adjust the training procedure of Graehl et al. (2008), which yields that we can automatically obtain a weighted LMBOT from a bi-parsed parallel corpus. Details on the run-time can be found in (Graehl et al., 2008).

5 Preservation of regularity

Clearly, LMBOT are not symmetric. Although, the backwards application of an LMBOT preserves regularity, this property does not hold for forward application. We will focus on forward application here. Given a set $\mathcal{T}$ of pre-translations and a tree language

$L \subseteq T_\Sigma$, we let

$$\mathcal{T}_{\mathrm{c}}(L) = \{u_i \mid (u_1, \ldots, u_k) \in \mathcal{T}(L), i \in [k]\} \ ,$$

which collects all translations of input trees in L. We say that $\mathcal{T}$ *preserves regularity* if $\mathcal{T}_{\mathrm{c}}(L)$ is regular for every regular tree language $L \subseteq T_\Sigma$. Correspondingly, an LMBOT R preserves regularity if its set $\tau(R)$ of pre-translations preserves regularity.

As mentioned, an LMBOT does not necessarily preserve regularity. The rules of an LMBOT have only alignments between the left-hand side (input tree) and the right-hand side (output tree), which are also called *inter-tree* alignments. However, several alignments to a single nonterminal in the left-hand side can transitively relate two different nonterminals in the output side and thus simulate an *intra-tree* alignment. For example, the right rule of Figure 1 relates a 'PV' and an 'NP-OBJ' node to a single 'VP' node in the left-hand side. This could lead to an intra-tree alignment (synchronization) between the 'PV' and 'NP-OBJ' nodes in the right-hand side.

Figure 7 displays the rules R of an LMBOT that does not preserve regularity. This can easily be seen on the leaf (word) languages because the LMBOT can translate the word x to any element of $L = \{wcwc \mid w \in \{a, b\}^*\}$. Clearly, this word language L is not context-free. Since the leaf language of every regular tree language is context-free and regular tree languages are closed under intersection (needed to single out the translations that have the symbol Y at the root), this also proves that $\tau(R)_{\mathrm{c}}(T_\Sigma)$ is not regular. Since T_Σ is regular, this proves that the LMBOT does not preserve regularity.

Preservation of regularity is an important property for a number of translation model manipulations. For example, the bucket-brigade and the on-the-fly method for the efficient inference described in (May et al., 2010) essentially build on it. Moreover, a regular tree grammar (i.e., a representation of a regular tree language) is an efficient representation. More complex representations such as context-free tree grammars [see, e.g., (Fujiyoshi, 2004)] have worse algorithmic properties (e.g., more complex parsing and problematic intersection).

In this section, we investigate three syntactic restrictions on the set R of rules that guarantees that the obtained LMBOT preserves regularity. Then we shortly discuss how to adjust the rule extraction algorithm, so that the extracted rules automatically have these property. First, we quickly recall the notion of composed rules from Definition 4 because it will play an essential role in all three properties. Figure 3 shows a composition of two rules from Figure 7. Mind that R^2 might not contain all rules of R, but it contains all those without leaf nonterminals.

Definition 8 *An* LMBOT R *is* finitely collapsing *if there is* $n \in \mathbb{N}$ *such that* $\psi\colon \downarrow_{\mathrm{NT}}(\mathbf{r}) \to \downarrow_{\mathrm{NT}}(l) \times \{1\}$ *for every rule* $l \to_\psi \mathbf{r} \in R^n$.

The following statement follows from a more general result of Raoult (1997), which we will introduce with our second property.

Theorem 9 *Every finitely collapsing* LMBOT *preserves regularity.*

Often the simple condition 'finitely collapsing' is fulfilled after rule extraction. In addition, it is automatically fulfilled in an LMBOT that was obtained from an STSG using Theorem 3. It can also be ensured in the rule extraction process by introducing *collapsing points* for output symbols that can appear recursively in the corpus. For example, we could enforce that all extracted rules for clause-level output symbols (assuming that there is no recursion not involving a clause-level output symbols) should have only 1 output tree in the right-hand side.

However, 'finitely collapsing' is a rather strict property. Finitely collapsing LMBOT have only slightly more expressive power than STSG. In fact, they could be called STSG with input desynchronization. This is due to the fact that the alignment in composed rules establishes an injective relation between leaf nonterminals (as in an STSG), but it need not be bijective. Consequently, there can be leaf nonterminals in the left-hand side that have no aligned leaf nonterminal in the right-hand side. In this sense, those leaf nonterminals are desynchronized. This feature is illustrated in Figure 8 and such an LMBOT can compute the transformation $\{(t, a) \mid t \in T_\Sigma\}$, which cannot be computed by an STSG (assuming that T_Σ is suitably rich). Thus STSG with input desynchronization are more expressive than STSG, but they still compute a class of transformations that is not closed under composition.

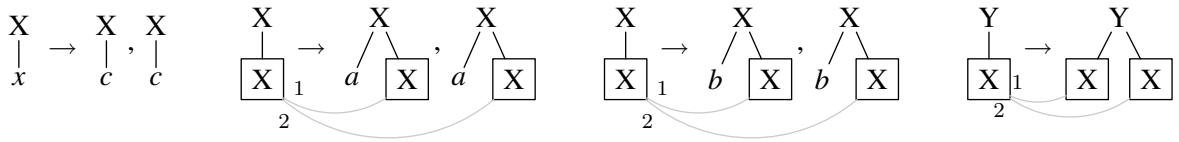


Figure 7: Output subtree synchronization (intra-tree).

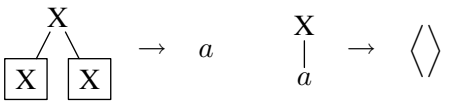


Figure 8: Finitely collapsing LMBOT.

Theorem 10 *For every* STSG*, we can construct an equivalent finitely collapsing* LMBOT *in linear time. Moreover, finitely collapsing* LMBOT *are strictly more expressive than* STSG*.*

Next, we investigate a weaker property by Raoult (1997) that still ensures preservation of regularity.

Definition 11 *An* LMBOT R *has* finite synchronization *if there is* $n \in \mathbb{N}$ *such that for every rule* $l \to_\psi \mathbf{r} \in R^n$ *and* $p \in \downarrow_{\mathrm{NT}}(l)$ *there exists* $i \in \mathbb{N}$ *with* $\psi^{-1}(\{p\} \times \mathbb{N}) \subseteq \{iw \mid w \in \mathbb{N}^*\}$.

In plain terms, multiple alignments to a single leaf nonterminal at p in the left-hand side are allowed, but all leaf nonterminals of the right-hand side that are aligned to p must be in the same tree. Clearly, an LMBOT with finite synchronization is finitely collapsing. Raoult (1997) investigated this restriction in the context of *rational tree relations*, which are a generalization of our LMBOT. Raoult (1997) shows that finite synchronization can be decided. The next theorem follows from the results of Raoult (1997).

Theorem 12 *Every* LMBOT *with finite synchronization preserves regularity.*

MBOT can compute arbitrary compositions of STSG (Maletti, 2010). However, this no longer remains true for MBOT (or LMBOT) with finite synchronization.[2] In Figure 9 we illustrate a translation that can be computed by a composition of two STSG, but that cannot be computed by an MBOT (or LMBOT) with finite synchronization. Intuitively, when processing the chain of 'X's of the transformation depicted in Figure 9, the first and second successor of the 'Z'-node at the root on the output side must be aligned to the 'X'-chain. This is necessary because those two mentioned subtrees must reproduce t_1 and t_2 from the end of the 'X'-chain. We omit the formal proof here, but obtain the following statement.

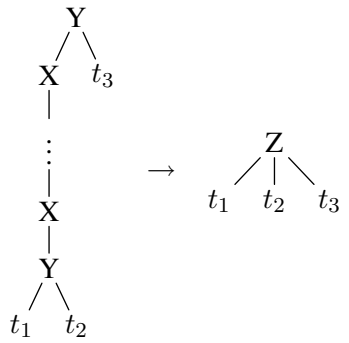


Figure 9: Transformation that cannot be computed by an MBOT with finite synchronization.

Theorem 13 *For every* STSG*, we can construct an equivalent* LMBOT *with finite synchronization in linear time.* LMBOT *and* MBOT *with finite synchronization are strictly more expressive than* STSG *and compute classes that are not closed under composition.*

Again, it is straightforward to adjust the rule extraction algorithm by the introduction of *synchronization points* (for example, for clause level output symbols). We can simply require that rules extracted for those selected output symbols fulfill the condition mentioned in Definition 11.

Finally, we introduce an even weaker version.

Definition 14 *An* LMBOT R *is* copy-free *if there is* $n \in \mathbb{N}$ *such that for every rule* $l \to_\psi \mathbf{r} \in R^n$ *and* $p \in \downarrow_{\mathrm{NT}}(l)$ *we have (i)* $\psi^{-1}(\{p\} \times \mathbb{N}) \subseteq \mathbb{N}$*, or (ii)* $\psi^{-1}(\{p\} \times \mathbb{N}) \subseteq \{iw \mid w \in \mathbb{N}^*\}$ *for an* $i \in \mathbb{N}$*.*

Intuitively, a copy-free LMBOT has rules whose right hand sides may use all leaf nonterminals that are aligned to a given leaf nonterminal in the left-hand side directly at the root (of one of the trees

[2]This assumes a straightforward generalization of the 'finite synchronization' property for MBOT.

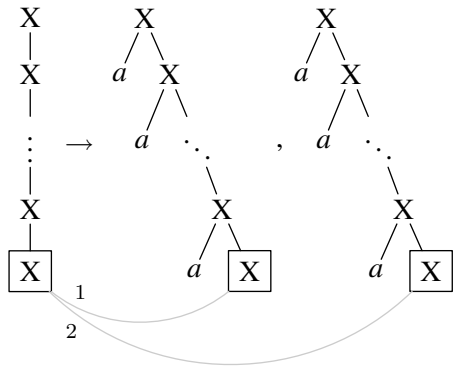


Figure 10: Composed rule that is not copy-free.

in the right-hand side forest) or group all those leaf nonterminals in a single tree in the forest. Clearly, the LMBOT of Figure 7 is not copy-free because the second rule composes with itself (see Figure 10) to a rule that does not fulfill the copy-free condition.

Theorem 15 *Every copy-free* LMBOT *preserves regularity.*

Proof sketch: Let n be the integer of Definition 14. We replace the LMBOT with rules R by the equivalent LMBOT M with rules R^n. Then all rules have the form required in Definition 14. Moreover, let $L \subseteq T_\Sigma$ be a regular tree language. Then we can construct the input product of $\tau(M)$ with L. In this way, we obtain an MBOT M', whose rules still fulfill the requirements (adapted for MBOT) of Definition 14 because the input product does not change the structure of the rules (it only modifies the state behavior). Consequently, we only need to show that the range of the MBOT M' is regular. This can be achieved using a decomposition into a relabeling, which clearly preserves regularity, and a deterministic finite-copying top-down tree transducer (Engelfriet et al., 1980; Engelfriet, 1982). □

Figure 11 shows some relevant rules of a copy-free LMBOT that computes the transformation of Figure 9. Clearly, copy-free LMBOT are more general than LMBOT with finite synchronization, so we again can obtain copy-free LMBOT from STSG. In addition, we can adjust the rule extraction process using synchronization points as for LMBOT with finite synchronization using the restrictions of Definition 14.

Theorem 16 *For every* STSG*, we can construct an equivalent copy-free* LMBOT *in linear time.*

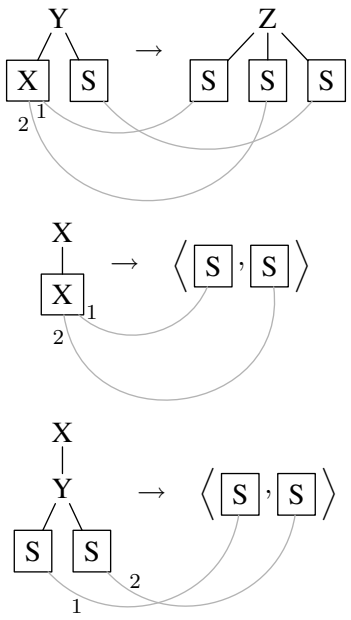


Figure 11: Copy-free LMBOT for the transformation of Figure 9.

Copy-free LMBOT *are strictly more expressive than* LMBOT *with finite synchronization.*

6 Conclusion

We have introduced a simple restriction of multi bottom-up tree transducers. It abstracts from the general state behavior of the general model and only uses the locality tests that are also present in STSG, STSSG, and STAG. Next, we introduced a rule extraction procedure and a corresponding rule weight training procedure for our LMBOT. However, LMBOT allow translations that do not preserve regularity, which is an important property for efficient algorithms. We presented 3 properties that ensure that regularity is preserved. In addition, we shortly discussed how these properties could be enforced in the presented rule extraction procedure.

Acknowledgements

The author gratefully acknowledges the support by KEVIN KNIGHT, who provided the inspiration and the data. JONATHAN MAY helped in many fruitful discussions.

The author was financially supported by the German Research Foundation (DFG) grant MA / 4959 / 1-1.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*. Prentice Hall.

André Arnold and Max Dauchet. 1982. Morphismes et bimorphismes d'arbres. *Theoret. Comput. Sci.*, 20(1):33–93.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. Mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. ACL*, pages 263–270. Association for Computational Linguistics.

David Chiang. 2006. An introduction to synchronous grammars. In *Proc. ACL*. Association for Computational Linguistics. Part of a tutorial given with Kevin Knight.

Jason Eisner. 2003. Simpler and more general minimization for weighted finite-state automata. In *Proc. NAACL*, pages 64–71. Association for Computational Linguistics.

Joost Engelfriet, Grzegorz Rozenberg, and Giora Slutzki. 1980. Tree transducers, L systems, and two-way machines. *J. Comput. System Sci.*, 20(2):150–202.

Joost Engelfriet, Eric Lilin, and Andreas Maletti. 2009. Composition and decomposition of extended multi bottom-up tree transducers. *Acta Inform.*, 46(8):561–590.

Joost Engelfriet. 1982. The copying power of one-state tree transducers. *J. Comput. System Sci.*, 25(3):418–435.

Akio Fujiyoshi. 2004. Restrictions on monadic context-free tree grammars. In *Proc. CoLing*, pages 78–84. Association for Computational Linguistics.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proc. HLT-NAACL*, pages 273–280. Association for Computational Linguistics.

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

Kevin Knight. 2007. Capturing practical natural language transformations. *Machine Translation*, 21(2):121–133.

Eric Lilin. 1981. Propriétés de clôture d'une extension de transducteurs d'arbres déterministes. In *Proc. CAAP*, volume 112 of LNCS, pages 280–289. Springer.

Andreas Maletti. 2010. Why synchronous tree substitution grammars? In *Proc. NAACL*, pages 876–884. Association for Computational Linguistics.

Jonathan May, Kevin Knight, and Heiko Vogler. 2010. Efficient inference through cascades of weighted tree transducers. In *Proc. ACL*, pages 1058–1066. Association for Computational Linguistics.

Frank G. Radmacher. 2008. An automata theoretic approach to rational tree relations. In *Proc. SOFSEM*, volume 4910 of LNCS, pages 424–435. Springer.

Jean-Claude Raoult. 1997. Rational tree relations. *Bull. Belg. Math. Soc. Simon Stevin*, 4(1):149–176.

Stuart M. Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proc. CoLing*, volume 3, pages 253–258. Association for Computational Linguistics.

Stuart M. Shieber. 2004. Synchronous grammars as tree transducers. In *Proc. TAG+7*, pages 88–95, Vancouver, BC, Canada. Simon Fraser University.

Stuart M. Shieber. 2007. Probabilistic synchronous tree-adjoining grammars for machine translation: The argument from bilingual dictionaries. In *Proc. SSST*, pages 88–95. Association for Computational Linguistics.

Jun Sun, Min Zhang, and Chew Lim Tan. 2009. A non-contiguous tree sequence alignment-based model for statistical machine translation. In *Proc. ACL*, pages 914–922. Association for Computational Linguistics.

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008a. A tree sequence alignment-based tree-to-tree translation model. In *Proc. ACL*, pages 559–567. Association for Computational Linguistics.

Min Zhang, Hongfei Jiang, Haizhou Li, Aiti Aw, and Sheng Li. 2008b. Grammar comparison study for translational equivalence modeling and statistical machine translation. In *Proc. CoLing*, pages 1097–1104. Association for Computational Linguistics.

Binarized Forest to String Translation

Hao Zhang
Google Research
haozhang@google.com

Licheng Fang
Computer Science Department
University of Rochester
lfang@cs.rochester.edu

Peng Xu
Google Research
xp@google.com

Xiaoyun Wu
Google Research
xiaoyunwu@google.com

Abstract

Tree-to-string translation is syntax-aware and efficient but sensitive to parsing errors. Forest-to-string translation approaches mitigate the risk of propagating parser errors into translation errors by considering a forest of alternative trees, as generated by a source language parser. We propose an alternative approach to generating forests that is based on combining sub-trees within the first best parse through binarization. Provably, our binarization forest can cover any non-consitituent phrases in a sentence but maintains the desirable property that for each span there is at most one nonterminal so that the grammar constant for decoding is relatively small. For the purpose of reducing search errors, we apply the synchronous binarization technique to forest-to-string decoding. Combining the two techniques, we show that using a fast shift-reduce parser we can achieve significant quality gains in NIST 2008 English-to-Chinese track (1.3 BLEU points over a phrase-based system, 0.8 BLEU points over a hierarchical phrase-based system). Consistent and significant gains are also shown in WMT 2010 in the English to German, French, Spanish and Czech tracks.

1 Introduction

In recent years, researchers have explored a wide spectrum of approaches to incorporate syntax and structure into machine translation models. The unifying framework for these models is *synchronous grammars* (Chiang, 2005) or *tree transducers* (Graehl and Knight, 2004). Depending on whether or not monolingual parsing is carried out on the source side or the target side for inference, there are four general categories within the framework:

- *string-to-string* (Chiang, 2005; Zollmann and Venugopal, 2006)
- *string-to-tree* (Galley et al., 2006; Shen et al., 2008)
- *tree-to-string* (Lin, 2004; Quirk et al., 2005; Liu et al., 2006; Huang et al., 2006; Mi et al., 2008)
- *tree-to-tree* (Eisner, 2003; Zhang et al., 2008)

In terms of search, the string-to-x models explore all possible source parses and map them to the target side, while the tree-to-x models search over the subspace of structures of the source side constrained by an input tree or trees. Hence, tree-to-x models are more constrained but more efficient. Models such as Huang et al. (2006) can match multi-level tree fragments on the source side which means larger contexts are taken into account for translation (Poutsma, 2000), which is a modeling advantage. To balance efficiency and accuracy, forest-to-string models (Mi et al., 2008; Mi and Huang, 2008) use a compact representation of exponentially many trees to improve tree-to-string models. Traditionally, such forests are obtained through hyper-edge pruning in the k-best search space of a monolingual parser (Huang, 2008). The pruning parameters that control the size of forests are normally hand-tuned. Such forests encode both syntactic variants and structural variants. By syntactic variants, we refer to the fact that a parser can parse a substring into either a noun phrase or verb phrase in certain cases.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 835–845,
Portland, Oregon, June 19-24, 2011.

We believe that structural variants which allow more source spans to be explored during translation are more important (DeNeefe et al., 2007), while syntactic variants might improve word sense disambiguation but also introduce more spurious ambiguities (Chiang, 2005) during decoding. To focus on structural variants, we propose a family of binarization algorithms to expand one single constituent tree into a packed forest of binary trees containing combinations of adjacent tree nodes. We control the freedom of tree node binary combination by restricting the distance to the lowest common ancestor of two tree nodes. We show that the best results are achieved when the distance is two, i.e., when combining tree nodes sharing a common grand-parent. In contrast to conventional parser-produced-forest-to-string models, in our model:

- Forests are not generated by a parser but by combining sub-structures using a tree binarizer.
- Instead of using arbitary pruning parameters, we control forest size by an integer number that defines the degree of tree structure violation.
- There is at most one nonterminal per span so that the grammar constant is small.

Since GHKM rules (Galley et al., 2004) can cover multi-level tree fragments, a synchronous grammar extracted using the GHKM algorithm can have synchronous translation rules with more than two nonterminals regardless of the branching factor of the source trees. For the first time, we show that similar to string-to-tree decoding, synchronous binarization significantly reduces search errors and improves translation quality for forest-to-string decoding.

To summarize, the whole pipeline is as follows. First, a parser produces the highest-scored tree for an input sentence. Second, the parse tree is restructured using our binarization algorithm, resulting in a binary packed forest. Third, we apply the forest-based variant of the GHKM algorithm (Mi and Huang, 2008) on the new forest for rule extraction. Fourth, on the translation forest generated by all applicable translation rules, which is not necessarily binary, we apply the synchronous binarization algorithm (Zhang et al., 2006) to generate a binary translation forest. Finally, we use a bottom-up decoding algorithm with intergrated LM intersection using the cube pruning technique (Chiang, 2005).

The rest of the paper is organized as follows. In Section 2, we give an overview of the forest-to-string models. In Section 2.1, we introduce a more efficient and flexible algorithm for extracting composed GHKM rules based on the same principle as cube pruning (Chiang, 2007). In Section 3, we introduce our source tree binarization algorithm for producing binarized forests. In Section 4, we explain how to do synchronous rule factorization in a forest-to-string decoder. Experimental results are in Section 5.

2 Forest-to-string Translation

Forest-to-string models can be described as

$$e = \mathcal{Y}(\underset{d \in D(T),\, T \in F(f)}{\arg\max} P(d|T)) \qquad (1)$$

where f stands for a source string, e stands for a target string, F stands for a forest, D stands for a set of synchronous derivations on a given tree T, and $\mathcal{Y}$ stands for the target side yield of a derivation. The search problem is finding the derivation with the highest probability in the space of all derivations for all parse trees for an input sentence. The log probability of a derivation is normally a linear combination of local features which enables dynamic programming to find the optimal combination efficiently. In this paper, we focus on the models based on the Synchronous Tree Substitution Grammars (STSG) defined by Galley et al. (2004). In contrast to a tree-to-string model, the introduction of F augments the search space systematically. When the first-best parse is wrong or no good translation rules are applicable to the first-best parse, the model can recover good translations from alternative parses.

In STSG, local features are defined on tree-to-string rules, which are synchronous grammar rules defining how a sequence of terminals and nonterminals on the source side translates to a sequence of target terminals and nonterminals. One-to-one mapping of nonterminals is assumed. But terminals do not necessarily need to be aligned. Figure 1 shows a typical English-Chinese tree-to-string rule with a reordering pattern consisting of two nonterminals and different numbers of terminals on the two sides.

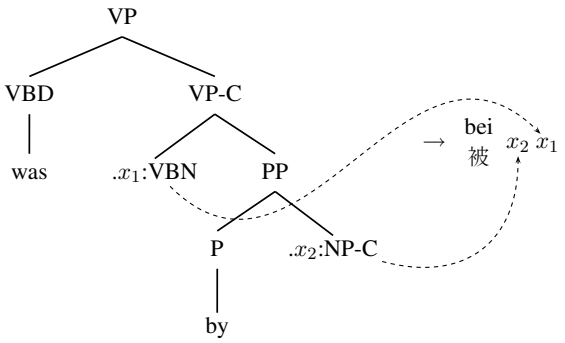


Figure 1: An example tree-to-string rule.

Forest-to-string translation has two stages. The first stage is rule extraction on word-aligned parallel texts with source forests. The second stage is rule enumeration and DP decoding on forests of input strings. In both stages, at each tree node, the task on the source side is to generate a list of tree fragments by composing the tree fragments of its children. We propose a cube-pruning style algorithm that is suitable for both rule extraction during training and rule enumeration during decoding.

At the highest level, our algorithm involves three steps. In the first step, we label each node in the input forest by a boolean variable indicating whether it is a site of interest for tree fragment generation. If it is marked true, it is an *admissible* node. In the case of rule extraction, a node is admissible if and only if it corresponds to a phrase pair according to the underlying word alignment. In the case of decoding, every node is admissible for the sake of completeness of search. An initial one-node tree fragment is placed at each admissible node for seeding the tree fragment generation process. In the second step, we do cube-pruning style bottom-up combinations to enumerate a pruned list of tree fragments at each tree node. In the third step, we extract or enumerate-and-match tree-to-string rules for the tree fragments at the admissible nodes.

2.1 A Cube-pruning-inspired Algorithm for Tree Fragment Composition

Galley et al. (2004) defined minimal tree-to-string rules. Galley et al. (2006) showed that tree-to-string rules made by composing smaller ones are important to translation. It can be understood by the analogy of going from word-based models to phrase-based models. We relate composed rule extraction to cube-pruning (Chiang, 2007). In cube-pruning, the process is to keep track of the k-best sorted language model states at each node and combine them bottom-up with the help of a priority queue. We can imagine substituting k-best LM states with k composed rules at each node and composing them bottom-up. We can also borrow the cube pruning trick to compose multiple lists of rules using a priority queue to lazily explore the space of combinations starting from the top-most element in the cube formed by the lists.

We need to define a ranking function for composed rules. To simulate the breadth-first expansion heuristics of Galley et al. (2006), we define the figure of merit of a tree-to-string rule as a tuple $m = (h, s, t)$, where h is the height of a tree fragment, s is the number of frontier nodes, i.e., bottom-level nodes including both terminals and non-terminals, and t is the number of terminals in the set of frontier nodes. We define an additive operator $+$:

$$\begin{aligned} & m_1 + m_2 \\ & \quad = (\ \max\{h_1, h_2\} + 1,\ s_1 + s_2,\ t_1 + t_2\ ) \end{aligned}$$

and a $\min$ operator based on the order $<$:

$$m_1 < m_2 \iff \begin{cases} h_1 < h_2 \vee \\ h_1 = h_2 \ \wedge\ s_1 < s_2 \vee \\ h_1 = h_2 \ \wedge\ s_1 = s_2 \ \wedge\ t_1 < t_2 \end{cases}$$

The $+$ operator corresponds to rule compositions. The $<$ operator corresponds to ranking rules by their sizes. A concrete example is shown in Figure 2, in which case the monotonicity property of $(+, <)$ holds: if $m_a < m_b$, $m_a + m_c < m_b + m_c$. However, this is not true in general for the operators in our definition, which implies that our algorithm is indeed like cube-pruning: an approximate k-shortest-path algorithm.

3 Source Tree Binarization

The motivation of tree binarization is to factorize large and rare structures into smaller but frequent ones to improve generalization. For example, Penn Treebank annotations are often flat at the phrase level. Translation rules involving flat phrases are unlikely to generalize. If long sequences are binarized,

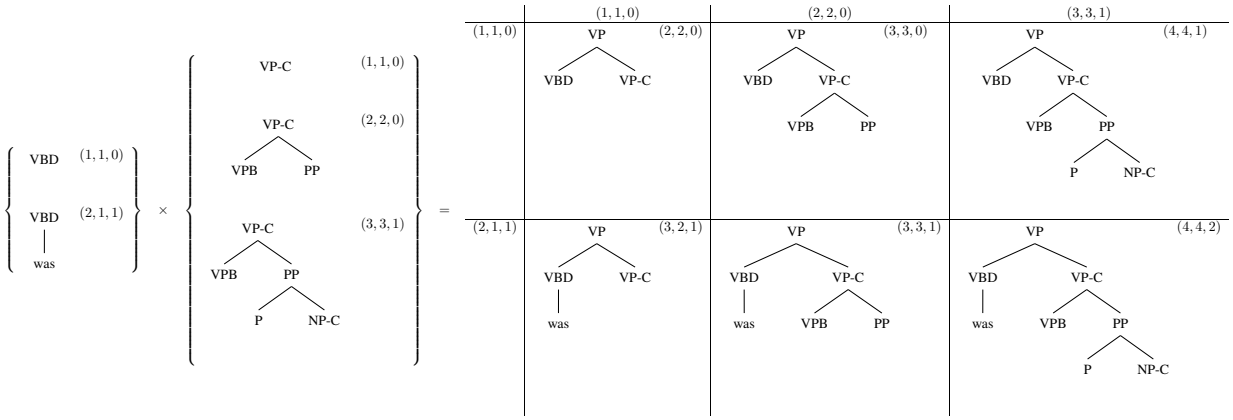


Figure 2: Tree-to-string rule composition as cube-pruning. The left shows two lists of composed rules sorted by their geometric measures $(height, \#\ frontiers, \#\ frontier\ terminals)$, under the gluing rule of VP $\rightarrow$ VBD VP$-$C. The right part shows a cube view of the combination space. We explore the space from the top-left corner to the neighbors.

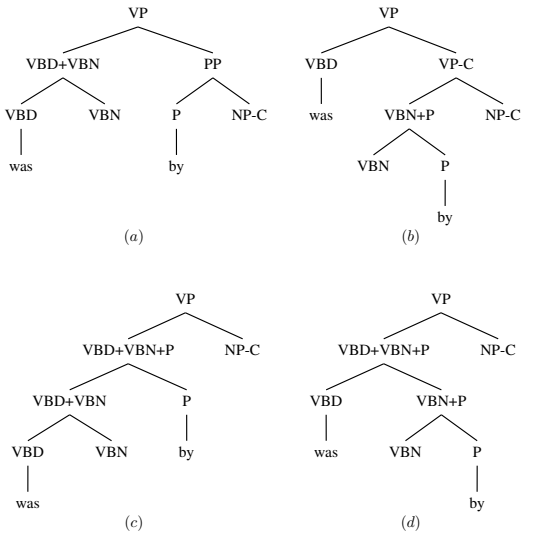
the commonality of subsequences can be discovered. For example, the simplest binarization methods *left-to-right*, *right-to-left*, and *head-out* explore sharing of prefixes or suffixes. Among exponentially many binarization choices, these algorithms pick a single bracketing structure for a sequence of sibling nodes. To explore all possible binarizations, we use a CYK algorithm to produce a packed forest of binary trees for a given sibling sequence.

With CYK binarization, we can explore any span that is nested within the original tree structure, but still miss all cross-bracket spans. For example, translating from English to Chinese, The phrase "There is" should often be translated into one verb in Chinese. In a correct English parse tree, however, the subject-verb boundary is between "There" and "is". As a result, tree-to-string translation based on constituent phrases misses the good translation rule.

The CYK-n binarization algorithm shown in Algorithm 1 is a parameterization of the basic CYK binarization algorithm we just outlined. The idea is that binarization can go beyond the scope of parent nodes to more distant ancestors. The CYK-n algorithm first annotates each node with its n nearest ancestors in the source tree, then generates a binarization forest that allows combining any two nodes with common ancestors. The ancestor chain labeled at each node licenses the node to only combine with nodes having common ancestors in the past n generations.

The algorithm creates new tree nodes on the fly. New tree nodes need to have their own states indicated by a node label representing what is covered internally by the node and an ancestor chain representing which nodes the node attaches to externally. Line 22 and Line 23 of Algorithm 1 update the label and ancestor annotations of new tree nodes. Using the parsing semiring notations (Goodman, 1999), the ancestor computation can be summarized by the $(\cap, \cup)$ pair. $\cap$ produces the ancestor chain of a hyper-edge. $\cup$ produces the ancestor chain of a hyper-node. The node label computation can be summarized by the $(\mathrm{concatenate}, \mathrm{min})$ pair. $\mathrm{concatenate}$ produces a concatenation of node labels. min yields the label with the shortest length. A *tree-sequence* (Liu et al., 2007) is a sequence of sub-trees covering adjacent spans. It can be proved that the final label of each new node in the forest corresponds to the tree sequence which has the minimum length among all sequences covered by the node span. The ancestor chain of a new node is the common ancestors of the nodes in its minimum tree sequence.

For clarity, we do full CYK loops over all $O(|w|^2)$ spans and $O(|w|^3)$ potential hyper-edges, where $|w|$ is the length of a source string. In reality, only descendants under a shared ancestor can combine. If we assume trees have a bounded branching factor b, the number of descendants after n generations is still bounded by a constant $c = b^n$. The algorithm is $O(c^3 \cdot |w|)$, which is still linear to the size of input sentence when the parameter n is a constant.



	1	2	3	4
0	VBD	*VBD+VBN*	*VBD+VBN+P*	VP
1		VBN	*VBN+P*	VP-C
2			P	PP
3				NP-C

Figure 3: Alternative binary parses created for the original tree fragment in Figure 1 through CYK-2 binarization (a and b) and CYK-3 binarization (c and d). In the chart representation at the bottom, cells with labels containing the concatenation symbol $+$ hold nodes created through binarization.

Figure 3 shows some examples of alternative trees generated by the CYK-n algorithm. In this example, standard CYK binarization will not create any new trees since the input is already binary. The CYK-2 and CYK-3 algorithms discover new trees with an increasing degree of freedom.

4 Synchronous Binarization for Forest-to-string Decoding

In this section, we deal with binarization of translation forests, also known as translation hypergraphs (Mi et al., 2008). A translation forest is a packed forest representation of all synchronous derivations composed of tree-to-string rules that match the source forest. Tree-to-string decoding algorithms work on a translation forest, rather than a source forest. A binary source forest does not necessarily always result in a binary translation forest. In the tree-to-string rule in Figure 4, the source tree is already

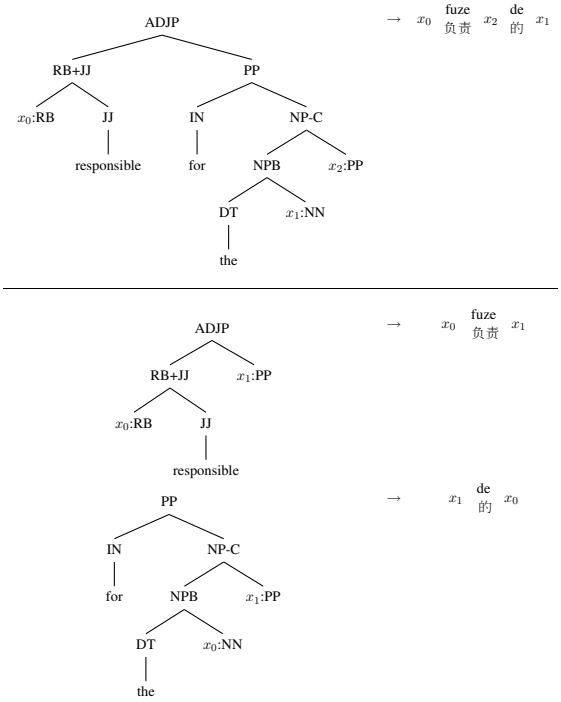


Figure 4: Synchronous binarization for a tree-to-string rule. The top rule can be binarized into two smaller rules.

binary with the help of source tree binarization, but the translation rule involves three variables in the set of frontier nodes. If we apply synchronous binarization (Zhang et al., 2006), we can factorize it into two smaller translation rules each having two variables. Obviously, the second rule, which is a common pattern, is likely to be shared by many translation rules in the derivation forest. When beams are fixed, search goes deeper in a factorized translation forest.

The challenge of synchronous binarization for a forest-to-string system is that we need to first match large tree fragments in the input forest as the first step of decoding. Our solution is to do the matching using the original rules and then run synchronous binarization to break matching rules down to factor rules which can be shared in the derivation forest. This is different from the offline binarization scheme described in (Zhang et al., 2006), although the core algorithm stays the same.

5 Experiments

We ran experiments on public data sets for English to Chinese, Czech, French, German, and Spanish

Algorithm 1 The CYK-n Binarization Algorithm

function CYKBINARIZER(T, n)
  for each tree $node \in T$ in bottom-up topological order **do**
    Make a copy of $node$ in the forest output F
    $Ancestors[node]$ = the nearest n ancestors of $node$
    $Label[node]$ = the label of $node$ in T
  $L \leftarrow$ the length of the yield of T
  for $k = 2...L$ **do**
    for $i = 0, ..., L - k$ **do**
      for $j = i + 1, ..., i + k - 1$ **do**
        $lnode \leftarrow Node[i, j]$; $rnode \leftarrow Node[j, i + k]$
        if $Ancestors[lnode] \cap Ancestors[rnode] \neq \emptyset$ **then**
          $pnode \leftarrow$ GETNODE($i, i + k$)
          ADDEDGE($pnode, lnode, rnode$)
  return F
function GETNODE($begin, end$)
  if $Node[begin, end] \notin F$ **then**
    Create a new $node$ for the span $(begin, end)$
    $Ancestors[node] = \emptyset$
    $Label[node]$ = the sequence of terminals in the span $(begin, end)$ in T
  return $Node[begin, end]$
function ADDEDGE($pnode, lnode, rnode$)
  Add a hyper-edge from $lnode$ and $rnode$ to $pnode$
  $Ancestors[pnode] = Ancestors[pnode] \cup (Ancestors[lnode] \cap Ancestors[rnode])$
  $Label[pnode] = \min\{Label[pnode],$ CONCATENATE($Label[lnode], Label[rnode]$)$\}$

translation to evaluate our methods.

5.1 Setup

For English-to-Chinese translation, we used all the allowed training sets in the NIST 2008 constrained track. For English to the European languages, we used the training data sets for WMT 2010 (Callison-Burch et al., 2010). For NIST, we filtered out sentences exceeding 80 words in the parallel texts. For WMT, the filtering limit is 60. There is no filtering on the test data set. Table 1 shows the corpus statistics of our bilingual training data sets.

	Source Words	Target Words
English-Chinese	287M	254M
English-Czech	66M	57M
English-French	857M	996M
English-German	45M	43M
English-Spanish	216M	238M

Table 1: The Sizes of Parallel Texts.

At the word alignment step, we did 6 iterations of IBM Model-1 and 6 iterations of HMM. For English-Chinese, we ran 2 iterations of IBM Model-4 in addition to Model-1 and HMM. The word alignments are symmetrized using the "union" heuristics. Then, the standard phrase extraction heuristics (Koehn et al., 2003) were applied to extract phrase pairs with a length limit of 6. We ran the hierarchical phrase extraction algorithm with the standard heuristics of Chiang (2005). The phrase-length limit is interpreted as the maximum number of symbols on either the source side or the target side of a given rule. On the same aligned data sets, we also ran the tree-to-string rule extraction algorithm described in Section 2.1 with a limit of 16 rules per tree node.

The default parser in the experiments is a shift-reduce dependency parser (Nivre and Scholz, 2004). It achieves 87.8% labelled attachment score and 88.8% unlabeled attachment score on the standard Penn Treebank test set. We convert dependency parses to constituent trees by propagating the part-of-speech tags of the head words to the corresponding phrase structures.

We compare three systems: a phrase-based system (Och and Ney, 2004), a hierarchical phrase-based system (Chiang, 2005), and our forest-to-string system with different binarization schemes. In the phrase-based decoder, jump width is set to 8. In the hierarchical decoder, only the glue rule is applied

to spans longer than 10. For the forest-to-string system, we do not have such length-based reordering constraints.

We trained two 5-gram language models with Kneser-Ney smoothing for each of the target languages. One is trained on the target side of the parallel text, the other is on a corpus provided by the evaluation: the Gigaword corpus for Chinese and news corpora for the others. Besides standard features (Och and Ney, 2004), the phrase-based decoder also uses a Maximum Entropy phrasal reordering model (Zens and Ney, 2006). Both the hierarchical decoder and the forest-to-string decoder only use the standard features. For feature weight tuning, we do Minimum Error Rate Training (Och, 2003). To explore a larger n-best list more efficiently in training, we adopt the hypergraph-based MERT (Kumar et al., 2009).

To evaluate the translation results, we use BLEU (Papineni et al., 2002).

5.2 Translation Results

Table 2 shows the scores of our system with the best binarization scheme compared to the phrase-based system and the hierarchical phrase-based system. Our system is consistently better than the other two systems in all data sets. On the English-Chinese data set, the improvement over the phrase-based system is 1.3 BLEU points, and 0.8 over the hierarchical phrase-based system. In the tasks of translating to European languages, the improvements over the phrase-based baseline are in the range of 0.5 to 1.0 BLEU points, and 0.3 to 0.5 over the hierarchical phrase-based system. All improvements except the *bf2s* and *hier* difference in English-Czech are significant with confidence level above 99% using the bootstrap method (Koehn, 2004). To demonstrate the strength of our systems including the two baseline systems, we also show the reported best results on these data sets from the 2010 WMT workshop. Our forest-to-string system (*bf2s*) outperforms or ties with the best ones in three out of four language pairs.

		BLEU dev	BLEU test
English-Chinese	pb	29.7	39.4
	hier	31.7	38.9
	bf2s	31.9	40.7**
English-Czech	*wmt best*	-	*15.4*
	pb	14.3	15.5
	hier	14.7	16.0
	bf2s	14.8	16.3*
English-French	*wmt best*	-	*27.6*
	pb	24.1	26.1
	hier	23.9	26.1
	bf2s	24.5	26.6**
English-German	*wmt best*	-	*16.3*
	pb	14.5	15.5
	hier	14.9	15.9
	bf2s	15.2	16.3**
English-Spanish	*wmt best*	-	*28.4*
	pb	24.1	27.9
	hier	24.2	28.4
	bf2s	24.9	28.9**

Table 2: Translation results comparing *bf2s*, the binarized-forest-to-string system, *pb*, the phrase-based system, and *hier*, the hierarchical phrase-based system. For comparison, the best scores from WMT 2010 are also shown. ** indicates the result is significantly better than both *pb* and *hier*. * indicates the result is significantly better than *pb* only.

5.3 Different Binarization Methods

The translation results for the *bf2s* system in Table 2 are based on the *cyk* binarization algorithm with bracket violation degree 2. In this section, we vary the degree to generate forests that are incrementally augmented from a single tree. Table 3 shows the scores of different tree binarization methods for the English-Chinese task.

It is clear from reading the table that *cyk*-2 is the optimal binarization parameter. We have verified this is true for other language pairs on non-standard data sets. We can explain it from two angles. At degree 2, we allow phrases crossing at most one bracket in the original tree. If the parser is reasonably good, crossing just one bracket is likely to cover most interesting phrases that can be translation units. From another point of view, enlarging the forests entails more parameters in the resulting translation model, making over-fitting likely to happen.

5.4 Binarizer or Parser?

A natural question is how the binarizer-generated forests compare with parser-generated forests in translation. To answer this question, we need a

	rules	BLEU dev	BLEU test
no binarization	378M	28.0	36.3
head-out	408M	30.0	38.2
cyk-1	527M	31.6	40.5
cyk-2	803M	31.9	40.7
cyk-3	1053M	32.0	40.6
cyk-∞	1441M	32.0	40.3

Table 3: Comparing different source tree binarization schemes for English-Chinese translation, showing both BLEU scores and model sizes. The rule counts include normal phrases which are used at the leaf level during decoding.

parser that can generate a packed forest. Our fast deterministic dependency parser does not generate a packed forest. Instead, we use a CRF constituent parser (Finkel et al., 2008) with state-of-the-art accuracy. On the standard Penn Treebank test set, it achieves an F-score of 89.5%. It uses a CYK algorithm to do full dynamic programming inference, so is much slower. We modified the parser to do hyperedge pruning based on posterior probabilities. The parser preprocesses the Penn Treebank training data through binarization. So the packed forest it produces is also a binarized forest. We compare two systems: one is using the *cyk*-2 binarizer to generate forests; the other is using the CRF parser with pruning threshold e^{-p}, where $p = 2$ to generate forests.[1] Although the parser outputs binary trees, we found cross-bracket *cyk*-2 binarization is still helpful.

	BLEU dev	BLEU test
cyk-2	14.9	16.0
parser	14.7	15.7

Table 4: Binarized forests versus parser-generated forests for forest-to-string English-German translation.

Table 4 shows the comparison of binarization forest and parser forest on English-German translation. The results show that *cyk*-2 forest performs slightly better than the parser forest. We have not done full exploration of forest pruning parameters to fine-tune the parser-forest. The speed of the constituent parser is the efficiency bottleneck. This actually demonstrates the advantage of the binarizer plus forest-to-string scheme. It is flexible, and works with any parser that generates projective parses. It does not require hand-tuning of forest pruning parameters for training.

[1]All hyper-edges with negative log posterior probability larger than p are pruned. In Mi and Huang (2008), the threshold is $p = 10$. The difference is that they do the forest pruning on a forest generated by a k-best algorithm, while we do the forest-pruning on the full CYK chart. As a result, we need more aggressive pruning to control forest size.

5.5 Synchronous Binarization

In this section, we demonstrate the effect of synchronous binarization for both tree-to-string and forest-to-string translation. The experiments are on the English-Chinese data set. The baseline systems use k-way cube pruning, where k is the branching factor, i.e., the maximum number of nonterminals on the right-hand side of any synchronous translation rule in an input grammar. The competing system does online synchronous binarization as described in Section 4 to transform the grammar intersected with the input sentence to the minimum branching factor k' ($k' < k$), and then applies k'-way cube pruning. Typically, k' is 2.

		BLEU dev	BLEU test
head-out	cube pruning	29.2	37.0
	+ synch. binarization	30.0	38.2
cyk-2	cube pruning	31.7	40.5
	+ synch. binarization	31.9	40.7

Table 5: The effect of synchronous binarization for tree-to-string and forest-to-string systems, on the English-Chinese task.

Table 5 shows that synchronous binarization does help reduce search errors and find better translations consistently in all settings.

6 Related Work

The idea of concatenating adjacent syntactic categories has been explored in various syntax-based models. Zollmann and Venugopal (2006) augmented hierarchial phrase based systems with joint syntactic categories. Liu et al. (2007) proposed tree-sequence-to-string translation rules but did not provide a good solution to place joint subtrees into connection with the rest of the tree structure. Zhang et

al. (2009) is the closest to our work. But their goal was to augment a k-best forest. They did not binarize the tree sequences. They also did not put constraint on the tree-sequence nodes according to how many brackets are crossed.

Wang et al. (2007) used target tree binarization to improve rule extraction for their string-to-tree system. Their binarization forest is equivalent to our *cyk*-1 forest. In contrast to theirs, our binarization scheme affects decoding directly because we match tree-to-string rules on a binarized forest.

Different methods of translation rule binarization have been discussed in Huang (2007). Their argument is that for tree-to-string decoding target side binarization is simpler than synchronous binarization and works well because creating discontinous source spans does not explode the state space. The forest-to-string senario is more similar to string-to-tree decoding in which state-sharing is important. Our experiments show that synchronous binarization helps significantly in the forest-to-string case.

7 Conclusion

We have presented a new approach to tree-to-string translation. It involves a source tree binarization step and a standard forest-to-string translation step. The method renders it unnecessary to have a k-best parser to generate a packed forest. We have demonstrated state-of-the-art results using a fast parser and a simple tree binarizer that allows crossing at most one bracket in each binarized node. We have also shown that reducing search errors is important for forest-to-string translation. We adapted the synchronous binarization technqiue to improve search and have shown significant gains. In addition, we also presented a new cube-pruning-style algorithm for rule extraction. In the new algorithm, it is easy to adjust the figure-of-merit of rules for extraction. In the future, we plan to improve the learning of translation rules with binarized forests.

Acknowledgments

We would like to thank the members of the MT team at Google, especially Ashish Venugopal, Zhifei Li, John DeNero, and Franz Och, for their help and discussions. We would also like to thank Daniel Gildea for his suggestions on improving the paper.

References

Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar Zaidan. 2010. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics(MATR)*, pages 17–53, Uppsala, Sweden, July. Association for Computational Linguistics. Revised August 2010.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Conference of the Association for Computational Linguistics (ACL-05)*, pages 263–270, Ann Arbor, MI.

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

Steve DeNeefe, Kevin Knight, Wei Wang, and Daniel Marcu. 2007. What can syntax-based MT learn from phrase-based MT? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 755–763, Prague, Czech Republic, June. Association for Computational Linguistics.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics, companion volume*, pages 205–208, Sapporo, Japan.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL-08: HLT*, pages 959–967, Columbus, Ohio, June. Association for Computational Linguistics.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of the 2004 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-04)*, pages 273–280.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the International Conference on Computational Linguistics/Association for Computational Linguistics (COLING/ACL-06)*, pages 961–968, July.

Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–605.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *Proceedings of the 2004 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-04)*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas (AMTA)*, Boston, MA.

Liang Huang. 2007. Binarization, synchronous binarization, and target-side binarization. In *Proceedings of the NAACL/AMTA Workshop on Syntax and Structure in Statistical Translation (SSST)*, pages 33–40, Rochester, NY.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the 46th Annual Conference of the Association for Computational Linguistics: Human Language Technologies (ACL-08:HLT)*, Columbus, OH. ACL.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-03)*, Edmonton, Alberta.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388–395, Barcelona, Spain, July.

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

Dekang Lin. 2004. A path-based transfer model for machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*, pages 625–630, Geneva, Switzerland.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the International Conference on Computational Linguistics/Association for Computational Linguistics (COLING/ACL-06)*, Sydney, Australia, July.

Yang Liu, Yun Huang, Qun Liu, and Shouxun Lin. 2007. Forest-to-string statistical translation rules. In *Proceedings of the 45th Annual Conference of the Association for Computational Linguistics (ACL-07)*, Prague.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 206–214, Honolulu, Hawaii, October. Association for Computational Linguistics.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of the 46th Annual Conference of the Association for Computational Linguistics: Human Language Technologies (ACL-08:HLT)*, pages 192–199.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of Coling 2004*, pages 64–70, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of the 41th Annual Conference of the Association for Computational Linguistics (ACL-03)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02)*.

Arjen Poutsma. 2000. Data-oriented translation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-00)*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd Annual Conference of the Association for Computational Linguistics (ACL-05)*, pages 271–279, Ann Arbor, Michigan.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of the 46th Annual Conference of the Association for Computational Linguistics: Human Language Technologies (ACL-08:HLT)*, Columbus, OH. ACL.

Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 746–754, Prague, Czech Republic, June. Association for Computational Linguistics.

Richard Zens and Hermann Ney. 2006. Discriminative reordering models for statistical machine translation. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 55–63, New York City, June. Association for Computational Linguistics.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the 2006 Meeting of the*

North American chapter of the Association for Computational Linguistics (NAACL-06), pages 256–263, New York, NY.

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008. A tree sequence alignment-based tree-to-tree translation model. In *Proceedings of ACL-08: HLT*, pages 559–567, Columbus, Ohio, June. Association for Computational Linguistics.

Hui Zhang, Min Zhang, Haizhou Li, Aiti Aw, and Chew Lim Tan. 2009. Forest-based tree sequence to string translation model. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 172–180, Suntec, Singapore, August. Association for Computational Linguistics.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 138–141, New York City, June. Association for Computational Linguistics.

Learning to Transform and Select Elementary Trees for Improved Syntax-based Machine Translations

Bing Zhao[†], and **Young-Suk Lee**[†], and **Xiaoqiang Luo**[†], and **Liu Li**[‡]
IBM T.J. Watson Research[†] and Carnegie Mellon University[‡]
{zhaob, ysuklee, xiaoluo}@us.ibm.com and liul@andrew.cmu.edu

Abstract

We propose a novel technique of learning how to transform the source parse trees to improve the translation qualities of syntax-based translation models using synchronous context-free grammars. We transform the source tree phrasal structure into a set of simpler structures, expose such decisions to the decoding process, and find the least expensive transformation operation to better model word reordering. In particular, we integrate synchronous binarizations, verb regrouping, removal of redundant parse nodes, and incorporate a few important features such as translation boundaries. We learn the structural preferences from the data in a generative framework. The syntax-based translation system integrating the proposed techniques outperforms the best Arabic-English unconstrained system in NIST-08 evaluations by 1.3 absolute BLEU, which is statistically significant.

1 Introduction

Most syntax-based machine translation models with synchronous context free grammar (SCFG) have been relying on the off-the-shelf monolingual parse structures to learn the translation equivalences for string-to-tree, tree-to-string or tree-to-tree grammars. However, state-of-the-art monolingual parsers are not necessarily well suited for machine translation in terms of both labels and chunks/brackets. For instance, in Arabic-to-English translation, we find only 45.5% of Arabic NP-SBJ structures are mapped to the English NP-SBJ with machine alignment and parse trees, and only 60.1% of NP-SBJs are mapped with human alignment and parse trees as in § 2. The chunking is of more concern; at best only 57.4% source chunking decisions are translated contiguously on the target side. To translate the rest of the chunks one has to frequently *break* the original structures. The main issue lies in the strong assumption behind SCFG-style nonterminals – each nonterminal (or variable) assumes a source chunk should be rewritten into a *contiguous* chunk in the target. Without integrating techniques to modify the parse structures, the SCFGs are not to be effective even for translating NP-SBJ in linguistically distant language-pairs such as Arabic-English.

Such problems have been noted in previous literature. Zollmann and Venugopal (2006) and Marcu et al. (2006) used broken syntactic fragments to augment their grammars to increase the rule coverage; while we learn optimal tree fragments transformed from the original ones via a generative framework, they enumerate the fragments available from the original trees without learning process. Mi and Huang (2008) introduced parse forests to blur the chunking decisions to a certain degree, to expand search space and reduce parsing errors from 1-best trees (Mi et al., 2008); others tried to use the parse trees as soft constraints on top of unlabeled grammar such as Hiero (Marton and Resnik, 2008; Chiang, 2010; Huang et al., 2010; Shen et al., 2010) without sufficiently leveraging rich tree context. Recent works tried more complex approaches to integrate both parsing and decoding in one *single* search space as in (Liu and Liu, 2010), at the cost of huge search space. In (Zhang et al., 2009), combinations of tree forest and *tree-sequence* (Zhang et al., 2008) based approaches were carried out by adding pseudo nodes and hyper edges into the forest. Overall, the forest-based translation can reduce the risks from upstream parsing errors and expand the search space, but it cannot sufficiently address the syntactic divergences between various language-pairs. The tree sequence approach adds pseudo nodes and hyper edges to the forest, which makes the forest even denser and harder for navigation and search. As trees thrive in the search space, especially with the pseudo nodes and edges being added to the already dense forest, it is becoming harder to wade through the deep forest for the best derivation path out.

We propose to simplify suitable subtrees to a reasonable level, at which the correct reordering can be easily identified. The transformed structure should be frequent enough to have rich statistics for learning a model. Instead of creating pseudo nodes and edges and make the forest dense, we transform a tree with a few simple operators; only meaningful frontier nodes, context nodes and edges are kept to induce the correct reordering; such operations also enable the model to share the statistics among all similar subtrees.

On the basis of our study on investigating the language divergence between Arabic-English with human aligned and parsed data, we integrate several simple statistical operations, to transform parse trees adaptively to serve the

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 846–855,
Portland, Oregon, June 19-24, 2011.

translation purpose better. For each source span in the given sentence, a subgraph, corresponding to an elementary tree (in Eqn. 1), is proposed for PSCFG translation; we apply a few operators to transform the subgraph into some frequent subgraphs seen in the whole training data, and thus introduce alternative similar translational equivalences to explain the same source span with enriched statistics and features. For instance, if we regroup two adjacent nodes IV and NP-SBJ in the tree, we can obtain the correct reordering pattern for verb-subject order, which is not easily available otherwise. By finding a set of similar elementary trees derived from the original elementary trees, statistics can be shared for robust learning.

We also investigate the features using the context beyond the phrasal subtree. This is to further disambiguate the transformed subgraphs so that informative neighboring nodes and edges can influence the reordering preferences for each of the transformed trees. For instance, at the beginning and end of a sentence, we do not expect dramatic long distance reordering to happen; or under SBAR context, the clause may prefer monotonic reordering for verb and subject. Such boundary features were treated as hard constraints in previous literature in terms of re-labeling (Huang and Knight, 2006) or re-structuring (Wang et al., 2010). The boundary cases were not addressed in the previous literature for trees, and here we include them in our feature sets for learning a MaxEnt model to predict the transformations. We integrate the neighboring context of the subgraph in our transformation preference predictions, and this improve translation qualities further.

The rest of the paper is organized as follows: in section 2, we analyze the projectable structures using human aligned and parsed data, to identify the problems for SCFG in general; in section 3, our proposed approach is explained in detail, including the statistical operators using a MaxEnt model; in section 4, we illustrate the integration of the proposed approach in our decoder; in section 5, we present experimental results; in section 6, we conclude with discussions and future work.

2 The Projectable Structures

A context-free style nonterminal in PSCFG rules means the source span governed by the nonterminal should be translated into a contiguous target chunk. A "projectable" phrase-structure means that it is translated into a contiguous span on the target side, and thus can be generalized into a nonterminal in our PSCFG rule. We carried out a controlled study on the projectable structures using human annotated parse trees and word alignment for 5k Arabic-English sentence-pairs.

In Table 1, the unlabeled F-measures with machine alignment and parse trees show that, for only 48.71% of the time, the boundaries introduced by the source parses

Alignment	Parse	Labels	Accuracy
H	H	NP-SBJ	0.6011
		PP	0.3436
		NP	0.4832
		unlabel	0.5739
M	H	NP-SBJ	0.5356
		PP	0.2765
		NP	0.3959
		unlabel	0.5305
M	M	NP-SBJ	0.4555
		PP	0.1935
		NP	0.3556
		unlabel	0.4871

Table 1: The labeled and unlabeled F-measures for projecting the source nodes onto the target side via alignments and parse trees; unlabeled F-measures show the bracketing accuracies for translating a source span contiguously. H: human, M: machine.

are real translation boundaries that can be explained by a nonterminal in PSCFG rule. Even for human parse and alignment, the unlabeled F-measures are still as low as 57.39%. Such statistics show that we should not blindly learn tree-to-string grammar; additional transformations to manipulate the bracketing boundaries and labels accordingly have to be implemented to guarantee the reliability of source-tree based syntax translation grammars. The transformations could be as simple as merging two adjacent nonterminals into one bracket to accommodate non-contiguity on the target side, or lexicalizing those words which have fork-style, many-to-many alignment, or unaligned content words to enable the rest of the span to be generalized into nonterminals. We illustrate several cases using the tree in Figure 1.

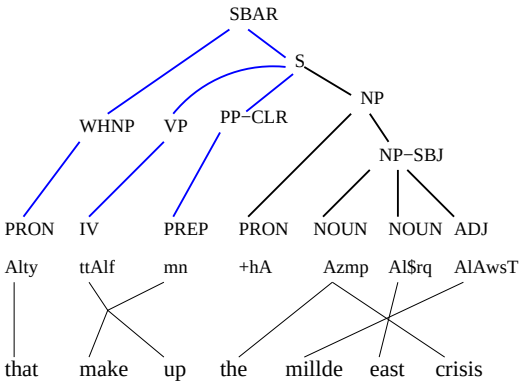


Figure 1: Non-projectable structures in an SBAR tree with human parses and alignment; there are non-projectable structures: the deleted nonterminals PRON (+hA), the many-to-many alignment for IV(ttAlf) PREP(mn), fork-style alignment for NOUN (Azmp).

In Figure 1, several non-projectable nodes were illus-

trated: the deleted nonterminals PRON (+hA), the many-to-many alignment for IV(ttAlf) PREP(mn), fork-style alignment for NOUN (Azmp). Intuitively, it would be good to glue the nodes NOUN(Al$rq) ADJ(AlAwsT) under the node of NP, because it is more frequent for moving ADJ before NOUN in our training data. It should be easier to model the swapping of (NOUN ADJ) using the tree (NP NOUN, ADJ) instead of the original bigger tree of (NP-SBJ Azmp, NOUN, ADJ) with one lexicalized node.

Approaches in tree-sequence based grammar (Zhang et al., 2009) tried to address the bracketing problem by using arbitrary pseudo nodes to weave a new "tree" back into the forest for further grammar extractions. Such approach may improve grammar coverage, but the pseudo node labels would be arguably a worse choice to split the already sparse data. Some of the interior nodes connecting the frontier nodes might be very informative for modeling reordering. Also, due to the introduced pseudo nodes, it would need *exponentially* many nonterminals to keep track of the matching tree-structures for translations. The created pseudo node could easily block the informative neighbor nodes associated with the subgraph which could change the reordering nature. For instance, IV and NP-SBJ tends to swap at the beginning of a sentence, but it may prefer monotone if they share a common parent of SBAR for a subclause. In this case, it is unnecessary to create a pseudo node "IV+SBJ" to block useful factors.

We propose to navigate through the forest, via simplifying trees by grouping the nodes, cutting the branches, and attaching connected neighboring informative nodes to further disambiguate the derivation path. We apply explicit translation motivated operators, on a given monolingual elementary tree, to transform it into similar but simpler trees, and expose such statistical preferences to the decoding process to select the best rewriting rule from the *enriched* grammar rule sets, for generating target strings.

3 Elementary Trees to String Grammar

We propose to use variations of an elementary tree, which is a connected subgraph fitted in the original monolingual parse tree. The subgraph is connected so that the frontiers (two or more) are connected by their *immediate common parent*. Let γ be a source elementary tree:

$$\gamma =< \ell; v^f, v^i, E >, \tag{1}$$

where v^f is a set of *frontier nodes* which contain nonterminals or words; v^i are the *interior nodes* with source labels/symbols; E is the set of *edges* connecting the nodes $v = v^f + v^i$ into a connected *subgraph* fitted in the source parse tree; ℓ is the *immediate common parent* of the frontier nodes v^f. Our proposed grammar rule is formulated as follows:

$$< \gamma; \alpha; \sim; \bar{m}; \bar{t} >, \tag{2}$$

where α is the target string, containing the terminals and/or nonterminals in a target language; $\sim$ is the one-to-one alignment of the nonterminals between γ and α; $\bar{t}$ contains possible sequence of transform operations (to be explained later in this section) associated with each rule; $\bar{m}$ is a function of enumerating the neighborhood of the source elementary tree γ, and certain tree context (nodes and edges) can be used to further disambiguate the reordering or the given lexical choices. The interior nodes of $\gamma.v^i$, however, are not necessarily informative for the reordering decisions, like the unary nodes WHNP,VP, and PP-CLR in Figure 1; while the frontier nodes $\gamma.v^f$ are the ones directly executing the reordering decisions. We can selectively cut off the interior nodes, which have no or only weak causal relations to the reordering decisions. This will enable the frequency or derived probabilities for executing the reordering to be more focused. We call such *transformation operators* $\bar{t}$. We specified a few operators for transforming an elementary tree γ, including flattening tree operators such as removing interior nodes in v^i, or grouping the children via binarizations.

Let's use the trigram "Alty ttAlf mn" in Figure 1 as an example, the immediate common parent for the span is SBAR: $\gamma.\ell$ = SBAR; the interior nodes are $\gamma.v^i$ = {WHNP VP S PP-CLR}; the frontier nodes are $\gamma.v^f$ = (x:PRON x:IV x:PREP). The edges $\gamma.E$ (as highlighted in Figure 1) connect $\gamma.v^i$ and $\gamma.v^f$ into a subgraph for the given source ngram.

For any source span, we look up one elementary tree γ covering the span, then we select an operator $\bar{t} \in T$, to explore a set of similar elementary trees $\bar{t}(\gamma, \bar{m}) = \{\gamma'\}$ as simplified alternatives for translating that source tree (span) γ into an optimal target string α^* accordingly. Our generative model is summarized in Eqn. 3:

$$\begin{aligned} \alpha^* = \underset{\bar{t}\in T;\gamma'\in\bar{t}(\gamma,\bar{m})}{\arg\max} \quad & p_a(\alpha'|\gamma')\times \\ & p_b(\gamma'|\bar{t},\gamma,\bar{m})\times \\ & p_c(\bar{t}|\gamma,\bar{m}). \end{aligned} \tag{3}$$

In our generative scheme, for a given elementary tree γ, we sample an operator (or a combination of operations) $\bar{t}$ with the probability of $p_c(\bar{t}|\gamma)$; with operation $\bar{t}$, we transform γ into a set of simplified versions $\gamma' \in \bar{t}(\gamma, \bar{m})$ with the probability of $p_b(\gamma'|\bar{t}, \gamma)$; finally we select the transformed version γ' to generate the target string α' with a probability of $p_a(\alpha'|\gamma')$. Note here, γ' and γ share the same immediate common parent ℓ, but not necessarily the frontier, or interior, or even neighbors. The frontier nodes can be merged, lexicalized, or even deleted in the tree-to-string rule associated with γ', as long as the alignment for the nonterminals are book-kept in the derivations. To simplify the model, one can choose the operator $\bar{t}$ to be only one level, and the model using a single operator $\bar{t}$ is to be deterministic. Thus, the final set of models

to learn are $p_a(\alpha'|\gamma')$ for rule alignment, and the preference model $p_b(\gamma'|\bar{t},\gamma,\bar{m})$, and the operator proposal model $p_c(\bar{t}|\gamma,\bar{m})$, which in our case is a maximum entropy model— the key model in our proposed approach in this paper for transforming the original elementary tree into similar trees for evaluating the reordering probabilities.

Eqn. 3 significantly enriches reordering powers for syntax-based machine translation. This is because it uses all *similar* set of elementary trees to generate the best target strings. In the next section, we'll first define the operators conceptually, and then explain how we learn each of the models.

3.1 Model $p_a(\alpha'|\gamma')$

A log linear model is applied here to approximate $p_a(\alpha'|\gamma') \propto \exp(\bar{\lambda}\cdot\overline{ff})$ via weighted combination ($\bar{\lambda}$) of feature functions $\overline{ff}(\alpha',\gamma')$, including relative frequencies in both directions, and IBM Model-1 scores in both directions as γ' and α' have lexical items within them. We also employed a few binary features listed in the following table.

γ' is observed less than 2 times
(α',γ') deletes a src content word
(α',γ') deletes a src function word
(α',γ') over generates a tgt content word
(α',γ') over generates a tgt function word

Table 2: Additional 5 Binary Features for $p_a(\alpha'|\gamma')$

3.2 Model $p_b(\gamma'|\bar{t},\gamma,\bar{m})$

$p_b(\gamma'|\bar{t},\gamma,\bar{m})$ is our preference model. For instance using the operator $\bar{t}$ of cutting an unary interior node in $\gamma.v^i$, if $\gamma.v^i$ has more than one unary interior node, like the SBAR tree in Figure 1, having three unary interior node: WHNP, VP and PP-CLR, $p_b(\gamma'|\bar{t},\gamma,\bar{m})$ specifies which one should have more probabilities to be cut. In our case, to make model simple, we simply choose histogram/frequency for modeling the choices.

3.3 Model $p_c(\bar{t}|\gamma,\bar{m})$

$p_c(\bar{t}|\gamma,\bar{m})$ is our operator proposal model. It ranks the operators which are valid to be applied for the given source tree γ together with its neighborhood $\bar{m}$. Here, in our approach, we applied a Maximum Entropy model, which is also employed to train our Arabic parser: $p_c(\bar{t}|\gamma,\bar{m}) \propto \exp\bar{\lambda}\cdot\overline{ff}(\bar{t},\gamma,\bar{m})$. The feature sets we use here are almost the same set we used to train our Arabic parser; the only difference is the future space here is operator categories, and we check bag-of-nodes for interior nodes and frontier nodes. The key feature categories we used are listed as in the Table 3. The headtable used in our training is manually built for Arabic.

bag-of-nodes $\gamma.v^i$
bag-of-nodes and ngram of $\gamma.v^f$
chunk-level features: left-child, right-child, etc.
lexical features: unigram and bigram
pos features: unigram and bigram
contextual features: surrounding words

Table 3: Feature Features for learning $p_c(\bar{t}|\gamma,\bar{m})$

3.4 $\bar{t}$: Tree Transformation Function

Obvious systematic linguistic divergences between language-pairs could be handled by some simple operators such as using binarization to re-group contiguously aligned children. Here, we start from the human aligned and parsed data as used in section 2 to explore potential useful operators.

3.4.1 Binarizations

One of the simplest way for transforming a tree is via binarization. Monolingual binarization chooses to re-group children into smaller subtree with a suitable label for the newly created root. We choose a function mapping to select the *top-frequent* label as the root for the grouped children; if such label is not found we simply use the label of the immediate common parent for γ. In decoding time, we need to select trees from all possible binarizations, while in the training time, we restrict the choices allowed with the alignment constraint, that every grouped children should be aligned contiguously on the target side. Our goal is to simulate the synchronous binarization as much as we can. In this paper, we applied the four basic operators for binarizing a tree: left-most, right-most and additionally head-out left and head-out right for more than three children. Two examples are given in Table 4, in which we used LDC style representation for the trees.

With the proper binarization, the structure becomes rich in sub-structures which allow certain reordering to happen more likely than others. For instance for the subtree (VP PV NP-SBJ), one would apply stronger statistics from training data to support the swap of NP-SBJ and PV for translation.

3.4.2 Regrouping verbs

Verbs are keys for reordering especially for Araic-English with VSO translated into SVO. However, if the verb and its relevant arguments for reordering are at different levels in the tree, the reordering is difficult to model as more interior nodes combinations will distract the distributions and make the model less focused. We provide the following two operations specific for verb in VP trees as in Table 5.

3.4.3 Removing interior nodes and edges

For reordering patterns, keeping the deep tree structure might not be the best choice. Sometimes it is not even

Binarization Operations	Examples
right-most	$(\text{NP}\ \ X_{\text{noun}}\ X_{\text{adj}_1}\ X_{\text{adj}_2}) \mapsto (\text{NP}\ \ X_{\text{noun}}\ (\text{ADJP}\ X_{\text{adj}_1}\ X_{\text{adj}_2}))$
left-most	$(\text{VP}\ \ X_{\text{pv}}\ X_{\text{NP-SBJ}}\ X_{\text{SBAR}}) \mapsto (\text{VP}\ \ (\text{VP}\ \ X_{\text{pv}}\ X_{\text{NP-SBJ}})\ X_{\text{SBAR}})$

Table 4: Operators for binarizing the trees

Operators for regroup verbs	Examples
regroup verb	$(VP_1\ X_v\ (VP_2\ Y)) \mapsto (VP_1\ (VP_2\ X_v\ Y))$
regroup verb and remove the top level VP	$(R\ \ (VP_1\ X_v\ (R_2\ Y))) \mapsto (R\ \ (R_2\ X_v Y))$

Table 5: Operators for manipulating the trees

possible due to the many-to-many alignment, insertions and deletions of terminals. So, we introduce the operators to remove the interior nodes $\gamma.v^i$ selectively; this way, we can flatten the tree, remove irrelevant nodes and edges, and can use more frequent observations of simplified structures to capture the reordering patterns. We use two operators as shown in Table 6.

The second operator deletes all the interior nodes, labels and edges; thus reordering will become a Hiero-alike (Chiang, 2007) *unlabeled rule*, and additionally a special glue rule: $X_1X_2 \rightarrow X_1X_2$. This operator is necessary, we need a scheme to automatically back off to the meaningful glue or Hiero-alike rules, which may lead to a cheaper derivation path for constructing a partial hypothesis, at the decoding time.

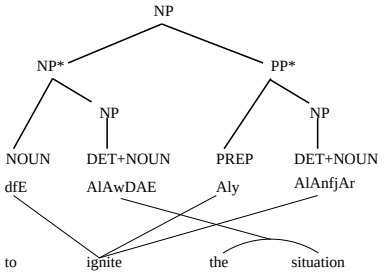


Figure 2: A NP tree with an "inside-out" alignment. The nodes "NP*" and "PP*" are not suitable for generalizing into NTs used in PSCFG rules.

As shown in Table 1, NP brackets has only 35.56% of time to be translated contiguously as an NP in machine aligned & parsed data. The NP tree in Figure 2 happens to be an "inside-out" style alignment, and context free grammar such as ITG (Wu, 1997) can not explain this structure well without necessary lexicalization. Actually, the Arabic tokens of "dfE Aly AlAnfjAr" form a combination and is turned into English word "ignite" in an idiomatic way. With lexicalization, a Hiero style rule "dfE X Aly AlAnfjAr $\mapsto$ to ignite X" is potentially a better alternative for translating the NP tree. Our operators allow us to back off to such Hiero-style rules to construct derivations, which share the immediate common parent NP, as defined for the elementary tree, for the given source span.

3.5 $\bar{m}$: Neighboring Function

For a given elementary tree, we use function $\bar{m}$ to check the context beyond the subgraph. This includes looking the nodes and edges connected to the subgraph. Similar to the features used in (Dyer et al., 2009), we check the following three cases.

3.5.1 Sentence boundaries

When the tree γ frontier sets contain the left-most token, right-most token, or both sides, we will add to the neighboring nodes the corresponding decoration tags L (left), R (right), and B (both), respectively. These decorations are important especially when the reordering patterns for the same trees are depending on the context. For instance, at the beginning or end of a sentence, we do not expect dramatic reordering – moving a token too far away in the middle of the sentences.

3.5.2 SBAR/IP/PP/FRAG boundaries

We check siblings of the root for γ for a few special labels, including SBAR, IP, PP, and FRAG. These labels indicate a partial sentence or clause, and the reordering patterns may get different distributions due to the position relative to these nodes. For instance, the PV and SBJ nodes under SBAR tends to have more monotone preference for word reordering (Carpuat et al., 2010). We mark the boundaries with position markers such as L-PP, to indicate having a left sibling PP, R-IP for having a right sibling IP, and C-SBAR to indicate the elementary tree is a child of SBAR. These labels are selected mainly based on our linguistic intuitions and errors in our translation system. A data-driven approach might be more promising for identifying useful markups w.r.t specific reordering patterns.

3.5.3 Translation boundaries

In the Figure 2, there are two special nodes under NP: NP* and PP*. These two nodes are aligned in a "inside-out" fashion, and none of them can be generalized into a nonterminal to be rewritten in a PSCFG rule. In other words, the phrasal brackets induced from NP* and PP*

operators for removing nodes/edges	Examples
remove unary nodes	$(R\ \ X_{t_1}(R_1\ \ (R2\ \ X_{t_2}))) \mapsto (R\ \ X_{t_1}(R2\ \ X_{t_2}))$
remove all labels	$(R\ \ (R_1\ \ X_{t_1}(R_2\ \ X_{t_2}))) \mapsto\ \ (R\ \ X_{t_2}X_{t_1})$

Table 6: Operators for simplifying the trees

are not *translation boundaries*, and to avoid translation errors we should identify them by applying a PSCFG rule on top of them. During training, we label nodes with translation boundaries, as one additional *function tag*; during decoding, we employ the MaxEnt model to predict the translation boundary label probability for each span associated with a subgraph γ, and discourage derivations accordingly for using nonterminals over the non-translation boundary span. The translation boundaries over elementary trees have much richer representation power. The previous works as in Xiong et al. (2010), defined translation boundaries on phrase-decoder style derivation trees due to the nature of their shift-reduce algorithm, which is a special case in our model.

4 Decoding

Decoding using the proposed elementary tree to string grammar naturally resembles bottom up chart parsing algorithms. The key difference is at the grammar querying step. Given a grammar G, and the input source parse tree π from a monolingual parser, we first construct the elementary tree for a source span, and then retrieve all the relevant subgraphs seen in the given grammar through the proposed operators. This step is called populating, using the proposed operators to find all relevant elementary trees γ which may have contributed to explain the source span, and put them in the corresponding cells in the chart. There would have been exponential number of relevant elementary trees to search if we do not have any restrictions in the populating step; we restrict the maximum number of interior nodes $|\gamma.v^i|$ to be 3, and the size of frontier nodes $|\gamma.v^f|$ to be less than 6; additional pruning for less frequent elementary trees is carried out.

After populating the elementary trees, we construct the partial hypotheses bottom up, by rewriting the frontier nodes of each elementary tree with the probabilities(costs) for $\gamma \rightarrow \alpha*$ as in Eqn. 3. Our decoder (Zhao and Al-Onaizan, 2008) is a template-based chart decoder in C++. It generalizes over the dotted-product operator in Earley style parser, to allow us to leverage many operators $\bar{t} \in T$ as above-mentioned, such as binarizations, at different levels for constructing partial hypothesis.

5 Experiments

In our experiments, we built our system using most of the parallel training data available to us: 250M Arabic running tokens, corresponding to the "unconstrained" condition in NIST-MT08. We chose the testsets of *newswire* and *weblog* genres from MT08 and DEV10[1]. In particular, we choose MT08 to enable the comparison of our results to the reported results in NIST evaluations. Our training and test data is summarized in Table 5. For testings, we have 129,908 tokens in our testsets. For language models (LM), we used 6-gram LM trained with 10.3 billion English tokens, and also a shrinkage-based LM (Chen, 2009) – "ModelM" (Chen and Chu, 2010; Emami et al., 2010) with 150 word-clusters learnt from 2.1 million tokens.

From the parallel data, we extract phrase pairs(blocks) and elementary trees to string grammar in various configurations: basic tree-to-string rules (Tr2str), elementary tree-to-string rules with boundaries $\bar{t}$(elm2str+$\bar{m}$), and with both $\bar{t}$ and $\bar{m}$ (elm2str+$\bar{t}$ + $\bar{m}$). This is to evaluate the operators' effects at different levels for decoding. To learn our MaxEnt models defined in § 3.3, we collect the events during extracting elm2str grammar in training time, and learn the model using improved iterative scaling. We use the same training data as that used in training our Arabic parser. There are 16 thousand human parse trees with human alignment; additional 1 thousand human parse and aligned sent-pairs are used as unseen test set to verify our MaxEnt models and parsers. For our Arabic parser, we have a labeled F-measure of 78.4%, and POS tag accuracy 94.9%. In particular, we'll evaluate model $p_c(\bar{t}|\gamma, \bar{m})$ in Eqn. 3 for predicting the translation boundaries in § 3.5.3 for projectable spans as detailed in § 5.1.

Our decoder (Zhao and Al-Onaizan, 2008) supports grammars including monotone, ITG, Hiero, tree-to-string, string-to-tree, and several mixtures of them (Lee et al., 2010). We used 19 feature functions, mainly from those used in phrase-based decoder like Moses (Koehn et al., 2007), including two language models (one for a 6-gram LM, one for ModelM, one brevity penalty, IBM Model-1 (Brown et al., 1993) style alignment probabilities in both directions, relative frequency in both directions, word/rule counts, content/function word mismatch, together with features on tr2str rule probabilities. We use BLEU (Papineni et al., 2002) and TER (Snover et al., 2006) to evaluate translation qualities. Our baseline used basic elementary tree to string grammar without any manipulations and boundary markers in the model,

[1]DEV10 are unseen testsets used in our GALE project. It was selected from recently released LDC data LDC2010E43.v3.

Data	Train	MT08-NW	MT08-WB	Dev10-NW	Dev10-WB
# Sents	8,032,837	813	547	1089	1059
# Tokens	349M(ar)/230M(en)	25,926	19,654	41,240	43,088

Table 7: Training and test data; using all training parallel training data for 4 test sets

and we achieved a BLEUr4n4 55.01 for MT08-NW, or a cased BLEU of 53.31, which is close to the best officially reported result 53.85 for unconstrained systems.[2] We expose the statistical decisions in Eqn. 3 as the rule probability as one of the 19 dimensions, and use Simplex Downhill algorithm with Armijo line search (Zhao and Chen, 2009) to optimize the weight vector for decoding. The algorithm moves all dimensions at the same time, and empirically achieved more stable results than MER(Och, 2003) in many of our experiments.

5.1 Predicting Projectable Structures

The projectable structure is important for our proposed elementary tree to string grammar (elm2str). When a span is predicted not to be a translation boundary, we want the decoder to prefer alternative derivations outside of the immediate elementary tree, or more aggressive manipulation of the trees, such as deleting interior nodes, to explore unlabeled grammar such as Hiero style rules, with proper costs. We test separately on predicting the projectable structures, like predicting function tags in § 3.5.3, for each node in syntactic parse tree. We use one thousand test sentences with two conditions: human parses and machine parses. There are totally 40,674 nodes excluding the sentence-level node. The results are shown in Table 8. It showed our MaxEnt model is very accurate using human trees: 94.5% of accuracy, and about 84.7% of accuracy for using the machine parsed trees. Our accuracies are higher compared with the 71+% accuracies reported in (Xiong et al., 2010) for their phrasal decoder.

Setups	Accuracy
Human Parses	94.5%
Machine Parses	84.7%

Table 8: Accuracies of predicting projectable structures

We zoom in the translation boundaries for MT08-NW, in which we studied a few important frequent labels including VP and NP-SBJ as in Table 9. According to our MaxEnt model, 20% of times we should discourage a VP tree to be translated contiguously; such VP trees have an average span length of 16.9 tokens in MT08-NW. Similar statistics are 15.9% for S-tree with an average span of 13.8 tokens.

[2]See link: http://www.itl.nist.gov/iad/mig/tests/mt/2008/doc/mt08_official_results_v0.html

Labels	total	NonProj	Percent	Avg.len
VP*	4479	920	20.5%	16.9
NP*	14164	825	5.8%	8.12
S*	3123	495	15.9%	13.8
NP-SBJ*	1284	53	4.12%	11.9

Table 9: The predicted projectable structures in MT08-NW

Using the predicted projectable structures for elm2str grammar, together with the probability defined in Eqn. 3 as additional cost, the translation results in Table 11 show it helps BLEU by 0.29 BLEU points (56.13 v.s. 55.84). The boundary decisions penalize the derivation paths using nonterminals for non-projectable spans for partial hypothesis construction.

Setups	TER	BLEUr4n4
Baseline	39.87	55.01
right-binz (rbz)	39.10	55.19
left-binz (lbz)	39.67	55.31
Head-out-left (hlbz)	39.56	55.50
Head-out-right (hrbz)	39.52	55.53
+all binzation (abz)	39.42	55.60
+regroup-verb	39.29	55.72
+deleting interior nodes $\gamma.v^i$	38.98	55.84

Table 10: TER and BLEU for MT08-NW, using only $\bar{t}(\gamma)$

5.2 Integrating $\bar{t}$ and $\bar{m}$

We carried out a series of experiments to explore the impacts using $\bar{t}$ and $\bar{m}$ for elm2str grammar. We start from transforming the trees via simple operator $\bar{t}(\gamma)$, and then expand the function with more tree context to include the neighboring functions: $\bar{t}(\gamma, \bar{m})$.

Setups	TER	BLEUr4n4
Baseline w/ $\bar{t}$	38.98	55.84
+ TM Boundaries	38.89	56.13
+ SENT Bound	38.63	56.46
all $\bar{t}(\gamma, \bar{m})$	38.61	56.87

Table 11: TER and BLEU for MT08-NW, using $\bar{t}(\gamma, \bar{m})$.

Experiments in Table 10 focus on testing operators especially binarizations for transforming the trees. In Table 10, the four possible binarization methods all improve

Data	MT08-NW	MT08-WB	Dev10-NW	Dev10-WB
Tr2Str	55.01	39.19	37.33	41.77
elm2str+$\bar{t}$	55.84	39.43	38.02	42.70
elm2str+$\bar{m}$	55.57	39.60	37.67	42.54
elm2str+$\bar{t}(\gamma, \bar{m})$	56.87	39.82	38.62	42.75

Table 12: BLEU scores on various test sets; comparing elementary tree-to-string grammar (tr2str), transformation of the trees (elm2str+$\bar{t}$), using the neighboring function for boundaries (elm2str+$\bar{m}$), and combination of all together (elm2str+$\bar{t}(\gamma, \bar{m})$). MT08-NW and MT08-WB have four references; Dev10-WB has three references, and Dev10-NW has one reference. BLEUn4 were reported.

over the baseline from +0.18 (via right-most binarization) to +0.52 (via head-out-right) BLEU points. When we combine all binarizations (abz), we did not see additive gains over the best individual case – hrbz. Because during our decoding time, we do not frequently see large number of children (maximum at 6), and for smaller trees (with three or four children), these operators will largely generate same transformed trees, and that explains the differences from these individual binarization are small. For other languages, these binarization choices might give larger differences. Additionally, regrouping the verbs is marginally helpful for BLEU and TER. Upon close examinations, we found it is usually beneficial to group verb (PV or IV) with its neighboring nodes for expressing phrases like "have to do" and "will not only". Deleting the interior nodes helps on shrinking the trees, so that we can translate it with more statistics and confidences. It helps more on TER than BLEU for MT08-NW.

Table 11 extends Table 10 with neighboring function to further disambiguate the reordering rule using the tree context. Besides the translation boundary, the reordering decisions should be different with regard to the positions of the elementary tree relative to the sentence. At the sentence-beginning one might expect more for monotone decoding, while in the middle of the sentence, one might expect more reorderings. Table 11 shows when we add such boundary markups in our rules, an improvement of 0.33 BLEU points were obtained (56.46 v.s. 56.13) on top of the already improved setups. A close check up showed that the sentence-begin/end markups significantly reduced the leading "and" (from Arabic word w#) in the decoding output. Also, the verb subject order under SBAR seems to be more like monotone with a leading pronoun, rather than the general strong reordering of moving verb after subject. Overall, our results showed that such boundary conditions are helpful for executing the correct reorderings. We conclude the investigation with full function $\bar{t}(\gamma, \bar{m})$, which leads to a BLEUr4n4 of 56.87 (cased BLEUr4n4c 55.16), a significant improvement of 1.77 BLEU point over a already strong baseline.

We apply the setups for several other NW and WEB datasets to further verify the improvement. Shown in Table 12, we apply separately the operators for $\bar{t}$ and $\bar{m}$ first, then combine them as the final results. Varied improvements were observed for different genres. On DEV10-NW, we observed 1.29 BLEU points improvement, and about 0.63 and 0.98 improved BLEU points for MT08-WB and DEV10-WB, respectively. The improvements for newwire are statistically significant. The improvements for weblog are, however, only marginally better. One possible reason is the parser quality for web genre is reliable, as our training data is all in newswire. Regarding to the individual operators proposed in this paper, we observed consistent improvements of applying them across all the datasets. The generative model in Eqn. 3 leverages the operators further by selecting the best transformed tree form for executing the reorderings.

5.3 A Translation Example

To illustrate the advantages of the proposed grammar, we use a testing case with long distance word reordering and the source side parse trees. We compare the translation from a strong phrasal decoder (DTM2) (Ittycheriah and Roukos, 2007), which is one of the top systems in NIST-08 evaluation for Arabic-English. The translations from both decoders with the same training data (LM+TM) are in Table 13. The highlighted parts in Figure 3 show that, the rules on partial trees are effectively selected and applied for capturing long-distance word reordering, which is otherwise rather difficult to get correct in a phrasal system even with a MaxEnt reordering model.

6 Discussions and Conclusions

We proposed a framework to learn models to predict how to transform an elementary tree into its simplified forms for better executing the word reorderings. Two types of operators were explored, including (a) transforming the trees via binarizations, grouping or deleting interior nodes to change the structures; and (b) neighboring boundary context to further disambiguate the reordering decisions. Significant improvements were observed on top of a strong baseline system, and consistent improvements were observed across genres; we achieved a cased BLEU of 55.16 for MT08-NW, which is significantly better than the officially reported results in NIST MT08 Arabic-English evaluations.

Src Sent	qAl AlAmyr EbdAlrHmn bn EbdAlEzyz nA}b wzyr AldfAE AlsEwdy AlsAbq fy tSryH SHAfy An +h mtfA}l b# qdrp Almmlkp Ely AyjAd Hl l# Alm$klp .
Phrasal Decoder	prince abdul rahman bin abdul aziz , deputy minister of defense former saudi said in a press statement that he was optimistic about the kingdom 's ability to find a solution to the problem .
Elm2Str+$\bar{t}(\gamma, \bar{m})$	former saudi deputy defense minister prince abdul rahman bin abdul aziz said in a press statement that he was optimistic of the kingdom 's ability to find a solution to the problem .

Table 13: A translation example, comparing with phrasal decoder.

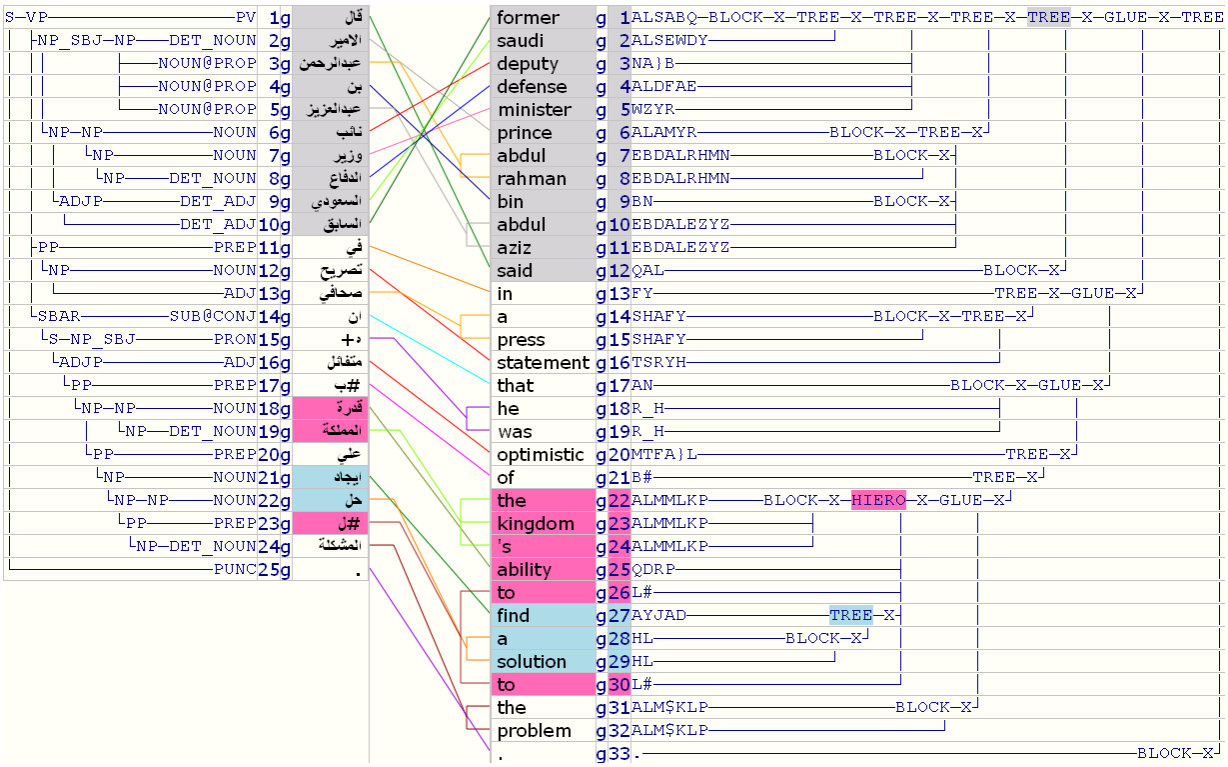


Figure 3: A testing case: illustrating the derivations from chart decoder. The left panel is source parse tree for the Arabic sentence — the input to our decoder; the right panel is the English translation together with the simplified derivation tree and alignment from our decoder output. Each "X" is a nonterminal in the grammar rule; a "Block" means a phrase pair is applied to rewrite a nonterminal; "Glue" and "Hiero" means the unlabeled rules were chosen to explain the span as explained in § 3.4.3 ; "Tree" means a labeled rule is applied for the span. For instance, for the source span [1,10], a rule is applied on a partial tree with PV and NP-SBJ; for the span [18,23], a rule is backed off to an unlabeled rule (Hiero-alike); for the span [21,22], it is another partial tree of NPs.

Within the proposed framework, we also presented several special cases including the translation boundaries for nonterminals in SCFG for translation. We achieved a high accuracy of 84.7% for predicting such boundaries using MaxEnt model on machine parse trees. Future works aim at transforming such non-projectable trees into projectable form (Eisner, 2003), driven by translation rules from aligned data(Burkett et al., 2010), and informative features form both the source [3] and the target sides (Shen et al., 2008) to enable the system to leverage more isomorphic trees, and avoid potential detour errors. We are exploring the incremental decoding framework, like (Huang and Mi, 2010), to improve pruning and speed.

Acknowledgments

This work was partially supported by the Defense Advanced Research Projects Agency under contract No. HR0011-08-C-0110. The views and findings contained in this material are those of the authors and do not necessarily reflect the position or policy of the U.S. government and no official endorsement should be inferred.

We are also very grateful to the three anonymous reviewers for their suggestions and comments.

[3] The BLEU score on MT08-NW has been improved to 57.55 since the acceptance of this paper, using the proposed technique but with our GALE P5 data pipeline and setups.

References

Peter F. Brown, Stephen A. Della Pietra, Vincent. J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. In *Computational Linguistics*, volume 19(2), pages 263–331.

David Burkett, John Blitzer, and Dan Klein. 2010. Joint parsing and alignment with weakly synchronized grammars. In *Proceedings of HLT-NAACL*, pages 127–135, Los Angeles, California, June. Association for Computational Linguistics.

Marine Carpuat, Yuval Marton, and Nizar Habash. 2010. Reordering matrix post-verbal subjects for arabic-to-english smt. In *17th Confrence sur le Traitement Automatique des Langues Naturelles*, Montral, Canada, July.

Stanley F. Chen and Stephen M. Chu. 2010. Enhanced word classing for model m. In *Proceedings of Interspeech*.

Stanley F. Chen. 2009. Shrinking exponential language models. In *Proceedings of NAACL HLT,*, pages 468–476.

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

David Chiang. 2010. Learning to translate with source and target syntax. In *Proc. ACL*, pages 1443–1452.

Chris Dyer, Hendra Setiawan, Yuval Marton, and Philip Resnik. 2009. The University of Maryland statistical machine translation system for the Fourth Workshop on Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 145–149, Athens, Greece, March.

Jason Eisner. 2003. Learning Non-Isomorphic tree mappings for Machine Translation. In *Proc. ACL-2003*, pages 205–208.

Ahmad Emami, Stanley F. Chen, Abe Ittycheriah, Hagen Soltau, and Bing Zhao. 2010. Decoding with shrinkage-based language models. In *Proceedings of Interspeech*.

Bryant Huang and Kevin Knight. 2006. Relabeling syntax trees to improve syntax-based machine translation quality. In *Proc. NAACL-HLT*, pages 240–247.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proceedings of EMNLP*, pages 273–283, Cambridge, MA, October. Association for Computational Linguistics.

Zhongqiang Huang, Martin Cmejrek, and Bowen Zhou. 2010. Soft syntactic constraints for hierarchical phrase-based translation using latent syntactic distributions. In *Proceedings of the 2010 EMNLP*, pages 138–147.

Abraham Ittycheriah and Salim Roukos. 2007. Direct translation model 2. In *Proc of HLT-07*, pages 57–64.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*, pages 177–180.

Young-Suk Lee, Bing Zhao, and Xiaoqian Luo. 2010. Constituent reordering and syntax models for english-to-japanese statistical machine translation. In *Proceedings of Coling-2010*, pages 626–634, Beijing, China, August.

Yang Liu and Qun Liu. 2010. Joint parsing and translation. In *Proceedings of COLING 2010,*, pages 707–715, Beijing, China, August.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. Spmt: Statistical machine translation with syntactified target language phraases. In *Proceedings of EMNLP-2006*, pages 44–52.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of ACL-08: HLT*, pages 1003–1011.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of EMNLP 2008*, pages 206–214.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *In Proceedings of ACL-HLT*, pages 192–199.

Franz Josef Och. 2003. Minimum error rate training in Statistical Machine Translation. In *ACL-2003*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of the ACL-02)*, pages 311–318, Philadelphia, PA, July.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585, Columbus, Ohio, June. Association for Computational Linguistics.

Libin Shen, Bing Zhang, Spyros Matsoukas, Jinxi Xu, and Ralph Weischedel. 2010. Statistical machine translation with a factorized grammar. In *Proceedings of the 2010 EMNLP*, pages 616–625, Cambridge, MA, October. Association for Computational Linguistics.

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

W. Wang, J. May, K. Knight, and D. Marcu. 2010. Restructuring, re-labeling, and re-aligning for syntax-based statistical machine translation. In *Computational Linguistics*, volume 36(2), pages 247–277.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. In *Computational Linguistics*, volume 23(3), pages 377–403.

Deyi Xiong, Min Zhang, and Haizhou Li. 2010. Learning translation boundaries for phrase-based decoding. In *NAACL-HLT 2010*, pages 136–144.

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008. A tree sequence alignment-based tree-to-tree translation model. In *ACL-HLT*, pages 559–567.

Hui Zhang, Min Zhang, Haizhou Li, Aiti Aw, and Chew Lim Tan. 2009. Forest-based tree sequence to string translation model. In *Proc. of ACL 2009*, pages 172–180.

Bing Zhao and Yaser Al-Onaizan. 2008. Generalizing local and non-local word-reordering patterns for syntax-based machine translation. In *Proceedings of EMNLP*, pages 572–581, Honolulu, Hawaii, October.

Bing Zhao and Shengyuan Chen. 2009. A simplex armijo downhill algorithm for optimizing statistical machine translation decoding parameters. In *Proceedings of HLT-NAACL*, pages 21–24, Boulder, Colorado, June. Association for Computational Linguistics.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proc. of NAACL 2006 - Workshop on SMT*, pages 138–141.

Rule Markov Models for Fast Tree-to-String Translation

Ashish Vaswani
Information Sciences Institute
University of Southern California
avaswani@isi.edu

Haitao Mi
Institute of Computing Technology
Chinese Academy of Sciences
htmi@ict.ac.cn

Liang Huang and **David Chiang**
Information Sciences Institute
University of Southern California
{lhuang,chiang}@isi.edu

Abstract

Most statistical machine translation systems rely on *composed rules* (rules that can be formed out of smaller rules in the grammar). Though this practice improves translation by weakening independence assumptions in the translation model, it nevertheless results in huge, redundant grammars, making both training and decoding inefficient. Here, we take the opposite approach, where we only use *minimal rules* (those that cannot be formed out of other rules), and instead rely on a *rule Markov model* of the derivation history to capture dependencies between minimal rules. Large-scale experiments on a state-of-the-art tree-to-string translation system show that our approach leads to a slimmer model, a faster decoder, yet the same translation quality (measured using B) as composed rules.

1 Introduction

Statistical machine translation systems typically model the translation process as a sequence of translation steps, each of which uses a translation rule, for example, a phrase pair in phrase-based translation or a tree-to-string rule in tree-to-string translation. These rules are usually applied independently of each other, which violates the conventional wisdom that translation should be done in context. To alleviate this problem, most state-of-the-art systems rely on *composed rules*, which are larger rules that can be formed out of smaller rules (including larger phrase pairs that can be formerd out of smaller phrase pairs), as opposed to *minimal rules*, which are rules that cannot be formed out of other rules. Although this approach does improve translation quality dramatically by weakening the independence assumptions in the translation model, they suffer from two main problems. First, composition can cause a combinatorial explosion in the number of rules. To avoid this, ad-hoc limits are placed during composition, like upper bounds on the number of nodes in the composed rule, or the height of the rule. Under such limits, the grammar size is manageable, but still much larger than the minimal-rule grammar. Second, due to large grammars, the decoder has to consider many more hypothesis translations, which slows it down. Nevertheless, the advantages outweigh the disadvantages, and to our knowledge, all top-performing systems, both phrase-based and syntax-based, use composed rules. For example, Galley et al. (2004) initially built a syntax-based system using only minimal rules, and subsequently reported (Galley et al., 2006) that composing rules improves B by 3.6 points, while increasing grammar size 60-fold and decoding time 15-fold.

The alternative we propose is to replace composed rules with a *rule Markov model* that generates rules conditioned on their context. In this work, we restrict a rule's context to the vertical chain of ancestors of the rule. This ancestral context would play the same role as the context formerly provided by rule composition. The dependency treelet model developed by Quirk and Menezes (2006) takes such an approach within the framework of dependency translation. However, their study leaves unanswered whether a rule Markov model can take the place of composed rules. In this work, we investigate the use of rule Markov models in the context of tree-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 856–864,
Portland, Oregon, June 19-24, 2011.

to-string translation (Liu et al., 2006; Huang et al., 2006). We make three new contributions.

First, we carry out a detailed comparison of rule Markov models with composed rules. Our experiments show that, using trigram rule Markov models, we achieve an improvement of 2.2 B over a baseline of minimal rules. When we compare against *vertically* composed rules, we find that our rule Markov model has the same accuracy, but our model is much smaller and decoding with our model is 30% faster. When we compare against *full* composed rules, we find that our rule Markov model still often reaches the same level of accuracy, again with savings in space and time.

Second, we investigate methods for pruning rule Markov models, finding that even very simple pruning criteria actually improve the accuracy of the model, while of course decreasing its size.

Third, we present a very fast decoder for tree-to-string grammars with rule Markov models. Huang and Mi (2010) have recently introduced an efficient incremental decoding algorithm for tree-to-string translation, which operates top-down and maintains a derivation history of translation rules encountered. This history is exactly the vertical chain of ancestors corresponding to the contexts in our rule Markov model, which makes it an ideal decoder for our model.

We start by describing our rule Markov model (Section 2) and then how to decode using the rule Markov model (Section 3).

2 Rule Markov models

Our model which conditions the generation of a rule on the vertical chain of its ancestors, which allows it to capture interactions between rules.

Consider the example Chinese-English tree-to-string grammar in Figure 1 and the example derivation in Figure 2. Each row is a derivation step; the tree on the left is the derivation tree (in which each node is a rule and its children are the rules that substitute into it) and the tree pair on the right is the source and target derived tree. For any derivation node r, let $anc_1(r)$ be the parent of r (or ϵ if it has no parent), $anc_2(r)$ be the grandparent of node r (or ϵ if it has no grandparent), and so on. Let $anc_1^n(r)$ be the chain of ancestors $anc_1(r) \cdots anc_n(r)$.

The derivation tree is generated as follows. With probability $\mathrm{P}(r_1 \mid \epsilon)$, we generate the rule at the root node, r_1. We then generate rule r_2 with probability $\mathrm{P}(r_2 \mid r_1)$, and so on, always taking the leftmost open substitution site on the English derived tree, and generating a rule r_i conditioned on its chain of ancestors with probability $\mathrm{P}(r_i \mid anc_1^n(r_i))$. We carry on until no more children can be generated. Thus the probability of a derivation tree T is

$$\mathrm{P}(T) = \prod_{r \in T} \mathrm{P}(r \mid anc_1^n(r)) \quad (1)$$

For the minimal rule derivation tree in Figure 2, the probability is:

$$\begin{aligned}\mathrm{P}(T) = \mathrm{P}(r_1 \mid \epsilon) \cdot \mathrm{P}(r_2 \mid r_1) \cdot \mathrm{P}(r_3 \mid r_1) \\ \cdot \mathrm{P}(r_4 \mid r_1, r_3) \cdot \mathrm{P}(r_6 \mid r_1, r_3, r_4) \\ \cdot \mathrm{P}(r_7 \mid r_1, r_3, r_4) \cdot \mathrm{P}(r_5 \mid r_1, r_3) \quad (2)\end{aligned}$$

Training We run the algorithm of Galley et al. (2004) on word-aligned parallel text to obtain a single derivation of minimal rules for each sentence

pair. (Unaligned words are handled by attaching them to the highest node possible in the parse tree.) The rule Markov model

can then be trained on the path set of these derivation trees.

Smoothing We use interpolation with absolute discounting (Ney et al., 1994):

$$\begin{aligned}\mathrm{P}_{abs}(r \mid anc_1^n(r)) = \frac{\max\left\{c(r \mid anc_1^n(r)) - D_n, 0\right\}}{\sum_{r'} c(r' \mid anc_1^n(r'))} \\ + \ (1 - \lambda_n)\mathrm{P}_{abs}(r \mid anc_1^{n-1}(r)), \quad (3)\end{aligned}$$

where $c(r \mid anc_1^n(r))$ is the number of times we have seen rule r after the vertical context $anc_1^n(r)$, D_n is the discount for a context of length n, and $(1 - \lambda_n)$ is set to the value that makes the smoothed probability distribution sum to one.

We experiment with bigram and trigram rule Markov models. For each, we try different values of D_1 and D_2, the discount for bigrams and trigrams, respectively. Ney et al. (1994) suggest using the following value for the discount D_n:

$$D_n = \frac{n_1}{n_1 + n_2} \quad (4)$$

rule id	translation rule
r_1	IP(x_1:NP x_2:VP) → x_1 x_2
r_2	NP(Bùshí) → Bush
r_3	VP(x_1:PP x_2:VP) → x_2 x_1
r_4	PP(x_1:P x_2:NP) → x_1 x_2
r_5	VP(VV(jǔxíng) AS(le) NPB(huìtán)) → held talks
r_6	P(yǔ) → with
r'_6	P(yǔ) → and
r_7	NP(Shālóng) → Sharon

Figure 1: Example tree-to-string grammar.

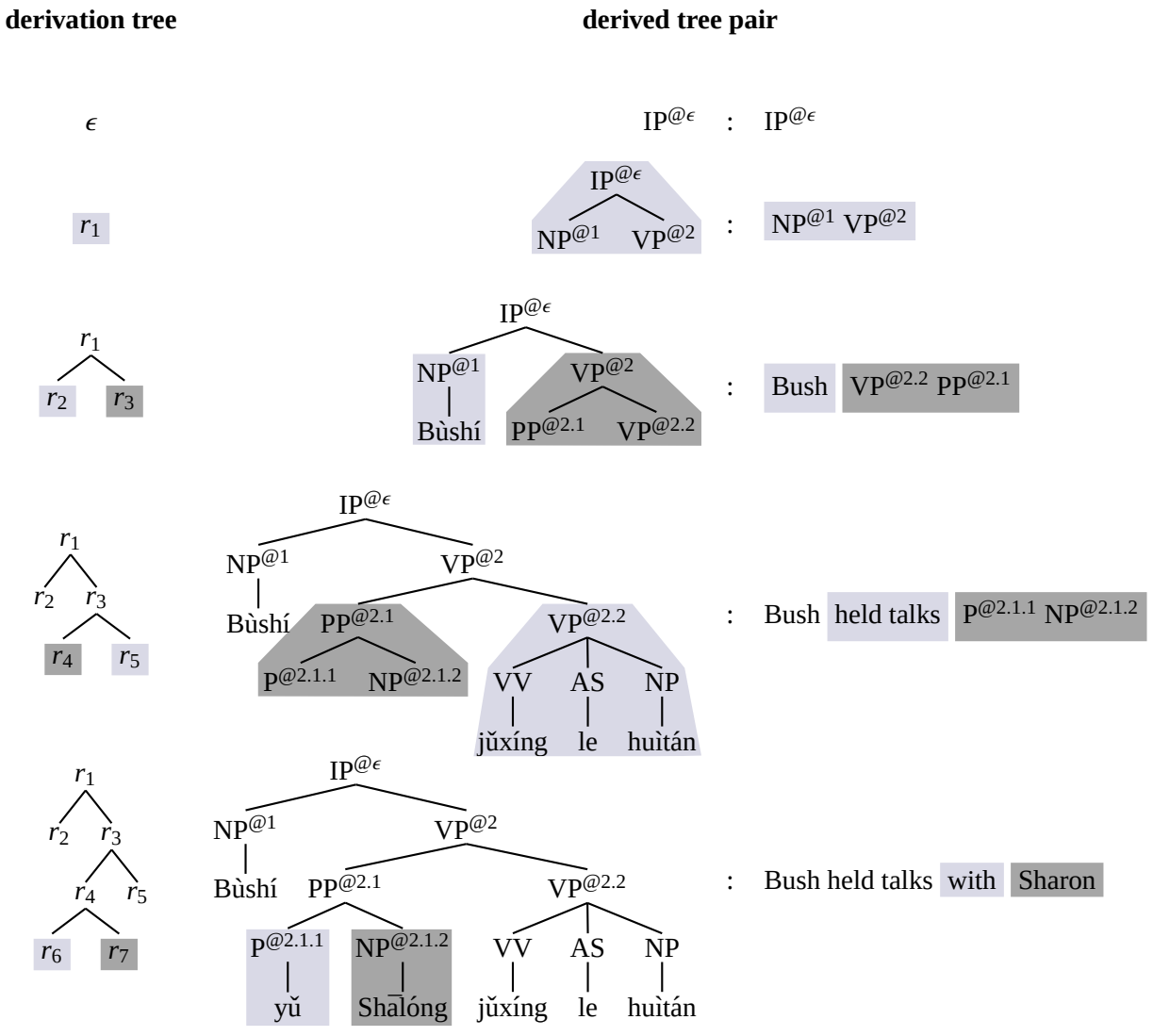


Figure 2: Example tree-to-string derivation. Each row shows a rewriting step; at each step, the leftmost nonterminal symbol is rewritten using one of the rules in Figure 1.

Here, n_1 and n_2 are the total number of n-grams with exactly one and two counts, respectively. For our corpus, $D_1 = 0.871$ and $D_2 = 0.902$. Additionally, we experiment with 0.4 and 0.5 for D_n.

Pruning In addition to full n-gram Markov models, we experiment with three approaches to build smaller models to investigate if pruning helps. Our results will show that smaller models indeed give a higher B score than the full bigram and trigram models. The approaches we use are:

- RM-A: We keep only those contexts in which more than P unique rules were observed. By optimizing on the development set, we set $P = 12$.
- RM-B: We keep only those contexts that were observed more than P times. Note that this is a superset of RM-A. Again, by optimizing on the development set, we set $P = 12$.
- RM-C: We try a more principled approach for learning variable-length Markov models inspired by that of Bejerano and Yona (1999), who learn a Prediction Suffix Tree (PST). They grow the PST in an iterative manner by starting from the root node (no context), and then add contexts to the tree. A context is added if the KL divergence between its predictive distribution and that of its parent is above a certain threshold and the probability of observing the context is above another threshold.

3 Tree-to-string decoding with rule Markov models

In this paper, we use our rule Markov model framework in the context of tree-to-string translation. Tree-to-string translation systems (Liu et al., 2006; Huang et al., 2006) have gained popularity in recent years due to their speed and simplicity. The input to the translation system is a source parse tree and the output is the target string. Huang and Mi (2010) have recently introduced an efficient incremental decoding algorithm for tree-to-string translation. The decoder operates top-down and maintains a derivation history of translation rules encountered. The history is exactly the vertical chain of ancestors corresponding to the contexts in our rule Markov model. This

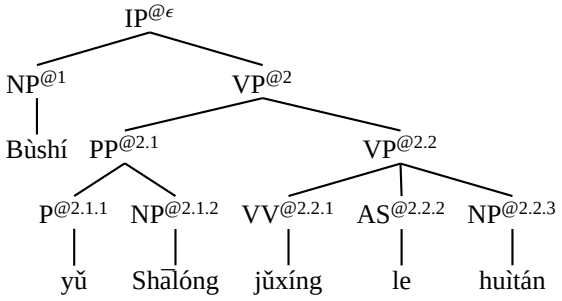


Figure 3: Example input parse tree with tree addresses.

makes incremental decoding a natural fit with our generative story. In this section, we describe how to integrate our rule Markov model into this incremental decoding algorithm. Note that it is also possible to integrate our rule Markov model with other decoding algorithms, for example, the more common non-incremental top-down/bottom-up approach (Huang et al., 2006), but it would involve a non-trivial change to the decoding algorithms to keep track of the vertical derivation history, which would result in significant overhead.

Algorithm Given the input parse tree in Figure 3, Figure 4 illustrates the search process of the incremental decoder with the grammar of Figure 1. We write $X^{@\eta}$ for a tree node with label X at tree address η (Shieber et al., 1995). The root node has address ϵ, and the ith child of node η has address $\eta.i$. At each step, the decoder maintains a stack of active rules, which are rules that have not been completed yet, and the rightmost $(n-1)$ English words translated thus far (the hypothesis), where n is the order of the word language model (in Figure 4, $n = 2$). The stack together with the translated English words comprise a state of the decoder. The last column in the figure shows the rule Markov model probabilities with the conditioning context. In this example, we use a trigram rule Markov model.

After initialization, the process starts at step 1, where we *predict* rule r_1 (the shaded rule) with probability $\mathrm{P}(r_1 \mid \epsilon)$ and push its English side onto the stack, with variables replaced by the corresponding tree nodes: x_1 becomes $\mathrm{NP}^{@1}$ and x_2 becomes $\mathrm{VP}^{@2}$. This gives us the following stack:

$$s = [\mathbf{.}\ \mathrm{NP}^{@1}\ \mathrm{VP}^{@2}]$$

The dot (**.**) indicates the next symbol to process in

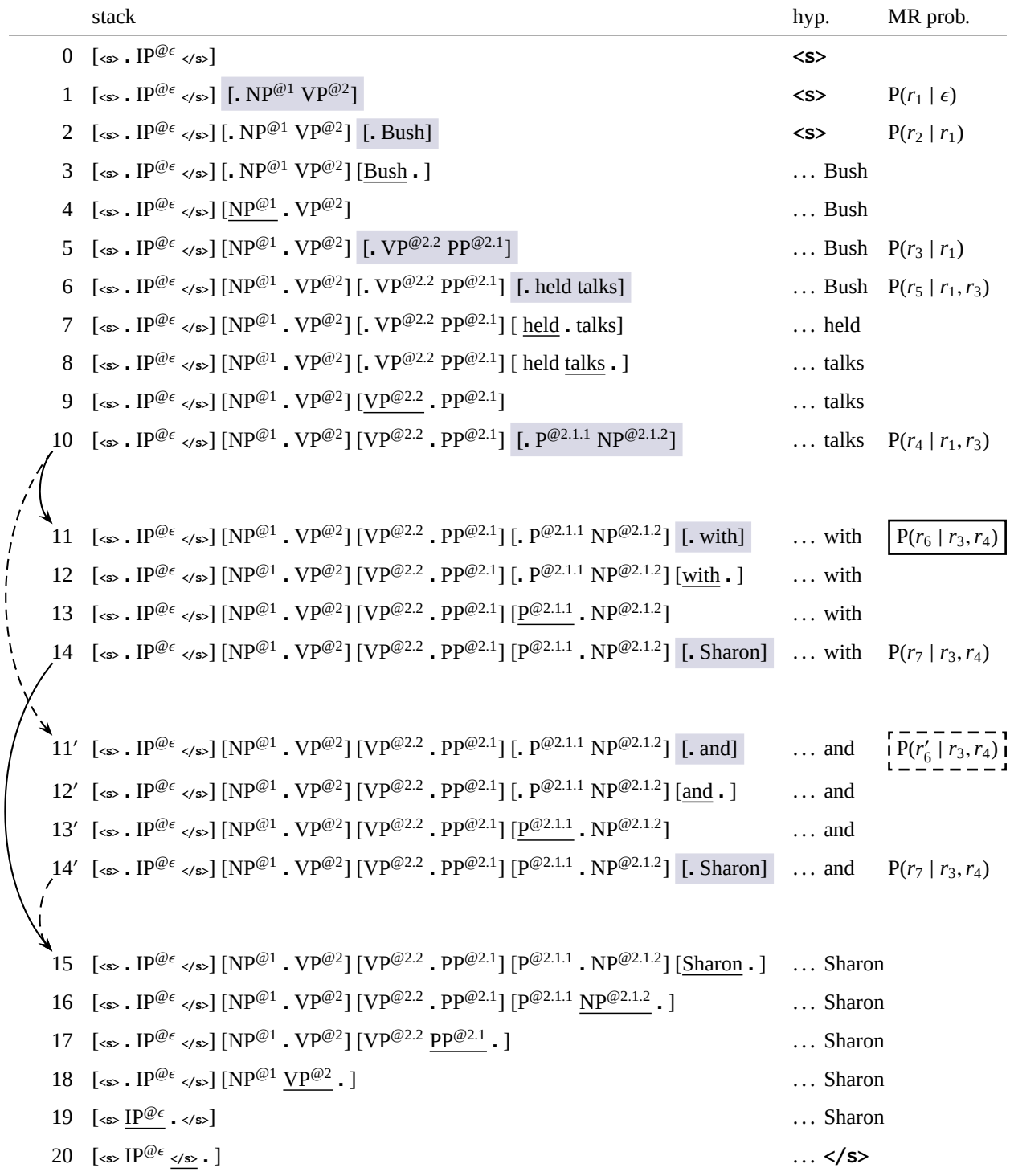


Figure 4: Simulation of incremental decoding with rule Markov model. The solid arrows indicate one path and the dashed arrows indicate an alternate path.

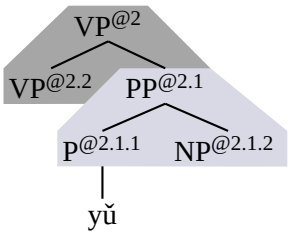


Figure 5: Vertical context r_3 r_4 which allows the model to correctly translate *yǔ* as *with*.

the English word order. We expand node $\text{NP}^{@1}$ first with English word order. We then predict lexical rule r_2 with probability $\text{P}(r_2 \mid r_1)$ and push rule r_2 onto the stack:

$$[\text{. NP}^{@1}\ \text{VP}^{@2}\ ]\ [\text{. Bush}]$$

In step 3, we perform a *scan* operation, in which we append the English word just after the dot to the current hypothesis and move the dot after the word. Since the dot is at the end of the top rule in the stack, we perform a *complete* operation in step 4 where we pop the finished rule at the top of the stack. In the *scan* and *complete* steps, we don't need to compute rule probabilities.

An interesting branch occurs after step 10 with two competing lexical rules, r_6 and r'_6. The Chinese word *yǔ* can be translated as either a preposition *with* (leading to step 11) or a conjunction *and* (leading to step 11′). The word *n*-gram model does not have enough information to make the correct choice, *with*. As a result, good translations might be pruned because of the beam. However, our rule Markov model has the correct preference because of the conditioning ancestral sequence (r_3, r_4), shown in Figure 5. Since $\text{VP}^{@2.2}$ has a preference for *yǔ* translating to *with*, our corpus statistics will give a higher probability to $\text{P}(r_6 \mid r_3, r_4)$ than $\text{P}(r'_6 \mid r_3, r_4)$. This helps the decoder to score the correct translation higher.

Complexity analysis With the incremental decoding algorithm, adding rule Markov models does not change the time complexity, which is $O(nc|V|^{g-1})$, where n is the sentence length, c is the maximum number of incoming hyperedges for each node in the translation forest, V is the target-language vocabulary, and g is the order of the n-gram language model (Huang and Mi, 2010). However, if one were to use rule Markov models with a conventional CKY-style bottom-up decoder (Liu et al., 2006), the complexity would increase to $O(nC^{m-1}|V|^{4(g-1)})$, where C is the maximum number of outgoing hyperedges for each node in the translation forest, and m is the order of the rule Markov model.

4 Experiments and results

4.1 Setup

The training corpus consists of 1.5M sentence pairs with 38M/32M words of Chinese/English, respectively. Our development set is the newswire portion of the 2006 NIST MT Evaluation test set (616 sentences), and our test set is the newswire portion of the 2008 NIST MT Evaluation test set (691 sentences).

We word-aligned the training data using GIZA++ followed by link deletion (Fossum et al., 2008), and then parsed the Chinese sentences using the Berkeley parser (Petrov and Klein, 2007). To extract tree-to-string translation rules, we applied the algorithm of Galley et al. (2004). We trained our rule Markov model on derivations of minimal rules as described above. Our trigram word language model was trained on the target side of the training corpus using the SRILM toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing. The base feature set for all systems is similar to the set used in Mi et al. (2008). The features are combined into a standard log-linear model, which we trained using minimum error-rate training (Och, 2003) to maximize the B score on the development set.

At decoding time, we again parse the input sentences using the Berkeley parser, and convert them into translation forests using rule pattern-matching (Mi et al., 2008). We evaluate translation quality using case-insensitive IBM B -4, calculated by the script `mteval-v13a.pl`.

4.2 Results

Table 1 presents the main results of our paper. We used grammars of minimal rules and composed rules of maximum height 3 as our baselines. For decoding, we used a beam size of 50. Using the best bigram rule Markov models and the minimal rule grammar gives us an improvement of 1.5 B over the minimal rule baseline. Using the best trigram rule Markov model brings our gain up to 2.3 B .

grammar	rule Markov model	max rule height	parameters ($\times 10^6$) full	parameters ($\times 10^6$) dev+test	B test	time (sec/sent)
minimal	None	3	4.9	0.3	24.2	1.2
	RM-B bigram	3	4.9+4.7	0.3+0.5	25.7	1.8
	RM-A trigram	3	4.9+7.6	0.3+0.6	26.5	2.0
vertical composed	None	7	176.8	1.3	26.5	2.9
composed	None	3	17.5	1.6	26.4	2.2
	None	7	448.7	3.3	27.5	6.8
	RM-A trigram	7	448.7+7.6	3.3+1.0	28.0	9.2

Table 1: Main results. Our trigram rule Markov model strongly outperforms minimal rules, and performs at the same level as composed and vertically composed rules, but is smaller and faster. The number of parameters is shown for both the full model and the model filtered for the concatenation of the development and test sets (dev+test).

These gains are statistically significant with $p < 0.01$, using bootstrap resampling with 1000 samples (Koehn, 2004). We find that by just using bigram context, we are able to get at least 1 B point higher than the minimal rule grammar. It is interesting to see that using just bigram rule interactions can give us a reasonable boost. We get our highest gains from using trigram context where our best performing rule Markov model gives us 2.3 B points over minimal rules. This suggests that using longer contexts helps the decoder to find better translations.

We also compared rule Markov models against composed rules. Since our models are currently limited to conditioning on vertical context, the closest comparison is against *vertically* composed rules. We find that our approach performs equally well using much less time and space.

Comparing against *full* composed rules, we find that our system matches the score of the baseline composed rule grammar of maximum height 3, while using many fewer parameters. (It should be noted that a parameter in the rule Markov model is just a floating-point number, whereas a parameter in the composed-rule system is an entire rule; therefore the difference in memory usage would be even greater.) Decoding with our model is 0.2 seconds faster per sentence than with composed rules.

These experiments clearly show that rule Markov models with minimal rules increase translation quality significantly and with lower memory requirements than composed rules. One might wonder if the best performance can be obtained by combining composed rules with a rule Markov model. This is straightforward to implement: the rule Markov model is still defined over derivations of minimal rules, but in the decoder's prediction step, the rule Markov model's value on a composed rule is calculated by decomposing it into minimal rules and computing the product of their probabilities. We find that using our best trigram rule Markov model with composed rules gives us a 0.5 B gain on top of the composed rule grammar, statistically significant with $p < 0.05$, achieving our highest score of 28.0.[1]

rule Markov model	D_1	B dev	time (sec/sent)
RM-A	0.871	29.2	1.8
RM-B	0.4	29.9	1.8
RM-C	0.871	29.8	1.8
RM-Full	0.4	29.7	1.9

Table 2: For rule bigrams, RM-B with $D_1 = 0.4$ gives the best results on the development set.

rule Markov model	D_1	D_2	B dev	time (sec/sent)
RM-A	0.5	0.5	30.3	2.0
RM-B	0.5	0.5	29.9	2.0
RM-C	0.5	0.5	30.1	2.0
RM-Full	0.4	0.5	30.1	2.2

Table 3: For rule bigrams, RM-A with $D_1, D_2 = 0.5$ gives the best results on the development set.

4.3 Analysis

Tables 2 and 3 show how the various types of rule Markov models compare, for bigrams and trigrams,

[1] For this experiment, a beam size of 100 was used.

parameters ($\times 10^6$) dev/test	B dev/test without RMM	with RMM	time (sec/sent) without/with RMM
2.6	31.0/27.0	31.1/27.4	4.5/7.0
2.9	31.5/27.7	31.4/27.3	5.6/8.1
3.3	31.4/27.5	31.4/28.0	6.8/9.2

Table 6: Adding rule Markov models to composed-rule grammars improves their translation performance.

D_2	D_1 0.4	0.5	0.871
0.4	30.0	30.0	
0.5	29.3	30.3	
0.902			30.0

Table 4: RM-A is robust to different settings of D_n on the development set.

parameters ($\times 10^6$) dev+test	B dev	test	time (sec/sent)
1.2	30.2	26.1	2.8
1.3	30.1	26.5	2.9
1.3	30.1	26.2	3.2

Table 5: Comparison of vertically composed rules using various settings (maximum rule height 7).

respectively. It is interesting that the full bigram and trigram rule Markov models do not give our highest B scores; pruning the models not only saves space but improves their performance. We think that this is probably due to overfitting.

Table 4 shows that the RM-A trigram model does fairly well under all the settings of D_n we tried. Table 5 shows the performance of *vertically* composed rules at various settings. Here we have chosen the setting that gives the best performance on the test set for inclusion in Table 1.

Table 6 shows the performance of *fully* composed rules and fully composed rules with a rule Markov Model at various settings.[2] In the second line (2.9 million rules), the drop in B score resulting from adding the rule Markov model is not statistically significant.

5 Related Work

Besides the Quirk and Menezes (2006) work discussed in Section 1, there are two other previous efforts both using a rule bigram model in machine translation, that is, the probability of the current rule only depends on the immediate previous rule in the vertical context, whereas our rule Markov model can condition on longer and sparser derivation histories. Among them, Ding and Palmer (2005) also use a dependency treelet model similar to Quirk and Menezes (2006), and Liu and Gildea (2008) use a tree-to-string model more like ours. Neither compared to the scenario with composed rules.

Outside of machine translation, the idea of weakening independence assumptions by modeling the derivation history is also found in parsing (Johnson, 1998), where rule probabilities are conditioned on parent and grand-parent nonterminals. However, besides the difference between parsing and translation, there are still two major differences. First, our work conditions rule probabilities on parent and grandparent *rules*, not just nonterminals. Second, we compare against a composed-rule system, which is analogous to the Data Oriented Parsing (DOP) approach in parsing (Bod, 2003). To our knowledge, there has been no direct comparison between a history-based PCFG approach and DOP approach in the parsing literature.

6 Conclusion

In this paper, we have investigated whether we can eliminate composed rules without any loss in translation quality. We have developed a rule Markov model that captures vertical bigrams and trigrams of minimal rules, and tested it in the framework of tree-to-string translation. We draw three main conclusions from our experiments. First, our rule Markov models dramatically improve a grammar of minimal rules, giving an improvement of 2.3 B . Second, when we compare against *vertically* composed rules we are able to get about the same B score, but our model is much smaller and decoding with our

[2] For these experiments, a beam size of 100 was used.

model is faster. Finally, when we compare against full composed rules, we find that we can reach the same level of performance under some conditions, but in order to do so consistently, we believe we need to extend our model to condition on horizontal context in addition to vertical context. We hope that by modeling context in both axes, we will be able to completely replace composed-rule grammars with smaller minimal-rule grammars.

Acknowledgments

We would like to thank Fernando Pereira, Yoav Goldberg, Michael Pust, Steve DeNeefe, Daniel Marcu and Kevin Knight for their comments. Mi's contribution was made while he was visiting USC/ISI. This work was supported in part by DARPA under contracts HR0011-06-C-0022 (subcontract to BBN Technologies), HR0011-09-1-0028, and DOI-NBC N10AP20031, by a Google Faculty Research Award to Huang, and by the National Natural Science Foundation of China under contracts 60736014 and 90920004.

References

Gill Bejerano and Golan Yona. 1999. Modeling protein families using probabilistic suffix trees. In *Proc. RECOMB*, pages 15–24. ACM Press.

Rens Bod. 2003. An efficient implementation of a new DOP model. In *Proceedings of EACL*, pages 19–26.

Yuan Ding and Martha Palmer. 2005. Machine translation using probablisitic synchronous dependency insertion grammars. In *Proceedings of ACL*, pages 541–548.

Victoria Fossum, Kevin Knight, and Steve Abney. 2008. Using syntax to improve word alignment precision for syntax-based machine translation. In *Proceedings of the Workshop on Statistical Machine Translation*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT-NAACL*, pages 273–280.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of COLING-ACL*, pages 961–968.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proceedings of EMNLP*, pages 273–283.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of AMTA*, pages 66–73.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24:613–632.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, pages 388–395.

Ding Liu and Daniel Gildea. 2008. Improved tree-to-string transducer for machine translation. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 62–69.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of COLING-ACL*, pages 609–616.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL: HLT*, pages 192–199.

H. Ney, U. Essen, and R. Kneser. 1994. On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8:1–38.

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

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of HLT-NAACL*, pages 404–411.

Chris Quirk and Arul Menezes. 2006. Do we need phrases? Challenging the conventional wisdom in statistical machine translation. In *Proceedings of NAACL HLT*, pages 9–16.

Stuart Shieber, Yves Schabes, and Fernando Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–36.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of ICSLP*, volume 30, pages 901–904.

A Hierarchical Pitman-Yor Process HMM for Unsupervised Part of Speech Induction

Phil Blunsom
Department of Computer Science
University of Oxford
Phil.Blunsom@cs.ox.ac.uk

Trevor Cohn
Department of Computer Science
University of Sheffield
T.Cohn@dcs.shef.ac.uk

Abstract

In this work we address the problem of unsupervised part-of-speech induction by bringing together several strands of research into a single model. We develop a novel hidden Markov model incorporating sophisticated smoothing using a hierarchical Pitman-Yor processes prior, providing an elegant and principled means of incorporating lexical characteristics. Central to our approach is a new type-based sampling algorithm for hierarchical Pitman-Yor models in which we track fractional table counts. In an empirical evaluation we show that our model consistently out-performs the current state-of-the-art across 10 languages.

1 Introduction

Unsupervised part-of-speech (PoS) induction has long been a central challenge in computational linguistics, with applications in human language learning and for developing portable language processing systems. Despite considerable research effort, progress in fully unsupervised PoS induction has been slow and modern systems barely improve over the early Brown et al. (1992) approach (Christodoulopoulos et al., 2010). One popular means of improving tagging performance is to include supervision in the form of a tag dictionary or similar, however this limits portability and also comprimises any cognitive conclusions. In this paper we present a novel approach to fully unsupervised PoS induction which uniformly outperforms the existing state-of-the-art across all our corpora in 10 different languages. Moreover, the performance of our unsupervised model approaches that of many existing semi-supervised systems, despite our method not receiving any human input.

In this paper we present a Bayesian hidden Markov model (HMM) which uses a non-parametric prior to infer a latent tagging for a sequence of words. HMMs have been popular for unsupervised PoS induction from its very beginnings (Brown et al., 1992), and justifiably so, as the most discriminating feature for deciding a word's PoS is its local syntactic context.

Our work brings together several strands of research including Bayesian non-parametric HMMs (Goldwater and Griffiths, 2007), Pitman-Yor language models (Teh, 2006b; Goldwater et al., 2006b), tagging constraints over word types (Brown et al., 1992) and the incorporation of morphological features (Clark, 2003). The result is a non-parametric Bayesian HMM which avoids overfitting, contains no free parameters, and exhibits good scaling properties. Our model uses a hierarchical Pitman-Yor process (PYP) prior to affect sophisicated smoothing over the transition and emission distributions. This allows the modelling of sub-word structure, thereby capturing tag-specific morphological variation. Unlike many existing approaches, our model is a principled generative model and does not include any hand tuned language specific features.

Inspired by previous successful approaches (Brown et al., 1992), we develop a new type-level inference procedure in the form of an MCMC sampler with an approximate method for incorporating the complex dependencies that arise between jointly sampled events. Our experimental evaluation demonstrates that our model, particularly when restricted to a single tag per type, produces

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 865–874,
Portland, Oregon, June 19-24, 2011.

state-of-the-art results across a range of corpora and languages.

2 Background

Past research in unsupervised PoS induction has largely been driven by two different motivations: a task based perspective which has focussed on inducing word classes to improve various applications, and a linguistic perspective where the aim is to induce classes which correspond closely to annotated part-of-speech corpora. Early work was firmly situtated in the task-based setting of improving generalisation in language models. Brown et al. (1992) presented a simple first-order HMM which restricted word types to always be generated from the same class. Though PoS induction was not their aim, this restriction is largely validated by empirical analysis of treebanked data, and moreover conveys the significant advantage that all the tags for a given word type can be updated at the same time, allowing very efficient inference using the exchange algorithm. This model has been popular for language modelling and bilingual word alignment, and an implementation with improved inference called `mkcls` (Och, 1999)[1] has become a standard part of statistical machine translation systems.

The HMM ignores orthographic information, which is often highly indicative of a word's part-of-speech, particularly so in morphologically rich languages. For this reason Clark (2003) extended Brown et al. (1992)'s HMM by incorporating a character language model, allowing the modelling of limited morphology. Our work draws from these models, in that we develop a HMM with a one class per tag restriction and include a character level language model. In contrast to these previous works which use the maximum likelihood estimate, we develop a Bayesian model with a rich prior for smoothing the parameter estimates, allowing us to move to a trigram model.

A number of researchers have investigated a semi-supervised PoS induction task in which a tag dictionary or similar data is supplied a priori (Smith and Eisner, 2005; Haghighi and Klein, 2006; Goldwater and Griffiths, 2007; Toutanova and Johnson, 2008; Ravi and Knight, 2009). These systems achieve much higher accuracy than fully unsupervised systems, though it is unclear whether the tag dictionary assumption has real world application. We focus solely on the fully unsupervised scenario, which we believe is more practical for text processing in new languages and domains.

Recent work on unsupervised PoS induction has focussed on encouraging sparsity in the emission distributions in order to match empirical distributions derived from treebank data (Goldwater and Griffiths, 2007; Johnson, 2007; Gao and Johnson, 2008). These authors took a Bayesian approach using a Dirichlet prior to encourage sparse distributions over the word types emitted from each tag. Conversely, Ganchev et al. (2010) developed a technique to optimize the more desirable reverse property of the word types having a sparse posterior distribution over tags. Recently Lee et al. (2010) combined the one class per word type constraint (Brown et al., 1992) in a HMM with a Dirichlet prior to achieve both forms of sparsity. However this work approximated the derivation of the Gibbs sampler (omitting the interdependence between events when sampling from a collapsed model), resulting in a model which underperformed Brown et al. (1992)'s one-class HMM.

Our work also seeks to enforce both forms of sparsity, by developing an algorithm for type-level inference under the one class constraint. This work differs from previous Bayesian models in that we explicitly model a complex backoff path using a hierachical prior, such that our model jointly infers distributions over tag trigrams, bigrams and unigrams and whole words and their character level representation. This smoothing is critical to ensure adequate generalisation from small data samples.

Research in language modelling (Teh, 2006b; Goldwater et al., 2006a) and parsing (Cohn et al., 2010) has shown that models employing Pitman-Yor priors can significantly outperform the more frequently used Dirichlet priors, especially where complex hierarchical relationships exist between latent variables. In this work we apply these advances to unsupervised PoS tagging, developing a HMM smoothed using a Pitman-Yor process prior.

[1] Available from `http://fjoch.com/mkcls.html`.

3 The PYP-HMM

We develop a trigram hidden Markov model which models the joint probability of a sequence of latent tags, $\mathbf{t}$, and words, $\mathbf{w}$, as

$$P_\theta(\mathbf{t}, \mathbf{w}) = \prod_{l=1}^{L+1} P_\theta(t_l|t_{l-1}, t_{l-2}) P_\theta(w_l|t_l)\,,$$

where $L = |\mathbf{w}| = |\mathbf{t}|$ and $t_0 = t_{-1} = t_{L+1} = \$$ are assigned a sentinel value to denote the start or end of the sentence. A key decision in formulating such a model is the smoothing of the tag trigram and emission distributions, which would otherwise be too difficult to estimate from small datasets. Prior work in unsupervised PoS induction has employed simple smoothing techniques, such as additive smoothing or Dirichlet priors (Goldwater and Griffiths, 2007; Johnson, 2007), however this body of work has overlooked recent advances in smoothing methods used for language modelling (Teh, 2006b; Goldwater et al., 2006b). Here we build upon previous work by developing a PoS induction model smoothed with a sophisticated non-parametric prior. Our model uses a hierarchical Pitman-Yor process prior for both the transition and emission distributions, encoding a backoff path from complex distributions to succesSsively simpler ones. The use of complex distributions (e.g., over tag trigrams) allows for rich expressivity when sufficient evidence is available, while the hierarchy affords a means of backing off to simpler and more easily estimated distributions otherwise. The PYP has been shown to generate distributions particularly well suited to modelling language (Teh, 2006a; Goldwater et al., 2006b), and has been shown to be a generalisation of Kneser-Ney smoothing, widely recognised as the best smoothing method for language modelling (Chen and Goodman, 1996).

The model is depicted in the plate diagram in Figure 1. At its centre is a standard trigram HMM, which generates a sequence of tags and words,

$$\begin{aligned} t_l|t_{l-1}, t_{l-2}, T &\sim T_{t_{l-1},t_{l-2}} \\ w_l|t_l, E &\sim E_{t_l}\,. \end{aligned}$$

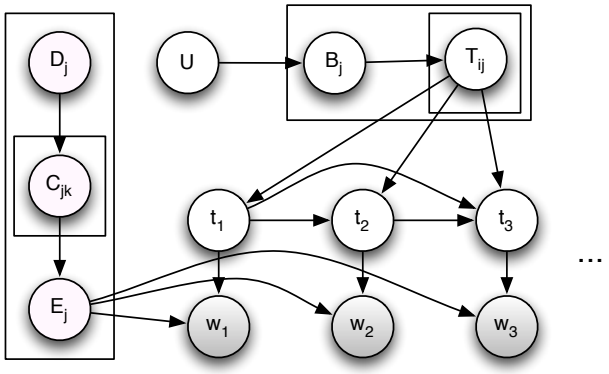


Figure 1: Plate diagram representation of the trigram HMM. The indexes i and j range over the set of tags and k ranges over the set of characters. Hyper-parameters have been omitted from the figure for clarity.

The trigram transition distribution, T_{ij}, is drawn from a hierarchical PYP prior which backs off to a bigram B_j and then a unigram U distribution,

$$\begin{aligned} T_{ij}|a^T, b^T, B_j &\sim \text{PYP}(a^T, b^T, B_j) \\ B_j|a^B, b^B, U &\sim \text{PYP}(a^B, b^B, U) \\ U|a^U, b^U &\sim \text{PYP}(a^U, b^U, \text{Uniform})\,, \end{aligned}$$

where the prior over U has as its base distribition a uniform distribution over the set of tags, while the priors for B_j and T_{ij} back off by discarding an item of context. This allows the modelling of trigram tag sequences, while smoothing these estimates with their corresponding bigram and unigram distributions. The degree of smoothing is regulated by the hyper-parameters a and b which are tied across each length of n-gram; these hyper-parameters are inferred during training, as described in 3.1.

The tag-specific emission distributions, E_j, are also drawn from a PYP prior,

$$E_j|a^E, b^E, C \sim \text{PYP}(a^E, b^E, C_j)\,.$$

We consider two different settings for the base distribution C_j: 1) a simple uniform distribution over the vocabulary (denoted HMM for the experiments in section 4); and 2) a character-level language model (denoted HMM+LM). In many languages morphological regularities correlate strongly with a word's part-of-speech (e.g., suffixes in English), which we hope to capture using a basic character language model. This model was inspired by Clark (2003)

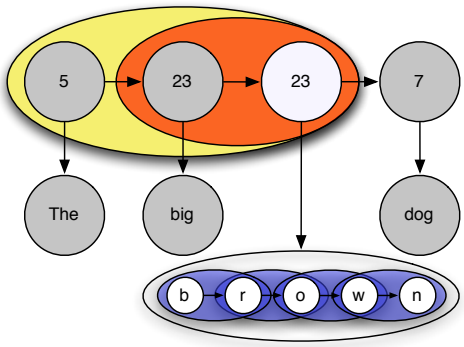


Figure 2: The conditioning structure of the hierarchical PYP with an embedded character language models.

who applied a character level distribution to the single class HMM (Brown et al., 1992). We formulate the character-level language model as a bigram model over the character sequence comprising word w_l,

$$\begin{aligned} w_{lk}|w_{lk-1}, t_l, C &\sim C_{t_l w_{lk-1}} \\ C_{jk}|a^C, b^C, D_j &\sim \text{PYP}(a^C, b^C, D_j) \\ D_j|a^D, b^D &\sim \text{PYP}(a^D, b^D, \text{Uniform})\,, \end{aligned}$$

where k indexes the characters in the word and, in a slight abuse of notation, the character itself, w_0 and is set to a special sentinel value denoting the start of the sentence (ditto for a final end of sentence marker) and the uniform base distribution ranges over the set of characters. We expect that the HMM+LM model will outperform the uniform HMM as it can capture many consistent morphological affixes and thereby better distinguish between different parts-of-speech. The HMM+LM is shown in Figure 2, illustrating the decomposition of the tag sequence into n-grams and a word into its component character bigrams.

3.1 Training

In order to induce a tagging under this model we use Gibbs sampling, a Markov chain Monte Carlo (MCMC) technique for drawing samples from the posterior distribution over the tag sequences given observed word sequences. We present two different sampling strategies: First, a simple Gibbs sampler which randomly samples an update to a single tag given all other tags; and second, a type-level sampler which updates all tags for a given word under a one-tag-per-word-type constraint. In order to extract a single tag sequence to test our model against the gold standard we find the tag at each site with maximum marginal probability in the sample set.

Following standard practice, we perform inference using a collapsed sampler whereby the model parameters U, B, T, E and C are marginalised out. After marginalisation the posterior distribution under a PYP prior is described by a variant of the Chinese Restaurant Process (CRP). The CRP is based around the analogy of a restaurant with an infinite number of tables, with customers entering one at a time and seating themselves at a table. The choice of table is governed by

$$P(z_l = k|\mathbf{z}_{-l}) = \begin{cases} \frac{n_k^- - a}{l-1+b} & 1 \le k \le K^- \\ \frac{K^- a + b}{l-1+b} & k = K^- + 1 \end{cases} \quad (1)$$

where z_l is the table chosen by the lth customer, $\mathbf{z}_{-l}$ is the seating arrangement of the $l-1$ previous customers, n_k^- is the number of customers in $\mathbf{z}_{-l}$ who are seated at table k, $K^- = K(\mathbf{z}_{-l})$ is the total number of tables in $\mathbf{z}_{-l}$, and $z_1 = 1$ by definition. The arrangement of customers at tables defines a clustering which exhibits a power-law behavior controlled by the hyperparameters a and b.

To complete the restaurant analogy, a dish is then served to each table which is shared by all the customers seated there. This corresponds to a draw from the base distribution, which in our case ranges over tags for the transition distribution, and words for the observation distribution. Overall the PYP leads to a distribution of the form

$$P^T(t_l = i|\mathbf{z}_{-l}, \mathbf{t}_{-l}) = \frac{1}{n_{\mathbf{h}}^- + b^T} \times \quad (2)$$
$$\left(n_{\mathbf{h}i}^- - K_{\mathbf{h}i}^- a^T + \left(K_{\mathbf{h}}^- a^T + b^T\right) P^B(i|\mathbf{z}_{-l}, \mathbf{t}_{-l})\right),$$

illustrating the trigram transition distribution, where $\mathbf{t}_{-l}$ are all previous tags, $\mathbf{h} = (t_{l-2}, t_{l-1})$ is the conditioning bigram, $n_{\mathbf{h}i}^-$ is the count of the trigram $\mathbf{h}i$ in $\mathbf{t}_{-l}$, $n_{\mathbf{h}}^-$ the total count over all trigrams beginning with $\mathbf{h}$, $K_{\mathbf{h}i}^-$ the number of tables served dish i and $P^B(\cdot)$ is the base distribution, in this case the bigram distribution.

A hierarchy of PYPs can be formed by making the base distribution of a PYP another PYP, following a

semantics whereby whenever a customer sits at an empty table in a restaurant, a new customer is also said to enter the restaurant for its base distribution. That is, each table at one level is equivalent to a customer at the next deeper level, creating the invariants: $K^-_{\mathbf{h}i} = n^-_{\mathbf{u}i}$ and $K^-_{\mathbf{u}i} = n^-_i$, where $\mathbf{u} = t_{l-1}$ indicates the unigram backoff context of $\mathbf{h}$. The recursion terminates at the lowest level where the base distribution is static. The hierarchical setting allows for the modelling of elaborate backoff paths from rich and complex structure to successively simpler structures.

Gibbs samplers Both our Gibbs samplers perform the same calculation of conditional tag distributions, and involve first decrementing all trigrams and emissions affected by a sampling action, and then reintroducing the trigrams one at a time, conditioning their probabilities on the updated counts and table configurations as we progress.

The first local Gibbs sampler (PYP-HMM) updates a single tag assignment at a time, in a similar fashion to Goldwater and Griffiths (2007). Changing one tag affects three trigrams, with posterior

$$P(t_l|\mathbf{z}_{-l}, \mathbf{t}_{-l}, \mathbf{w}) \propto P(\mathbf{t}_{l\pm2}, w_l|\mathbf{z}_{-l\pm2}, \mathbf{t}_{-l\pm2}) \,,$$

where $l\pm2$ denotes the range $l-2, l-1, l, l+1, l+2$. The joint distribution over the three trigrams contained in $\mathbf{t}_{l\pm2}$ can be calculated using the PYP formulation. This calculation is complicated by the fact that these events are not independent; the counts of one trigram can affect the probability of later ones, and moreover, the table assignment for the trigram may also affect the bigram and unigram counts, of particular import when the same tag occurs twice in a row such as in Figure 2.

Many HMMs used for inducing word classes for language modelling include the restriction that all occurrences of a word type always appear with the same class throughout the corpus (Brown et al., 1992; Och, 1999; Clark, 2003). Our second sampler (PYP-1HMM) restricts inference to taggings which adhere to this one tag per type restriction. This restriction permits efficient inference techniques in which all tags of all occurrences of a word type are updated in parallel. Similar techniques have been used for models with Dirichlet priors (Liang et al., 2010), though one must be careful to manage the dependencies between multiple draws from the posterior.

The dependency on table counts in the conditional distributions complicates the process of drawing samples for both our models. In the non-hierarchical model (Goldwater and Griffiths, 2007) these dependencies can easily be accounted for by incrementing customer counts when such a dependence occurs. In our model we would need to sum over all possible table assignments that result in the same tagging, at all levels in the hierarchy: tag trigrams, bigrams and unigrams; and also words, character bigrams and character unigrams. To avoid this rather onerous marginalisation[2] we instead use expected table counts to calculate the conditional distributions for sampling. Unfortunately we know of no efficient algorithm for calculating the expected table counts, so instead develop a novel approximation

$$\mathrm{E}_{n+1}\left[K_i\right] \approx \mathrm{E}_n\left[K_i\right] + \frac{(a^U \mathrm{E}_n\left[K\right] + b^U) P_0(i)}{(n - \mathrm{E}_n\left[K_i\right] b^U) + (a^U \mathrm{E}_n\left[K\right] + b^U) P_0(i)} \,, \quad (3)$$

where K_i is the number of tables for the tag unigram i of which there are $n+1$ occurrences, $\mathrm{E}_n\left[\cdot\right]$ denotes an expectation after observing n items and $\mathrm{E}_n\left[K\right] = \sum_j \mathrm{E}_n\left[K_j\right]$. This formulation defines a simple recurrence starting with the first customer seated at a table, $\mathrm{E}_1\left[K_i\right] = 1$, and as each subsequent customer arrives we *fractionally* assign them to a new table based on their conditional probability of sitting alone. These fractional counts are then carried forward for subsequent customers.

This approximation is tight for small n, and therefore it should be effective in the case of the local Gibbs sampler where only three trigrams are being resampled. For the type based resampling where large numbers of n are involved (consider resampling *the*), this approximation can deviate from the actual value due to errors accumulated in the recursion. Figure 3 illustrates a simulation demonstrating that the approximation is a close match for small a and n but underestimates the true value for high a

[2] Marginalisation is intractable in general, i.e. for the 1HMM where many sites are sampled jointly.

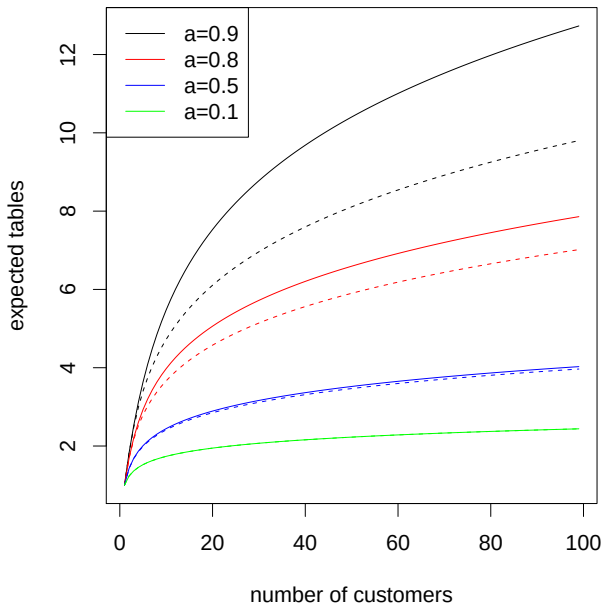


Figure 3: Simulation comparing the expected table count (solid lines) versus the approximation under Eq. 3 (dashed lines) for various values of a. This data was generated from a single PYP with $b = 1, P_0(i) = \frac{1}{4}$ and $n = 100$ customers which all share the same tag.

and n. The approximation was much less sensitive to the choice of b (not shown).

To resample a sequence of trigrams we start by removing their counts from the current restaurant configuration (resulting in $\mathbf{z}_-$). For each tag we simulate adding back the trigrams one at a time, calculating their probability under the given $\mathbf{z}_-$ plus the fractional table counts accumulated by Equation 3. We then calculate the expected table count contribution from this trigram and add it to the accumulated counts. The fractional table count from the trigram then results in a fractional customer entering the bigram restaurant, and so on down to unigrams. At each level we must update the expected counts before moving on to the next trigram. After performing this process for all trigrams under consideration and for all tags, we then normalise the resulting tag probabilities and sample an outcome. Once a tag has been sampled, we then add all the trigrams to the restaurants sampling their tables assignments explicitly (which are no longer fractional), recorded in $\mathbf{z}$. Because we do not marginalise out the table counts and our expectations are only approximate, this sampler will be biased. We leave to future work properly accounting for this bias, e.g., by devising a Metropolis Hastings acceptance test.

Sampling hyperparameters We treat the hyper-parameters $\{(a^x, b^x), x \in (U, B, T, E, C)\}$ as random variables in our model and infer their values. We place prior distributions on the PYP discount a^x and concentration b^x hyperparamters and sample their values using a slice sampler. For the discount parameters we employ a uniform Beta distribution ($a^x \sim \text{Beta}(1, 1)$), and for the concentration parameters we use a vague gamma prior ($b^x \sim \text{Gamma}(10, 0.1)$). All the hyper-parameters are resampled after every 5th sample of the corpus.

The result of this hyperparameter inference is that there are no user tunable parameters in the model, an important feature that we believe helps explain its consistently high performance across test settings.

4 Experiments

We perform experiments with a range of corpora to both investigate the properties of our proposed models and inference algorithms, as well as to establish their robustness across languages and domains. For our core English experiments we report results on the entire Penn. Treebank (Marcus et al., 1993), while for other languages we use the corpora made available for the CoNLL-X Shared Task (Buchholz and Marsi, 2006). We report results using the many-to-one (M-1) and v-measure (VM) metrics considered best by the evaluation of Christodoulopoulos et al. (2010). M-1 measures the accuracy of the model after mapping each predicted class to its most frequent corresponding tag, while VM is a variant of the F-measure which uses conditional entropy analogies of precision and recall. The log-posterior for the HMM sampler levels off after a few hundred samples, so we report results after five hundred. The 1HMM sampler converges more quickly so we use two hundred samples for these models. All reported results are the mean of three sampling runs.

An important detail for any unsupervised learning algorithm is its initialisation. We used slightly different initialisation for each of our inference strategies. For the unrestricted HMM we randomly assigned each word token to a class. For the restricted 1HMM we use a similar initialiser to

Model	M-1	VM
Prototype meta-model (CGS10)	76.1	68.8
MEMM (BBDK10)	75.5	-
`mkcls` (Och, 1999)	73.7	65.6
MLE 1HMM-LM (Clark, 2003)*	71.2	65.5
BHMM (GG07)	63.2	56.2
PR (Ganchev et al., 2010)*	62.5	54.8
Trigram PYP-HMM	69.8	62.6
Trigram PYP-1HMM	76.0	68.0
Trigram PYP-1HMM-LM	**77.5**	69.7
Bigram PYP-HMM	66.9	59.2
Bigram PYP-1HMM	72.9	65.9
Trigram DP-HMM	68.1	60.0
Trigram DP-1HMM	76.0	68.0
Trigram DP-1HMM-LM	76.8	**69.8**

Table 1: WSJ performance comparing previous work to our own model. The columns display the many-to-1 accuracy and the V measure, both averaged over 5 independent runs. Our model was run with the local sampler (HMM), the type-level sampler (1HMM) and also with the character LM (1HMM-LM). Also shown are results using Dirichlet Process (DP) priors by fixing $\mathbf{a} = 0$. The system abbreviations are CGS10 (Christodoulopoulos et al., 2010), BBDK10 (Berg-Kirkpatrick et al., 2010) and GG07 (Goldwater and Griffiths, 2007). Starred entries denote results reported in CGS10.

Clark (2003), assigning each of the k most frequent word types to its own class, and then randomly dividing the rest of the types between the classes.

As a baseline we report the performance of `mkcls` (Och, 1999) on all test corpora. This model seems not to have been evaluated in prior work on unsupervised PoS tagging, which is surprising given its consistently good performance.

First we present our results on the most frequently reported evaluation, the WSJ sections of the Penn. Treebank, along with a number of state-of-the-art results previously reported (Table 1). All of these models are allowed 45 tags, the same number of tags as in the gold-standard. The performance of our models is strong, particularly the 1HMM. We also see that incorporating a character language model (1HMM-LM) leads to further gains in performance, improving over the best reported scores under both M-1 and VM. We have omitted the results for the HMM-LM as experimentation showed that the local Gibbs sampler became hopelessly stuck, failing to mix due to the model's deep structure (its peak performance was $\approx 55\%$).

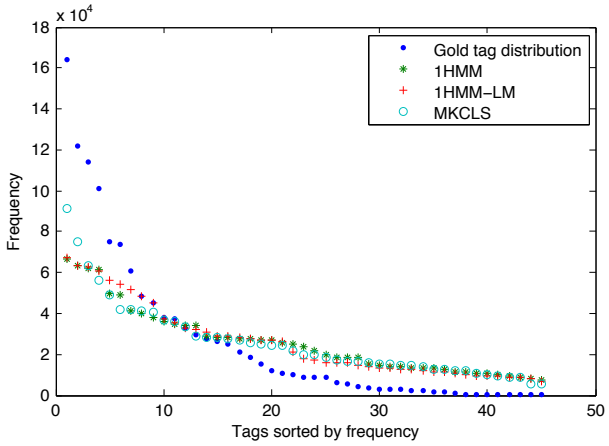


Figure 4: Sorted frequency of tags for WSJ. The gold standard distribution follows a steep exponential curve while the induced model distributions are more uniform.

To evaluate the effectiveness of the PYP prior we include results using a Dirichlet Process prior (DP). We see that for all models the use of the PYP provides some gain for the HMM, but diminishes for the 1HMM. This is perhaps a consequence of the expected table count approximation for the type-sampled PYP-1HMM: the DP relies less on the table counts than the PYP.

If we restrict the model to bigrams we see a considerable drop in performance. Note that the bigram PYP-HMM outperforms the closely related BHMM (the main difference being that we smooth tag bigrams with unigrams). It is also interesting to compare the bigram PYP-1HMM to the closely related model of Lee et al. (2010). That model incorrectly assumed independence of the conditional sampling distributions, resulting in a accuracy of 66.4%, well below that of our model.

Figures 4 and 5 provide insight into the behavior of the sampling algorithms. The former shows that both our models and `mkcls` induce a more uniform distribution over tags than specified by the treebank. It is unclear whether it is desirable for models to exhibit behavior closer to the treebank, which dedicates separate tags to very infrequent phenomena while lumping the large range of noun types into a single category. The graph in Figure 5 shows that the type-based 1HMM sampler finds a good tagging extremely quickly and then sticks with it,

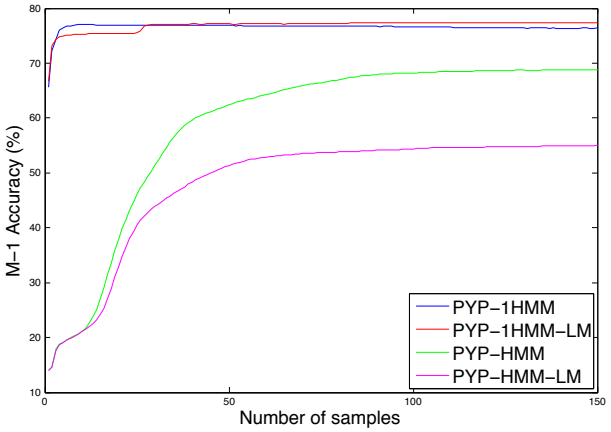


Figure 5: M-1 accuracy vs. number of samples.

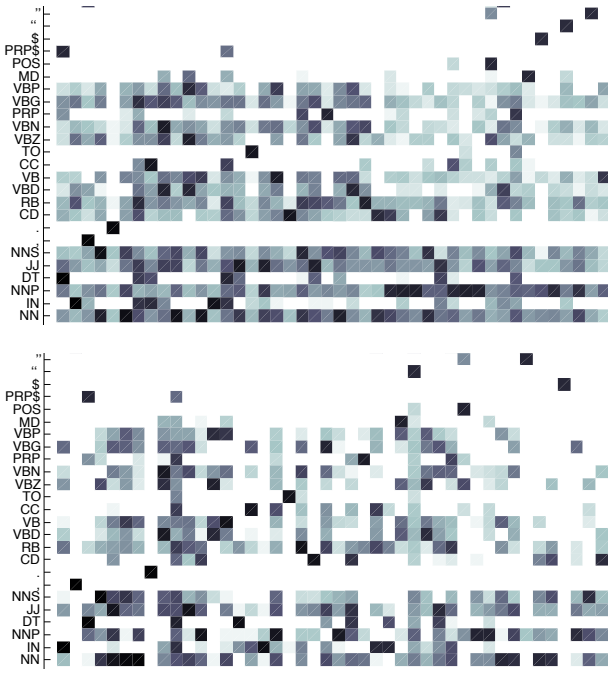


Figure 6: Cooccurence between frequent gold (y-axis) and predicted (x-axis) tags, comparing `mkcls` (top) and PYP-1HMM-LM (bottom). Both axes are sorted in terms of frequency. Darker shades indicate more frequent cooccurence and columns represent the induced tags.

save for the occasional step change demonstrated by the 1HMM-LM line. The locally sampled model is far slower to converge, rising slowly and plateauing well below the other models.

In Figure 6 we compare the distributions over WSJ tags for `mkcls` and the PYP-1HMM-LM. On the macro scale we can see that our model induces a sparser distribution. With closer inspection we can identify particular improvements our model makes. In the first column for `mkcls` and the third column for our model we can see similar classes with significant counts for DTs and PRPs, indicating a class that the models may be using to represent the start of sentences (informed by start transitions or capitalisation). This column exemplifies the sparsity of the PYP model's posterior.

We continue our evaluation on the CoNLL multilingual corpora (Table 2). These results show a highly consistent story of performance for our models across diverse corpora. In all cases the PYP-1HMM outperforms the PYP-HMM, which are both outperformed by the PYP-1HMM-LM. The character language model provides large gains in performance on a number of corpora, in particular those with rich morphology (Arabic +5%, Portuguese +5%, Spanish +4%). We again note the strong performance of the `mkcls` model, significantly beating recently published state-of-the-art results for both Dutch and Swedish. Overall our best model (PYP-1HMM-LM) outperforms both the state-of-the-art, where previous work exists, as well as `mkcls` consistently across all languages.

5 Discussion

The hidden Markov model, originally developed by Brown et al. (1992), continues to be an effective modelling structure for PoS induction. We have combined hierarchical Bayesian priors with a trigram HMM and character language model to produce a model with consistently state-of-the-art performance across corpora in ten languages. However our analysis indicates that there is still room for improvement, particularly in model formulation and developing effective inference algorithms.

Induced tags have already proven their usefulness in applications such as Machine Translation, thus it will prove interesting as to whether the improvements seen from our models can lead to gains in downstream tasks. The continued successes of models combining hierarchical Pitman-Yor priors with expressive graphical models attests to this framework's enduring attraction, we foresee continued interest in applying this technique to other NLP tasks.

Language	mkcls	HMM	1HMM	1HMM-LM	Best pub.	Tokens	Tag types
Arabic	58.5	57.1	62.7	**67.5**	-	54,379	20
Bulgarian	66.8	67.8	69.7	**73.2**	-	190,217	54
Czech	59.6	62.0	66.3	**70.1**	-	1,249,408	12^c
Danish	62.7	69.9	73.9	**76.2**	66.7*	94,386	25
Dutch	64.3	66.6	68.7	**70.4**	67.3†	195,069	13^c
Hungarian	54.3	65.9	69.0	**73.0**	-	131,799	43
Portuguese	68.5	72.1	73.5	**78.5**	75.3*	206,678	22
Spanish	63.8	71.6	74.7	**78.8**	73.2*	89,334	47
Swedish	64.3	66.6	67.0	**68.6**	60.6†	191,467	41

Table 2: Many-to-1 accuracy across a range of languages, comparing our model with `mkcls` and the best published result (*Berg-Kirkpatrick et al. (2010) and †Lee et al. (2010)). This data was taken from the CoNLL-X shared task training sets, resulting in listed corpus sizes. Fine PoS tags were used for evaluation except for items marked with ^{c}, which used the coarse tags. For each language the systems were trained to produce the same number of tags as the gold standard.

References

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590, Los Angeles, California, June. Association for Computational Linguistics.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Comput. Linguist.*, 18:467–479, December.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 149–164, Morristown, NJ, USA. Association for Computational Linguistics.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318, Morristown, NJ, USA. Association for Computational Linguistics.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two decades of unsupervised POS induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 575–584, Cambridge, MA, October. Association for Computational Linguistics.

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *Proceedings of the tenth Annual Meeting of the European Association for Computational Linguistics (EACL)*, pages 59–66.

Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2010. Inducing tree-substitution grammars. *Journal of Machine Learning Research*, pages 3053–3096.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 99:2001–2049, August.

Jianfeng Gao and Mark Johnson. 2008. A comparison of bayesian estimators for unsupervised hidden markov model pos taggers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 344–352, Morristown, NJ, USA. Association for Computational Linguistics.

Sharon Goldwater and Tom Griffiths. 2007. A fully bayesian approach to unsupervised part-of-speech tagging. In *Proc. of the 45th Annual Meeting of the ACL (ACL-2007)*, pages 744–751, Prague, Czech Republic, June.

Sharon Goldwater, Tom Griffiths, and Mark Johnson. 2006a. Contextual dependencies in unsupervised word segmentation. In *Proc. of the 44th Annual Meeting of the ACL and 21st International Conference on Computational Linguistics (COLING/ACL-2006)*, Sydney.

Sharon Goldwater, Tom Griffiths, and Mark Johnson. 2006b. Interpolating between types and tokens by estimating power-law generators. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural*

Information Processing Systems 18, pages 459–466. MIT Press, Cambridge, MA.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 320–327, Morristown, NJ, USA. Association for Computational Linguistics.

Mark Johnson. 2007. Why doesnt EM find good HMM POS-taggers? In *Proc. of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP-2007)*, pages 296–305, Prague, Czech Republic.

Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. 2010. Simple type-level unsupervised pos tagging. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 853–861, Morristown, NJ, USA. Association for Computational Linguistics.

P. Liang, M. I. Jordan, and D. Klein. 2010. Type-based MCMC. In *North American Association for Computational Linguistics (NAACL)*.

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

Franz Josef Och. 1999. An efficient method for determining bilingual word classes. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 71–76, Morristown, NJ, USA. Association for Computational Linguistics.

Sujith Ravi and Kevin Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of the Joint Conferenceof the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*, pages 504–512.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 354–362, Ann Arbor, Michigan, June.

Y. W. Teh. 2006a. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 985–992.

Yee Whye Teh. 2006b. A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 985–992, Morristown, NJ, USA. Association for Computational Linguistics.

Kristina Toutanova and Mark Johnson. 2008. A bayesian lda-based model for semi-supervised part-of-speech tagging. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1521–1528. MIT Press, Cambridge, MA.

Using Deep Morphology to Improve Automatic Error Detection in Arabic Handwriting Recognition

Nizar Habash and **Ryan M. Roth**
Center for Computational Learning Systems
Columbia University
{habash,ryanr}@ccls.columbia.edu

Abstract

Arabic handwriting recognition (HR) is a challenging problem due to Arabic's connected letter forms, consonantal diacritics and rich morphology. In this paper we isolate the task of identification of erroneous words in HR from the task of producing corrections for these words. We consider a variety of linguistic (morphological and syntactic) and non-linguistic features to automatically identify these errors. Our best approach achieves a roughly $\sim$15% absolute increase in F-score over a simple but reasonable baseline. A detailed error analysis shows that linguistic features, such as lemma (i.e., citation form) models, help improve HR-error detection precisely where we expect them to: semantically incoherent error words.

1 Introduction

After years of development, optical character recognition (OCR) for Latin-character languages, such as English, has been refined greatly. Arabic, however, possesses a complex orthography and morphology that makes OCR more difficult (Märgner and Abed, 2009; Halima and Alimi, 2009; Magdy and Darwish, 2006). Because of this, only a few systems for Arabic OCR of printed text have been developed, and these have not been thoroughly evaluated (Märgner and Abed, 2009). OCR of Arabic handwritten text (handwriting recognition, or HR), whether online or offline, is even more challenging compared to printed Arabic OCR, where the uniformity of letter shapes and other factors allow for easier recognition (Biadsy et al., 2006; Natarajan et al., 2008; Saleem et al., 2009).

OCR and HR systems are often improved by performing post-processing; these are attempts to evaluate whether each word, phrase or sentence in the OCR/HR output is legal and/or probable. When an illegal word or phrase is discovered (error detection), these systems usually attempt to generate a legal alternative (error correction). In this paper, we present a HR error *detection* system that uses deep lexical and morphological feature models to locate possible "problem zones" – words or phrases that are likely incorrect – in Arabic HR output. We use an off-the-shelf HR system (Natarajan et al., 2008; Saleem et al., 2009) to generate an N-best list of hypotheses for each of several scanned segments of Arabic handwriting. Our problem zone detection (PZD) system then tags the potentially erroneous (problem) words. A subsequent HR post-processing system can then focus its effort on these words when generating additional alternative hypotheses. We only discuss the PZD system and not the task of new hypothesis generation; the evaluation is on error/problem identification. PZD can also be useful in highlighting erroneous text for human post-editors.

This paper is structured as follows: Section 2 provides background on the difficulties of the Arabic HR task. Section 3 presents an analysis of HR errors and defines what is considered a problem zone to be tagged. The experimental features, data and other variables are outlined in Section 4. The experiments are presented and discussed in Section 5. We discuss and compare to some related work in detail in Section 6. Conclusions and suggested avenues of for future progress are presented in Section 7.

2 Arabic Handwriting Recognition Challenges

Arabic has several orthographic and morphological properties that make HR challenging (Darwish and Oard, 2002; Magdy and Darwish, 2006; Märgner and Abed, 2009).

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 875–884,
Portland, Oregon, June 19-24, 2011.

2.1 Arabic Orthography Challenges

The use of cursive, connected script creates problems in that it becomes more difficult for a machine to distinguish between individual characters. This is certainly not a property unique to Arabic; methods developed for other cursive script languages (such as Hidden Markov Models) can be applied successfully to Arabic (Natarajan et al., 2008; Saleem et al., 2009; Märgner and Abed, 2009; Lu et al., 1999).

Arabic writers often make use of elongation (tatweel/kashida) to beautify the script. Arabic also contains certain ligature constructions that require consideration during OCR/HR (Darwish and Oard, 2002). Sets of dots and optional diacritic markers are used to create character distinctions in Arabic. However, trace amounts of dust or dirt on the original document scan can be easily mistaken for these markers (Darwish and Oard, 2002). Alternatively, these markers in handwritten text may be too small, light or closely-spaced to readily distinguish, causing the system to drop them entirely. While Arabic disconnective letters may make it hard to determine word boundaries, they could plausibly contribute to reduced ambiguity of otherwise similar shapes.

2.2 Arabic Morphology Challenges

Arabic words can be described in terms of their morphemes. In addition to concatenative prefixes and suffixes, Arabic has templatic morphemes called *roots* and *patterns*. For example, the word وكمكاتبهم *wkmkAtbhm*[1] (*w+k+mkAtb+hm*) 'and like their offices' has two prefixes and one suffix, in addition to a stem composed of the root كتب *k-t-b* 'writing related' and the pattern *m1A23*.[2] Arabic words can also be described in terms of lexemes and inflectional features. The set of word forms that only vary inflectionally among each other is called the *lexeme*. A *lemma* is a particular word form used to represent the lexeme word set – a citation form that stands in for the class (Habash, 2010). For instance, the lemma مكتب *mktb* 'office' represents the class of all forms sharing the core meaning 'office': مكاتب *mkAtb* 'offices' (irregular plural), المكتب *Almktb* 'the office', لمكتبها *lmktbhA* 'for her office', and so on.

Just as the lemma abstracts over inflectional morphology, the root abstracts over both inflectional and derivational morphology and thus provides a very high level of lexical abstraction, indicating the "core" meaning of the word. The Arabic root كتب *k-t-b* 'writing related', e.g., relates words like مكتب *mktb* 'office', مكتوب *mktwb* 'letter', and كتيبة *ktybħ* 'military unit (of conscripts)'.

Arabic morphology allows for tens of billions of potential, legal words (Magdy and Darwish, 2006; Moftah et al., 2009). The large potential vocabulary size by itself complicates HR methods that rely on conventional, word-based dictionary lookup strategies. In this paper we consider the value of morpho-lexical and morpho-syntactic features such as lemmas and part-of-speech tags, respectively, that may allow machine learning algorithms to learn generalizations. We do not consider the root since it has been shown to be too general for NLP purposes (Larkey et al., 2007). Other researchers have used stems for OCR correction (Magdy and Darwish, 2006); we discuss their work and compare to it in Section 6, but we do not present a direct experimental comparison.

3 Problem Zones in Handwriting Recognition

3.1 HR Error Classifications

We can classify three types of HR errors: substitutions, insertions and deletions. Substitutions involve replacing the correct word by another incorrect form. Insertions are words that are incorrectly added into the HR hypothesis. An insertion error is typically paired with a substitution error, where the two errors reflect a mis-identification of a single word as two words. Deletions are simply missing words. Examples of these different types of errors appear in Table 1. In the dev set that we study here (see Section 4.1), 25.8% of the words are marked as problematic. Of these, 87.2% are letter-based words (henceforth words), as opposed to 9.3% punctuation and 3.5% digits.

Orthogonally, 81.4% of all problem words are substitution errors, 10.6% are insertion errors and 7.9% are deletion errors. Whereas punctuation symbols are 9.3% of all errors, they represent over 38%

[1] All Arabic transliterations are presented in the HSB transliteration scheme (Habash et al., 2007).

[2] The digits in the pattern correspond to positions where root radicals are inserted.

REF	!	الجودة	عالية	زبدة		علبة	تصنعوا	أن	والأندلس		وفارس	القسطنطينية	يافاتحي	المسلمون	أيها	أعجزتم		
	!	Aljwdħ	ςAlyħ	zbdħ		ςlbħ	tSnςwA	Ân	wAlÂndls		wfArs	AlqsTnTynyħ	yAfAtHy	Almslmwn	ÂyhA	Âςjztm		
HYP		الجودة	عالية	يوه	ن	عليه	تصرفوا	ان	لمن	والاخذ	وفارس	القسطنطينية	باتانى	المسلمون	ايها	ثم	أعير	
		Aljwdħ	ςAlyħ	ywh	n	ςlyh	tSrfwA	An	lmn	wAlAxð	wfArs	AlqsTnTynyħ	bAtAný	Almslmwn	AyhA	θm	Âςyr	
PZD		PROB DELX	OK	PROB INS	PROB SUB	PROB DOTS	PROB SUB	PROB ORTH	PROB INS	PROB SUB	OK	OK	PROB SUB	OK	PROB ORTH	PROB INS	PROB SUB	

Table 1: An example highlighting the different types of Arabic HR errors. The first row shows the reference sentence (right-to-left). The second row shows an automatically generated hypothesis of the same sentence. The last row shows which words in the hypothesis are marked as *problematic* (PROB) by the system and the specific category of the problem (illustrative, not used by system): SUB (substituted), ORTH (substituted by an orthographic variant), DOTS (substituted by a word with different dotting), INS (inserted), and DELX (adjacent to a deleted word). The remaining words are tagged as OK. The reference translates as '*Are you unable O'Moslems, you who conquered Constantinople and Persia and Andalusia, to manufacture a tub of high quality butter!*'. The hypothesis roughly translates as '*I loan then O'Moslems Pattani Constantinople, and Persia and taking from whom that you spend on him N Yeoh high quality*'.

of all deletion errors, almost 22% of all insertion errors and less than 5% of substitution errors. Similarly digits, which are 3.5% of all errors, are almost 14% of deletions, 7% of insertions and just over 2% of all substitutions. Punctuation and digits bring different challenges: whereas punctuation marks are a small class, their shape is often confusable with Arabic letters or letter components, e.g., إ *Ǎ* and **!** or ر *r* and **,**. Digits on the other hand are a hard class to language model since the vocabulary (of multi-digit numbers) is infinite. Potentially this can be addressed using a pattern-based model that captures forms of digit sequences (such as date and currency formats); we leave this as future work.

Words (non-digit, non-punctuation) still constitute the majority in every category of error: 47.7% of deletions, 71.3% of insertions and over 93% of substitutions. Among substitutions, 26.5% are simple orthographic variants that are often normalized in Arabic NLP because they result from frequent inconsistencies in spelling: Alef Hamza forms (ٱأإآ *A/Â/Ǎ/Ā*) and Ya/Alef-Maqsura (ى/ي *y/ý*). If we consider whether the lemma of the correct word and its incorrect form are matchable, an additional 6.9% can be added to the orthographic variant sum (since all of these cases can share the same lemmas). The rest of the cases, or 59.7% of the words, involve complex orthographic errors. Simple dot misplacement can only account for 2.4% of all substitution errors. The HR system output does not contain any illegal non-words since its vocabulary is restricted by its training data and language models. The large proportion of errors involving lemma differences is consistent with the perception that most OCR/HR errors create semantically incoherent sentences. This suggests that lemma models can be helpful in identifying such errors.

3.2 Problem Zone Definition

Prior to developing a model for PZD, it is necessary to define what is considered a 'problem'. Once a definition is chosen, gold problem tags can be generated for the training and test data by comparing the hypotheses to their references.[3] We decided in this paper to use a simple binary *problem* tag: a hypothesis word is tagged as "PROB" if it is the result of an insertion or substitution of a word. Deleted words in a hypothesis, which cannot be tagged themselves, cause their adjacent words to be marked as PROB instead. In this way, a subsequent HR post-processing system can be alerted to the possibility of a missing word via its surroundings (hence the idea of a problem 'zone'). Any words not marked as PROB are given an "OK" tag (see the PZD row of Table 1). We describe in Section 5.6 some preliminary experiments we conducted using more fine-grained tags.

4 Experimental Settings

4.1 Training and Evaluation Data

The data used in this paper is derived from image scans provided by the Linguistic Data Consortium (LDC) (Strassel, 2009). This data consists of high-resolution (600 dpi) handwriting scans of Arabic text taken from newswire articles, web logs and

[3]For clarity, we refer to these tags as 'gold', whereas the correct segment for a given hypothesis set is called the 'reference'.

newsgroups, along with ground truth annotations and word bounding box information. The scans include variations in scribe demographic background, writing instrument, paper and writing speed.

The BBN Byblos HR system (Natarajan et al., 2008; Saleem et al., 2009) is then used to process these scanned images into sequences of *segments* (sentence fragments). The system generates a ranked N-best list of hypotheses for each segment, where N could be as high as 300. On average, a segment has 6.87 words (including punctuation).

We divide the N-best list data into training, development (dev) and test sets.[4] For training, we consider two sets of size 2000 and 4000 segments (S) with the 10 top-ranked hypotheses (H) for each segment to provide additional variations.[5] The references are also included in the training sets to provide examples of perfect text. The dev and test sets use 500 segments with one top-ranked hypothesis each {H=1}. We can construct a trivial PZD baseline by assuming all the input words are PROBs; this results in baseline % Precision/Recall/F-scores of 25.8/100/41.1 and 26.0/100/41.2 for the dev and test sets, respectively. Note that in this paper we eschew these baselines in favor of comparison to a non-trivial baseline generated by a simple PZD model.

4.2 PZD Models and Features

The PZD system relies on a set of SVM classifiers trained using morphological and lexical features. The SVM classifiers are built using Yamcha (Kudo and Matsumoto, 2003). The SVMs use a quadratic polynomial kernel. For the models presented in this paper, the static feature window context size is set to +/- 2 words; the previous two (dynamic) classifications (i.e. targets) are also used as features. Experiments with smaller window sizes result in poorer performance, while a larger window size (+/- 6 words) yields roughly the same performance at the expense of an order-of-magnitude increase in required training time. Over 30 different combinations of features were considered. Table 2 shows the individual feature definitions.

Simple	**Description**
`word`	The surface word form
`nw`	Normalized word: the word after Alef, Ya and digit normalization
`pos`	The part-of-speech (POS) of the word
`lem`	The lemma of the word
`na`	No-analysis: a binary feature indicating whether the morphological analyzer produced any analyses for the word
Binned	**Description**
`nw` N-grams	Normword 1/2/3-gram probabilities
`lem` N-grams	Lemma 1/2/3-gram probabilities
`pos` N-grams	POS 1/2/3-gram probabilities
`conf`	Word confidence: the ratio of the number of hypotheses in the N-best list that contain the word over the total number of hypotheses

Table 2: PZD model features. Simple features are used directly by the PZD SVM models, whereas Binned features' (numerical) values are reduced to a small, labeled category set whose labels are used as model features.

In order to obtain the morphological features, all of the training and test data is passed through MADA 3.0, a software tool for Arabic morphological analysis disambiguation (Habash and Rambow, 2005; Roth et al., 2008; Habash et al., 2010). For these experiments, MADA provides the `pos` (using MADA's native 34-tag set) and the lemma for each word. Occasionally MADA will not be able to produce any interpretations (analyses) for a word; since this is often a sign that the word is misspelled or uncommon, we define a binary `na` feature to indicate when MADA fails to generate analyses.

In addition to using the MADA features directly, we also develop a set of nine N-gram models (where N=1, 2, and 3) for the `nw`, `pos`, and `lem` features defined in Table 2. We train these models using 220M words from the Arabic Gigaword 3 corpus (Graff, 2007) which had also been run through MADA 3.0 to extract the `pos` and `lem` information. The models are built using the SRI Language Modeling Toolkit (Stolcke, 2002). Each word in a hypothesis can then be assigned a probability by each of these nine models. We reduce these probabilities into one of nine bins, with each successive bin representing an order of magnitude drop in probability (the final bin is re-

[4] Naturally, we do not use data that the BBN Byblos HR system was trained on.

[5] We conducted additional experiments where we varied the number of segments and hypotheses and found that the system benefited from added variety of segments more than hypotheses. We also modified training composition in terms of the ratio of problem/non-problem words; this did not help performance.

served for word N-grams which did not appear in the models). The bin labels are used as the SVM features.

Finally, we also use a word confidence (`conf`) feature, which is aimed at measuring the frequency with which a given word is chosen by the HR system for a given segment scan. The `conf` is defined here as the ratio of the number of hypotheses in the N-best list that the word appears in to the total number of hypotheses. These numbers are calculated using the original N-best hypothesis list, before the data is trimmed to H={1, 10}. Like the N-grams, this number is binned; in this case there are 11 bins, with 10 spread evenly over the [0,1) range, and an extra bin for values of exactly 1 (i.e., when the word appears in every hypothesis in the set).

5 Results

We describe next different experiments conducted by varying the features used in the PZD model. We present the results in terms of F-score only for simplicity; we then conduct an error analysis that examines precision and recall.

5.1 Effect of Feature Set Choice

Selecting an appropriate set of features for PZD requires extensive testing. Even when only considering the few features described in Table 2, the parameter space is quite large. Rather then exhaustively test every possible feature combination, we selectively choose feature subsets that can be compared to gain a sense of the incremental benefit provided by individual features.

5.1.1 Simple Features

Table 3 illustrates the result of taking a baseline feature set (containing `word` as the only feature) and adding a single feature from the Simple set to it. The result of combining all the Simple features is also indicated. From this, we see that Simple features, even collectively, provide only minor improvements.

5.1.2 Binned Features

Table 4 shows models which include both Simple and Binned features. First, Table 4 shows the effect of adding `nw` N-grams of successively higher orders to the `word` baseline. Here we see that even a simple unigram provides a significant benefit (compared to the improvements gained in Table 3). The largest improvement comes with the addition of the bigram (thus introducing context into the model), but the trigram provides only a slight improvement above that. This implies that pursuing higher order N-grams will result in negligible returns.

In the next part of Table 4, we see that the single feature (`pos`) which provided the highest single-feature benefit in Table 3 does not provide similar improvements under these combinations, and in fact seems detrimental. We also note that using all the features in one model is outperformed by more selective choices. Here, the best performer is the model which utilizes the `word`, `nw` N-grams,

Feature Set	**F-score**	*%Imp*
`word`	43.85	–
`word+nw`	43.86	∼0
`word+na`	44.78	2.1
`word+lem`	45.85	4.6
`word+pos`	45.91	4.7
`word+nw+pos+lem+na`	**46.34**	**5.7**

Table 3: PZD F-scores for simple feature combinations. The training set used was {S=2000, H=10} and the models were evaluated on the dev set. The improvement over the **word** baseline case is also indicated. *%Imp* is the relative improvement over the first row.

Feature Set	**F-score**	*%Imp*
`word`	43.85	–
`word+nw` 1-gram	49.51	12.9
`word+nw` 1-gram+`nw` 2-gram	59.26	35.2
`word+nw` N-grams	59.33	35.3
+`pos`	58.50	33.4
+`pos` N-grams	57.35	30.8
+`lem+lem` N-grams	59.63	36.0
+`lem+lem` N-grams+`na`	59.93	36.7
+`lem+lem` N-grams+`na+nw`	59.77	36.3
+`lem`	**60.92**	**38.9**
+`lem+na`	60.47	37.9
+`lem+lem` N-grams	60.44	37.9

Table 4: PZD F-scores for models that include Binned features. The training set used was {S=2000, H=10} and the models were evaluated on the dev set. The improvement over the **word** baseline case is also indicated. The label "N-grams" following a Binned feature refers to using 1, 2 *and* 3-grams of that feature. Indentation marks accumulative features in model. The best performing row (with bolded score) is *word+nw N-grams+lem*.

Base Feature Set	F-score		%Imp
		+conf	
word	43.85	55.83	27.3
+nw N-grams	59.33	61.71	4.0
+lem	**60.92**	62.60	2.8
+lem+na	60.47	**63.14**	4.4
+lem+lem N-grams	60.44	62.88	4.0
+pos+pos N-grams +na+nw (**all** system)	59.77	62.44	4.5

Table 5: PZD F-scores for models when word confidence is added to the feature set. The training set used was {S=2000, H=10} and the models were evaluated on the dev set. The improvement generated by including word confidence is indicated. The label "N-grams" following a Binned feature refers to using 1, 2 *and* 3-grams of that feature. Indentation marks accumulative features in model. *%Imp* is the relative improvement gained by adding the *conf* feature.

Feature Set	S = 2000 F-score	S = 4000 F-score	%Imp
word	43.85	52.08	18.8
word+conf	55.83	57.50	3.0
word+nw N-grams+lem +conf (**best** system)	62.60	**66.34**	6.0
+na	**63.14**	66.21	4.9
+lem N-grams	62.88	64.43	2.5
all	62.44	65.62	5.1

Table 6: PZD F-scores for selected models when the number of training segments (S) is doubled. The training set used was {S=2000, H=10} and {S=4000, H=10}, and the models were evaluated on the dev set. The label "N-grams" following a Binned feature refers to using 1, 2 *and* 3-grams of that feature. Indentation marks accumulative features in model.

and lem as the only features. However, the differences among this model and the other models using lem Table 4 are not statistically significant. The differences between this model and the other lower performing models are statistically significant ($p<0.05$).

5.1.3 Word Confidence

The conf feature deserves special consideration because it is the only feature which draws on information from across the entire hypothesis set. In Table 5, we show the effect of adding conf as a feature to several base feature sets taken from Table 4. Except for the baseline case, conf provides a relatively consistent benefit. The large (27.3%) improvement gained by adding conf to the **word** baseline shows that conf is a valuable feature, but the smaller improvements in the other models indicate that the information it provides largely overlaps with the information already present in those models. The differences among the last four models (all including lem) in Table 5 are not statistically significant. The differences between these four models and the first two are statistically significant ($p<0.05$).

5.2 Effect of Training Data Size

In order to allow for rapid examination of multiple feature combinations, we restricted the size of the training set (S) to maintain manageable training times. With this decision comes the implicit assumption that the results obtained will scale with additional training data. We test this assumption by taking the best-performing feature sets from Table 5 and training new models using twice the training data {S=4000}. The results are shown in Table 6. In each case, the improvements are relatively consistent (and on the order of the gains provided by the inclusion of conf as seen in Table 5), indicating that the model performance does scale with data size. However, these improvements come with a cost of a roughly 4-7x increase in training time. We note that the value of doubling S is roughly 3-6x times greater for the **word** baseline than the others; however, simply adding conf to the baseline provides an even greater improvement than doubling S. The differences between the final four models in Table 6 are not statistically significant. The differences between these models and the first two models in the table are statistically significant ($p<0.05$). For convenience, in the next section we refer to the third model listed in Table 6 as the **best** system (because it has the highest absolute F-score on the large data set), but readers should recall that these four models are roughly equivalent in performance.

5.3 Error Analysis

In this section, we look closely at the performance of a subset of systems on different types of problem words. We compare the following model settings: for {S=4000} training, we use word, word + conf, the best system from Table 6 and the model

(a)	S=4000				S=2000
	word	**wconf**	**best**	**all**	**all**
Precision	54.7	59.5	67.1	**67.4**	62.4
Recall	49.7	55.7	**65.6**	64.0	62.5
F-score	52.1	57.5	**66.3**	65.6	62.4
Accuracy	76.4	78.7	**82.8**	82.7	80.6

(b)	**%Prob**	**word**	**wconf**	**best**	**all**	**all**
Words	87.2	51.8	57.3	**68.5**	67.1	64.9
Punc.	9.3	39.5	44.7	**50.0**	46.1	40.8
Digits	3.5	24.1	44.8	34.5	34.5	**62.1**
INS	10.6	46.0	49.4	**62.1**	**62.1**	55.2
DEL	7.9	**29.2**	20.0	24.6	21.5	27.7
SUB	81.4	52.2	60.0	**70.0**	68.4	66.9
Ortho	21.6	**63.3**	51.4	52.5	53.7	48.6
Lemma	5.6	45.7	52.2	**63.0**	52.2	54.4
Semantic	54.2	48.4	64.2	**77.7**	75.9	75.5

Table 7: Error analysis results comparing the performance of multiple systems over different metrics (a) and word/error types (b). %Prob shows the distribution of problem words into different word types (word, punctuation and digit) and error types. INS, DEL and SUB stand for insertion, deletion and substitution error types, respectively. *Ortho* stands for orthographic variant. *Lemma* stands for 'shared lemma'. The columns to the right of the %Prob column show recall percentage for each word/error type.

using all possible features (**word**, **wconf**, **best** and **all**, respectively); and we also use **all** trained with {S=2000}. We consider the performance in terms of precision and recall in addition to F-score – see Table 7 (a). We also consider the percentage of recall per error type, such as word/punctuation/digit or deletion/insertion/substitution and different types of substitution errors – see Table 7 (b). The second column in this table (**%Prob**) shows the distribution of gold-tagged problem words into word and error type categories.

Overall, there is no major tradeoff between precision and recall across the different settings; although we can observe the following: (i) adding more training data helps precision more than recall (over three times more) – compare the last two columns in Table 7 (a); and (ii) the best setting has a slightly lower precision than **all** features, although a much better recall – compare columns 4 and 5 in Table 7 (a).

The performance of different settings on words is generally better than punctuation and that is better than digits. The only exceptions are in the digit category, which may be explained by that category's small count which makes it prone to large percentage fluctuations.

In terms of error type, the performance on substitutions is better than insertions, which is in turn better than deletions, for all systems compared. This makes sense since deletions are rather hard to detect and they are marked on possibly correct adjacent words, which may confuse the classifiers. One insight for future work is to develop systems for different types of errors. Considering substitutions in more detail, we see that surprisingly, the simple approach of using the word feature only (without `conf`) correctly recalls a bigger proportion of problems involving orthographic variants than other settings. It seems that the more complex the model, the harder it is to model these cases correctly. Error types that include semantic variations (different lemmas) or shared lemmas (but not explained by orthographic variation), are by contrast much harder for the simple models. The more complex models do quite well recalling errors involving semantically incoherent substitutions (around 77.7% of those cases) and words that share the same lemma but vary in inflectional features (63% of those cases). These two results are quite a jump from the basic word baseline (around 29% and 18% respectively).

The simple addition of data seems to contribute more towards the orthographic variation errors and less towards semantic errors. The different settings we use (training size and features) show some degree of complementarity in how they identify errors. We try to exploit this fact in Section 5.5 exploring some simple system combination ideas.

5.4 Blind Test Set

Table 8 shows the results of applying the same models described in Table 7 to a blind test set of yet unseen data. As mentioned in Section 4.1, the trivial baseline of the test set is comparable to the dev set. However, the test set is harder to tag than the dev set; this can be seen in the overall lower F-scores. That said, the relative order of performing features is the same as with the dev set, confirming that our **best** model is optimal for test too. On further study, we noticed that the reason for the test set difference is that the overlap in word forms between test and

	word	wconf	best	all
Precision	37.55	51.48	**57.01**	55.46
Recall	51.73	53.39	**61.97**	60.44
F-score	43.51	52.42	**59.39**	57.84
Accuracy	65.13	74.83	**77.99**	77.13

Table 8: Results on test set of 500 segments with one hypothesis each. The models were trained on the {S=4000, H=10} training set.

train is less than dev and train: 63% versus 81%, respectively on {S=4000}.

5.5 Preliminary Combination Analysis

In a preliminarily investigation of the value of complementarity across these different systems, we tried two simple model combination techniques. We restricted the search to the systems in the error analysis (Table 7).

First, we considered a sliding voting scheme where a word is marked as problematic if at least n systems agreed to that. Naturally, as n increases, precision increases and recall decreases, providing multiple tradeoff options. The range spans 49.1/83.2/61.8 (% Precision/Recall/F-score) at one end ($n = 1$) to 80.4/27.5/41.0 on the other ($n = all$). The best F-score combination was with $n = 2$ (any two agree) producing 62.8/72.4/67.3, an almost 1% higher than our best system.

In a different combination exploration, we exhaustively sought the best three systems from which any agreement (2 or 3) can produce an even better system. The best combination included the **word** model, the **best** model (both in {S=4000} training) and the **all** model (in {S=2000}). This combination yields 70.2/64.0/66.9, a lower F-score than the best general voting approach discussed above, but with a different bias towards better precision.

These basic exploratory experiments show that there is a lot of value in pursuing combinations of systems, if not for overall improvement, then at least to benefit from tradeoffs in precision and recall that may be appropriate for different applications.

5.6 Preliminary Tag Set Exploration

In all of the experiments described so far, the PZD models tag words using a binary tag set of PROB/OK. We may also consider more complex tag sets based on problem subtypes, such as SUB/INS/DEL/OK (where all the problem subtypes are differentiated), SUB/INS/OK (ignores deletions), and SUB/OK (ignores deletions and insertions). Care must be taken when comparing these systems, because the differences in tag set definition results in different baselines. Therefore we compare the % error reduction over the trivial baseline achieved in each case.

For an **all** model trained on the {S=2000, H=10} set, using the PROB/OK tag set results in a 36.3% error reduction over its trivial baseline (using the dev set). The corresponding SUB/INS/DEL/OK tag set only achieves a 34.8% error reduction. The SUB/INS/OK tag set manages a 40.1% error reduction, however. The SUB/OK tag set achieves a 38.9% error reduction. We suspect that the very low relative number of deletions (7.9% in the dev data) and the awkwardness of a DEL tag indicating a neighboring deletion (rather than the current word) may be confusing the models, and so ignoring them seems to result in a clearer picture.

6 Related Work

Common OCR/HR post-processing strategies are similar to spelling correction solutions involving dictionary lookup (Kukich, 1992; Jurafsky and Martin, 2000) and morphological restrictions (Domeij et al., 1994; Oflazer, 1996). Error detection systems using dictionary lookup can sometimes be improved by adding entries representing morphological variations of root words, particularly if the language involved has a complex morphology (Pal et al., 2000). Alternatively, morphological information can be used to construct supplemental lexicons or language models (Sari and Sellami, 2002; Magdy and Darwish, 2006).

In comparison to (Magdy and Darwish, 2006), our paper is about error detection only (done in using discriminative machine learning); whereas their work is on error correction (done in a standard generative manner (Kolak and Resnik, 2002)) with no assumptions of some cases being correct or incorrect. In essence, their method of detection is the same as our trivial baseline. The morphological features they use are *shallow* and restricted to breaking up a word into *prefix*+*stem*+*suffix*; whereas we analyze words into their lemmas, abstracting away over a large number of variations. We also made use of

part-of-speech tags, which they do not use, but suggest may help. In their work, the morphological features did not help (and even hurt a little), whereas for us, the lemma feature actually helped. Their hypothesis that their large language model (16M words) may be responsible for why the word-based models outperformed stem-based (morphological) models is challenged by the fact that our language model data (220M words) is an order of magnitude larger, but we are still able to show benefit for using morphology. We cannot directly compare to their results because of the different training/test sets and target (correction vs detection); however, we should note that their starting error rate was quite high (39% on Alef/Ya normalized words), whereas our starting error rate is almost half of that ($\sim$26% with unnormalized Alef/Yas, which account for almost 5% absolute of the errors). Perhaps a combination of the two kinds of efforts can push the perfomance on correction even further by biasing towards problematic words and avoiding incorrectly changing correct words. Magdy and Darwish (2006) do not report on percentages of words that they incorrectly modify.

7 Conclusions and Future Work

We presented a study with various settings (linguistic and non-linguistic features and learning curve) for automatically detecting problem words in Arabic handwriting recognition. Our best approach achieves a roughly $\sim$15% absolute increase in F-score over a simple baseline. A detailed error analysis shows that linguistic features, such as lemma models, help improve HR-error detection specifically where we expect them to: identifying semantically inconsistent error words.

In the future, we plan to continue improving our system by considering smarter trainable combination techniques and by separating the training for different types of errors, particularly deletions from insertions and substitutions. We would also like to conduct an extended evaluation comparing other types of morphological features, such as roots and stems, directly. One additional idea is to implement a lemma-confidence feature that examines lemma use in hypotheses across the document. This could potentially provide valuable semantic information at the document level.

We also plan to integrate our system with a system for producing correction hypotheses. We also will consider different uses for the basic system setup we developed to identify other types of text errors, such as spelling errors or code-switching between languages and dialects.

Acknowledgments

We would like to thank Premkumar Natarajan, Rohit Prasad, Matin Kamali, Shirin Saleem, Katrin Kirchhoff, and Andreas Stolcke. This work was funded under DARPA project number HR0011-08-C-0004.

References

Fadi Biadsy, Jihad El-Sana, and Nizar Habash. 2006. Online Arabic handwriting recognition using Hidden Markov Models. In *The 10th International Workshop on Frontiers in Handwriting Recognition (IWFHR'10)*, La Baule, France.

Kareem Darwish and Douglas W. Oard. 2002. Term Selection for Searching Printed Arabic. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 261–268, New York, NY, USA. ACM.

Rickard Domeij, Joachim Hollman, and Viggo Kann. 1994. Detection of spelling errors in Swedish not using a word list en clair. *J. Quantitative Linguistics*, 1:1–195.

David Graff. 2007. Arabic Gigaword 3, LDC Catalog No.: LDC2003T40. Linguistic Data Consortium, University of Pennsylvania.

Nizar Habash and Owen Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 573–580, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Nizar Habash, Owen Rambow, and Ryan Roth. 2010. MADA+TOKAN Manual. Technical Report CCLS-10-01, Center for Computational Learning Systems (CCLS), Columbia University.

Nizar Habash. 2010. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Mohamed Ben Halima and Adel M. Alimi. 2009. A multi-agent system for recognizing printed Arabic words. In Khalid Choukri and Bente Maegaard, editors, *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, Cairo, Egypt, April. The MEDAR Consortium.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing*. Prentice Hall, New Jersey, USA.

Okan Kolak and Philip Resnik. 2002. OCR error correction using a noisy channel model. In *Proceedings of the second international conference on Human Language Technology Research*.

Taku Kudo and Yuji Matsumoto. 2003. Fast Methods for Kernel-Based Text Analysis. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03)*, pages 24–31, Sapporo, Japan, July. Association for Computational Linguistics.

Karen Kukich. 1992. Techniques for Automatically Correcting Words in Text. *ACM Computing Surveys*, 24(4).

Leah S. Larkey, Lisa Ballesteros, and Margaret E. Connell, 2007. *Arabic Computational Morphology: Knowledge-based and Empirical Methods*, chapter Light Stemming for Arabic Information Retrieval. Springer Netherlands, Kluwer/Springer edition.

Zhidong Lu, Issam Bazzi, Andras Kornai, John Makhoul, Premkumar Natarajan, and Richard Schwartz. 1999. A Robust, Language-Independent OCR System. In *the 27th AIPR Workshop: Advances in Computer Assisted Recognition, SPIE*.

Walid Magdy and Kareem Darwish. 2006. Arabic OCR Error Correction Using Character Segment Correction, Language Modeling, and Shallow Morphology. In *Proceedings of 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 408–414, Sydney, Austrailia.

Volker Märgner and Haikal El Abed. 2009. Arabic Word and Text Recognition - Current Developments. In Khalid Choukri and Bente Maegaard, editors, *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, Cairo, Egypt, April. The MEDAR Consortium.

Mohsen Moftah, Waleed Fakhr, Sherif Abdou, and Mohsen Rashwan. 2009. Stem-based Arabic language models experiments. In Khalid Choukri and Bente Maegaard, editors, *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, Cairo, Egypt, April. The MEDAR Consortium.

Prem Natarajan, Shirin Saleem, Rohit Prasad, Ehry MacRostie, and Krishna Subramanian, 2008. *Arabic and Chinese Handwriting Recognition*, volume 4768 of *Lecture Notes in Computer Science*, pages 231–250. Springer, Berlin, Germany.

Kemal Oflazer. 1996. Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22:73–90.

U. Pal, P. K. Kundu, and B. B. Chaudhuri. 2000. OCR error correction of an inflectional Indian language using morphological parsing. *J. Information Sci. and Eng.*, 16:903–922.

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin. 2008. Arabic Morphological Tagging, Diacritization, and Lemmatization Using Lexeme Models and Feature Ranking. In *Proceedings of ACL-08: HLT, Short Papers*, pages 117–120, Columbus, Ohio, June. Association for Computational Linguistics.

Shirin Saleem, Huaigu Cao, Krishna Subramanian, Marin Kamali, Rohit Prasad, and Prem Natarajan. 2009. Improvements in BBN's HMM-based Offline Handwriting Recognition System. In Khalid Choukri and Bente Maegaard, editors, *10th International Conference on Document Analysis and Recognition (ICDAR)*, Barcelona, Spain, July.

Toufik Sari and Mokhtar Sellami. 2002. MOrpho-LEXical analysis for correcting OCR-generated Arabic words (MOLEX). In *The 8th International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, Niagara-on-the-Lake, Canada.

Andreas Stolcke. 2002. SRILM - an Extensible Language Modeling Toolkit. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, volume 2, pages 901–904, Denver, CO.

Stephanie Strassel. 2009. Linguistic Resources for Arabic Handwriting Recognition. In Khalid Choukri and Bente Maegaard, editors, *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, Cairo, Egypt, April. The MEDAR Consortium.

A Discriminative Model for Joint Morphological Disambiguation and Dependency Parsing

John Lee
Department of Chinese,
Translation and Linguistics
City University of Hong Kong
jsylee@cityu.edu.hk

Jason Naradowsky, David A. Smith
Department of Computer Science
University of Massachusetts, Amherst
{narad,dasmith}@cs.umass.edu

Abstract

Most previous studies of morphological disambiguation and dependency parsing have been pursued independently. Morphological taggers operate on n-grams and do not take into account syntactic relations; parsers use the "pipeline" approach, assuming that morphological information has been separately obtained.

However, in morphologically-rich languages, there is often considerable interaction between morphology and syntax, such that neither can be disambiguated without the other. In this paper, we propose a discriminative model that jointly infers morphological properties and syntactic structures. In evaluations on various highly-inflected languages, this joint model outperforms both a baseline tagger in morphological disambiguation, and a pipeline parser in head selection.

1 Introduction

To date, studies of morphological analysis and dependency parsing have been pursued more or less independently. Morphological taggers disambiguate morphological attributes such as part-of-speech (POS) or case, without taking syntax into account (Hakkani-Tür et al., 2000; Hajič et al., 2001); dependency parsers commonly assume the "pipeline" approach, relying on morphological information as part of the input (Buchholz and Marsi, 2006; Nivre et al., 2007). This approach serves many languages well, especially those with less morphological ambiguity. In English, for example, accuracy of POS tagging has risen above 97% (Toutanova et al., 2003), and that of dependency parsing has reached the low nineties (Nivre et al., 2007). For these languages, there may be little to be gained to justify the computational cost of incorporating syntactic inference during the morphological tagging task; conversely, it is doubtful that errorful morphological information is a main cause of errors in English dependency parsing.

However, the pipeline approach seems more problematic for morphologically-rich languages with substantial interactions between morphology and syntax (Tsarfaty, 2006). Consider the Latin sentence, *Una dies omnis potuit praecurrere amantis*, 'One day was able to make up for all the lovers'[1]. As shown in Table 1, the adjective *omnis* ('all') is ambiguous in number, gender, and case; there are seven valid analyses. From the perspective of a finite-state morphological tagger, the most attractive analysis is arguably the singular nominative, since *omnis* is immediately followed by the singular verb *potuit* ('could'). Indeed, the baseline tagger used in this study did make this decision. Given its nominative case, the pipeline parser assigned the verb *potuit* to be its head; the two words form the typical subject-verb relation, agreeing in number.

Unfortunately, as shown in Figure 1, the word *omnis* in fact modifies the noun *amantis*, at the end of the sentence. As a result, despite the distance between them, they must agree in number, gender and case, i.e., both must be plural masculine (or feminine) accusative. The pipeline parser, acting on the input that *omnis* is nominative, naturally did not see

[1] Taken from poem 1.13 by Sextus Propertius, English translation by Katz (2004).

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 885–894,
Portland, Oregon, June 19-24, 2011.

Latin	*Una*		*dies*		*omnis*			*potuit*	*praecurrere*	*amantis*	
English	one		day		all			could	to surpass	lovers	
Number	**sg**	pl	**sg**	pl	sg	sg	**pl**	**sg**	**-**	sg	**pl**
Gender	**f**	n	m/**f**	m/f	m/f	m/f/n	**m**/f	**-**	**-**	m/f/n	**m**/f
Case	**nom**/ab	nom/acc	**nom**	nom/acc	nom	gen	**acc**	**-**	**-**	gen	**acc**

Table 1: The Latin sentence "*Una dies omnis potuit praecurrere amantis*", meaning 'One day was able to make up for all the lovers', shown with glosses and possible morphological analyses. The correct analyses are shown in bold. The word *omnis* has 7 possible combinations of number, gender and case, while *amantis* has 5. Disambiguation partly depends on establishing *amantis* as the head of *omnis*, and so the two must agree in all three attributes.

this agreement, and therefore did not consider this syntactic relation likely.

Such a dilemma is not uncommon in languages with relatively free word order. On the one hand, it appears difficult to improve morphological tagging accuracy on words like *omnis* without syntactic knowledge; on the other hand, a parser cannot reliably disambiguate syntax unless it has accurate morphological information, in this example the agreement in number, gender, and case.

In this paper we propose to attack this chicken-and-egg problem with a discriminative model that jointly infers morphological and syntactic properties of a sentence, given its words as input. In evaluations on various highly-inflected languages, the model outperforms both a baseline tagger in morphological disambiguation, and a pipeline parser in head selection.

After a description of previous work (§2), the joint model (§3) will be contrasted with the baseline pipeline model (§4). Experimental results (§5-6) will then be presented, followed by conclusions and future directions.

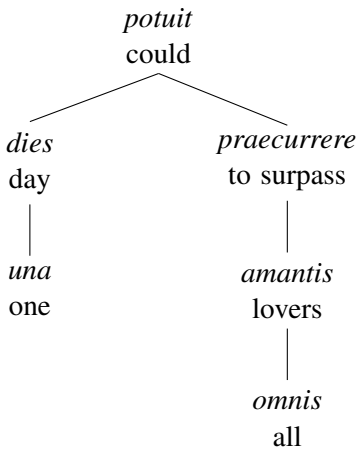


Figure 1: Dependency tree for the sentence "*Una dies omnis potuit praecurrere amantis*". The word *omnis* is an adjective modifying the noun *amantis*. This information is key to the morphological disambiguation of both words, as shown in Table 1.

2 Previous Work

Since space does not allow a full review of the vast literature on morphological analysis and parsing, we focus only on past research involving joint morphological and syntactic inference (§2.1); we then discuss Latin (§2.2), a language representative of the challenges that motivated our approach.

2.1 Joint Morphological and Syntactic Inference

Most previous work in morphological disambiguation, even when applied on morphologically complex languages with relatively free word order, such as Turkish (Hakkani-Tür et al., 2000) and Czech (Hajič et al., 2001), did not consider syntactic relationships between words. In the literature on data-driven parsing, two recent studies attempted joint inference on morphology and syntax, and both considered phrase-structure trees for Modern Hebrew (Cohen and Smith, 2007; Goldberg and Tsarfaty, 2008).

The primary focus of morphological processing in Modern Hebrew is splitting orthographic words into morphemes: clitics such as prepositions, pronouns, and the definite article must be separated from the core word. Each of the resulting morphemes is then tagged with an atomic "part-of-speech" to indicate word class and some morphological features. Similarly, the English POS tags in the Penn Treebank combine word class information with morphologi-

cal attributes such as "plural" or "past tense".

Cohen and Smith (2007) separately train a discriminative conditional random field (CRF) for segmentation and tagging, and a generative probabilistic context-free grammar (PCFG) for parsing. At decoding time, the two models are combined as a product of experts. Goldberg and Tsarfaty (2008) propose a generative joint model. This paper is the first to use a fully discriminative model for joint morphological and syntactic inference on dependency trees.

2.2 Latin

Unlike Modern Hebrew, Latin does not require extensive morpheme segmentation[2]. However, it does have a relatively free word order, and is also highly inflected, with each word having up to nine morphological attributes, listed in Table 2. In addition to its absolute numbers of cases, moods, and tenses, Latin morphology is *fusional*. For instance, the suffix *−is* in *omnis* cannot be segmented into morphemes that separately indicate gender, number, and case. According to the Latin morphological database encoded in MORPHEUS (Crane, 1991), 30% of Latin nouns can be parsed as another part-of-speech, and on average each has 3.8 possible morphological interpretations.

We know of only one previous attempt in data-driven dependency parsing for Latin (Bamman and Crane, 2008), with the goal of constructing a dynamic lexicon for a digital library. Parsing is performed using the usual pipeline approach, first with the TreeTagger analyzer (Schmid, 1994) and then with a state-of-the-art dependency parser (McDonald et al., 2005). Head selection accuracy was 61.49%, and rose to 64.99% with oracle morphological tags. Of the nine morphological attributes, gender and especially case had the lowest accuracy. This observation echoes the findings for Czech (Smith et al., 2005), where case was also the most difficult to disambiguate.

Attribute	Values
Part-of-speech (POS)	noun, verb, participle, adjective, adverb, conjunction, preposition, pronoun, numeral, interjection, exclamation, punctuation
Person	first, second, third
Number	singular, plural
Tense	present, imperfect, perfect, pluperfect, future perfect, future
Mood	indicative, subjunctive, infinitive, imperative, participle, gerund, gerundive, supine
Voice	active, passive
Gender	masculine, feminine, neuter
Case	nominative, genitive, dative, accusative, ablative, vocative, locative
Degree	comparative, superlative

Table 2: Morphological attributes and values for Latin. Ancient Greek has the same attributes; Czech and Hungarian lack some of them. In all categories except POS, a value of *null* ('-') may also be assigned. For example, a noun has '-' for the tense attribute.

3 Joint Model

This section describes a model that jointly infers morphological and syntactic properties of a sentence. It will be presented as a graphical model, starting with the variables and then the factors, which represents constraints on the variables. Let n be the number of words and m be the number of possible values for a morphological attribute. The variables are:

- WORD: the n words w_1,...,w_n of the input sentence, all observed.
- TAG: $O(nm)$ boolean variables[3] $T_{a,i,v}$, corresponding to each value of the morphological attributes listed in Table 2. $T_{a,i,v} = true$ when the word w_i has value v as its morphological attribute a. In Figure 2, CASE$_{3,acc}$ is the shorthand representing the variable $T_{case,3,acc}$. It is set to $true$ since the word w_3 has the accusative case.
- LINK: $O(n^2)$ boolean variables $L_{i,j}$ corresponding to a possible link between each pair

[2]Except for enclitics such as *-que*, *-ve*, and *-ne*, but their segmentation is rather straightforward compared to Modern Hebrew or other Semitic languages.

[3]The TAG variables were actually implemented as multinomials, but are presented here as booleans for ease of understanding.

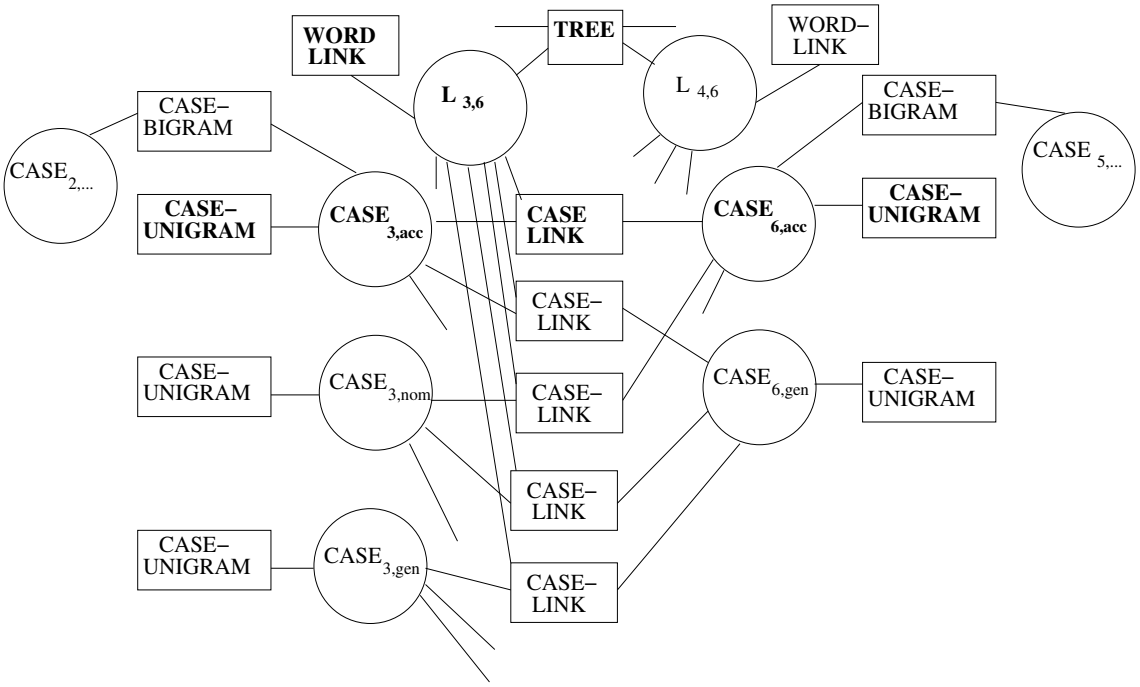


Figure 2: The joint model (§3) depicted as a graphical model. The variables, all boolean, are represented by circles and are bolded if their correct values are $true$. Factors are represented by rectangles and are bolded if they fire. For clarity, this graph shows only those variables and factors associated with one pair of words (i.e., w_3=*omnis* and w_6=*amantis*) and with one morphological attribute (i.e., case). The variables $L_{3,6}$, CASE$_{3,acc}$ and CASE$_{6,acc}$ are bolded, indicating that w_3 and w_6 are linked and both have the accusative case. The ternary factor CASE-LINK, that connects to these three variable, therefore fires.

of words[4]. $L_{i,j} = true$ when there is a dependency link from the word w_i to the word w_j. In Figure 2, the variable $L_{3,6}$ is set to $true$ since there is a dependency link between the words w_3 and w_6.

We define a probability distribution over all joint assignments $\mathcal{A}$ to the above variables,

$$p(\mathcal{A}) = \frac{1}{Z} \prod_k F_k(\mathcal{A}) \quad (1)$$

where Z is a normalizing constant. The assignment $\mathcal{A}$ is subject to a hard constraint, represented in Figure 2 as TREE, requiring that the values of the LINK variables must yield a tree, which may be non-projective. The factors $F_k(\mathcal{A})$ represent soft constraints evaluating various aspects of the "goodness" of the tree structure implied by $\mathcal{A}$. We say a factor "fires" when all its neighboring variables are $true$ and it evaluates to a non-negative real number; otherwise, it evaluates to 1 and has no effect on the product in equation (1). Soft constraints in the model are divided into **local** and **link** factors, to which we now turn.

3.1 Local Factors

The local factors consult either one word or two neighboring words, and their morphological attributes. These factors express the desirability of the assignments of morphological attributes based on local context. There are three types:

- TAG-UNIGRAM: There are $O(nm)$ such unary factors, each instance of which is connected to a TAG variable. The factor fires when $T_{a,i,v}$ is $true$. The features consist of the value v of the morphological attribute concerned, combined with the word identity of w_i, with backoff using all suffixes of the word. The CASE-UNIGRAM factors shown in Figure 2 are examples of this family of factors.

[4]Variables for link labels can be integrated in a straightforward manner, if desired.

- TAG-BIGRAM: There are $O(nm^2)$ of such binary factors, each connected to the TAG variables of a pair of neighboring words. The factor fires when T_{a,i,v_1} and $T_{a,i+1,v_2}$ are both $true$. The CASE-BIGRAM factors shown in Figure 2 are examples of this family of factors.

- TAG-CONSISTENCY: For each word, the TAG variables representing the possible POS values are connected to those representing the values of other morphological attributes, yielding $O(nm^2)$ binary factors. They fire when T_{pos,i,v_1} and T_{a,i,v_2} are both $true$. These factors are intended to discourage inconsistent assignments, such as a non-null tense for a noun.

It is clear that so far, none of these factors are aware of the morphological agreement between *omnis* and *amantis*, crucial for inferring their syntactic relation. We now turn our attention to link factors, which serve this purpose.

3.2 Link Factors

The link factors consult all pairs of words, possibly separated by a long distance, that may have a dependency link. These factors model the likelihood of such a link based on the word identities and their morphological attributes:

- WORD-LINK: There are $O(n^2)$ such unary factors, each connected to a LINK variable, as shown in Figure 2. The factor fires when $L_{i,j}$ is $true$. Features include various combinations of the word identities of the parent w_i and child w_j, and 5-letter prefixes of these words, replicating the so-called "basic features" used by McDonald et al. (2005).

- POS-LINK: There are $O(n^2m^2)$ such ternary factors, each connected to the variables $L_{i,j}$, T_{i,pos,v_i} and T_{j,pos,v_j}. It fires when all three are $true$ or, in other words, when the parent word w_i has POS v_i, and the child w_j has POS v_j. Features replicate all the so-called "basic features" used by McDonald et al. (2005) that involve POS. These factors are not shown in Figure 2, but would have exactly the same structure as the CASE-LINK factors.

  Beyond these basic features, McDonald et al. (2005) also utilize POS trigrams and POS 4-grams. Both include the POS of two linked words, w_i and w_j. The third component in the trigrams is the POS of each word w_k located between w_i and w_j, $i < k < j$. The two additional components that make up the 4-grams are subsets of the POS of words located to the immediate left and right of w_i and w_j.

  If fully implemented in our joint model, these features would necessitate two separate families of link factors: $O(n^3m^3)$ factors for the POS trigrams, and $O(n^2m^4)$ factors for the POS 4-grams. To avoid this substantial increase in model complexity, these features are instead approximated: the POS of all words involved in the trigrams and 4-grams, except those of w_i and w_j, are regarded as fixed, their values being taken from the output of a morphological tagger (§4.1), rather than connected to the appropriate TAG variables. This approximation allows these features to be incorporated in the POS-LINK factors.

- MORPH-LINK: There are $O(n^2m^2)$ such ternary factors, each connected to the variables $L_{i,j}$, T_{i,a,v_i} and T_{j,a,v_j}, for every attribute a other than POS. The factor fires when all three variables are $true$, and both v_i and v_j are non-null; i.e., it fires when the parent word w_i has v_i as its morphological attribute a, and the child w_j has v_j. Features include the combination of v_i and v_j themselves, and agreement between them. The CASE-LINK factors in Figure 2 are an example of this family of factors.

4 Baselines

To ensure a meaningful comparison with the joint model, our two baselines are both implemented in the same graphical model framework, and trained with the same machine-learning algorithm. Roughly speaking, they divide up the variables and factors of the joint model and train them separately. For morphological disambiguation, we use the baseline tagger described in §4.1. For dependency parsing, our baseline is a "pipeline" parser (§4.2) that infers syntax upon the output of the baseline tagger.

4.1 Baseline Morphological Tagger

The tagger is a graphical model with the WORD and TAG variables, connected by the local factors TAG-UNIGRAM, TAG-BIGRAM, and TAG-CONSISTENCY, all used in the joint model (§3).

4.2 Baseline Dependency Parser

The parser has no local factors, but has the same variables as the joint model and the same features from all three families of link factors (§3). However, since it takes as input the morphological attributes predicted by the tagger, the TAG variables are now observed. This leads to a change in the structure of the link factors — all features from the POS-LINK factors now belong to the WORD-LINK factors, since the POS of all words are observed. In short, the features of the parser are a replication of (McDonald et al., 2005), but also extended beyond POS to the other morphological attributes, with the features in the MORPH-LINK factors incorporated into WORD-LINK for similar reasons.

5 Experimental Set-up

5.1 Data

Our evaluation focused on the Latin Dependency Treebank (Bamman and Crane, 2006), created at the Perseus Digital Library by tailoring the Prague Dependency Treebank guidelines for the Latin language. It consists of excerpts from works by eight Latin authors. We randomly divided the 53K-word treebank into 10 folds of roughly equal sizes, with an average of 5314 words (347 sentences) per fold. We used one fold as the development set and performed cross-validation on the other nine.

To measure how well our model generalizes to other highly-inflected, relatively free-word-order languages, we considered Ancient Greek, Hungarian, and Czech. Their respective datasets consist of 8000 sentences from the Ancient Greek Dependency Treebank (Bamman et al., 2009), 5800 from the Hungarian Szeged Dependency Treebank (Vincze et al., 2010), and a subset of 3100 from the Prague Dependency Treebank (Böhmová et al., 2003).

5.2 Training

We define each factor in (1) as a log-linear function:

$$F_k(\mathcal{A}) = \exp \sum_h \theta_h f_h(\mathcal{A}, W, k) \qquad (2)$$

Given an assignment $\mathcal{A}$ and words W, f_h is an indicator function describing the presence or absence of the feature, and θ_h is the corresponding set of weights learned using stochastic gradient ascent, with the gradients inferred by loopy belief propagation (Smith and Eisner, 2008). The variance of the Gaussian prior is set to 1. The other two parameters in the training process, the number of belief propagation iterations and the number of training rounds, were tuned on the development set.

5.3 Decoding

The output of the joint model is the assignment to the TAG and LINK variables. Loopy belief propagation (BP) was used to calculate the posterior probabilities of these variables. For TAG, we emit the tag with the highest posterior probability as computed by sum-product BP. We produced head attachments by first calculating the posteriors of the LINK variables with BP and then passing them to an edge-factored tree decoder. This is equivalent to minimum Bayes risk decoding (Goodman, 1996), which is used by Cohen and Smith (2007) and Smith and Eisner (2008). This MBR decoding procedure enforces the hard constraint that the output be a tree but sums over possible morphological assignments.[5]

5.4 Reducing Model Complexity

In principle, the joint model should consider every possible combination of morphological attributes for every word. In practice, to reduce the complexity of the model, we used a pre-existing morphological database, MORPHEUS (Crane, 1991), to constrain the range of possible values of the attributes listed in Table 2; more precisely, we add a hard constraint, requiring that assignments to the TAG variables be compatible with MORPHEUS. This constraint significantly reduces the value of m in the big-O notation

[5] This approach to nuisance variables has also been used effectively for parsing with tree-substitution grammars, where several derived trees may correspond to each derivation tree, and parsing with PCFGs with latent annotations.

Model Attr. ↓	Tagger all	Joint all	Tagger non-null	Joint non-null
POS	94.4	94.5	94.4	94.5
Person	99.4	99.5	97.1	97.6
Number	95.3	95.9	93.7	94.5
Tense	98.0	98.2	93.2	93.9
Mood	98.1	98.3	93.8	94.4
Voice	98.5	98.6	95.3	95.7
Gender	93.1	93.9	87.7	89.1
Case	89.3	90.0	79.9	81.2
Degree	99.9	99.9	86.4	90.8
UAS	61.0	61.9	—	—

Table 3: **Latin** morphological disambiguation and parsing. For some attributes, such as degree, a substantial portion of words have the *null* value. The non-null columns provides a sharper picture by excluding these "easy" cases. Note that POS is never *null*.

for the number of variables and factors described in §3. To illustrate the effect, the graphical model of the sentence in Table 1, whose six words are all covered by the database, has 1,866 factors; without the benefit of the database, the full model would have 31,901 factors.

The MORPHEUS database was automatically generated from a list of stems, inflections, irregular forms and morphological rules. It covers about 99% of the distinct words in the Latin Dependency Treebank. At decoding time, for each fold, the database is further augmented with tags seen in training data. After this augmentation, an average of 44 words are "unseen" in each fold.

Similarly, we constructed morphological dictionaries for Czech, Ancient Greek, and Hungarian from words that occurred at least five times in the training data; words that occurred fewer times were unrestricted in the morphological attributes they could take on.

6 Experimental Results

We compare the performance of the pipeline model (§4) and the joint model (§3) on morphological disambiguation and unlabeled dependency parsing.

Model Attr. ↓	Tagger all	Joint all	Tagger non-null	Joint non-null
POS	95.5	95.7	95.5	95.7
Person	98.4	98.8	93.5	95.6
Number	91.2	92.3	87.0	88.4
Tense	98.4	98.8	92.7	96.1
Voice	98.5	98.7	93.2	95.8
Gender	86.6	87.9	75.6	78.0
Case	84.1	85.6	74.3	76.5
Degree	97.9	98.0	90.1	90.1
UAS	67.4	68.7	—	—

Table 4: **Czech** morphological disambiguation and parsing. As with Latin, the model is least accurate with noun/adjective categories of gender number, and case, particularly when considering only words whose true value is non-null for those attributes. Joint inference with syntactic features improves accuracy across the board.

Model Attr. ↓	Tagger all	Joint all	Tagger non-null	Joint non-null
POS	94.9	95.7	94.9	95.7
Person	98.7	99.0	92.2	94.6
Number	97.4	97.9	96.5	97.1
Tense	96.8	97.2	84.1	86.8
Mood	97.9	98.3	91.4	93.2
Voice	97.8	98.0	91.3	92.4
Gender	95.4	96.1	90.7	91.9
Case	95.9	96.3	92.0	92.6
Degree	99.8	99.9	33.3	55.6
UAS	68.0	70.5	—	—

Table 5: **Ancient Greek** morphological disambiguation and parsing. Noun/adjective morphology is more accurate, but verbal morphology is more problematic.

Model Attr. ↓	Tagger all	Joint all	Tagger non-null	Joint non-null
POS	95.8	95.8	95.8	95.8
Person	98.5	98.6	94.9	94.1
Number	97.4	97.5	96.8	96.6
Tense	98.9	99.3	97.2	97.3
Mood	98.7	99.2	95.8	97.3
Case	96.7	97.0	94.5	94.9
Degree	97.9	98.1	87.5	88.6
UAS	78.2	78.8	—	—

Table 6: **Hungarian** morphological disambiguation and parsing. The agglutinative morphological system makes local cues more effective, but syntactic information helps in almost all categories.

6.1 Morphological Disambiguation

As seen in Table 3, the joint model outperforms[6] the baseline tagger in all attributes in Latin morphological disambiguation. Among words not covered by the morphological database, accuracy in POS is slightly better, but lower for case, gender and number.

The joint model made the most gains on adjectives and participles. Both parts-of-speech are particularly ambiguous: according to MORPHEUS, 43% of the adjectives can be interpreted as another POS, most frequently nouns; while participles have an average of 5.5 morphological interpretations. Both also often have identical forms for different genders, numbers and cases. In these situations, syntactic considerations help nudge the joint model to the correct interpretations.

Experiments on the other three languages bear out similar results: the joint model improves morphological disambiguation. The performance of Czech (Table 4) exhibits the closest analogue to Latin: gender, number, and case are much less accurately predicted than are the other morphological attributes. Like Latin, Czech lacks definite and indefinite articles to provide high-confidence cues for noun phrase boundaries.

The Ancient Greek treebank comprises both archaic texts, before the development of a definite article, and later classic Greek, which has a definite article; Hungarian has both a definite and an indefinite article. In both languages (Tables 5 and 6), noun and adjective gender, number, and case are more accurately predicted than in Czech and Latin. The verbal system of ancient Greek, in contrast, is more complex than that of the other languages, so mood, voice, and tense accuracy are lower.

6.2 Dependency Parsing

In addition to morphological disambiguation, we also measured the performance of the joint model on dependency parsing of Latin and the other languages. The baseline pipeline parser (§4.2) yielded 61.00% head selection accuracy (i.e., unlabeled attachment score, UAS), outperformed[7] by the joint model at 61.88%. The joint model showed similar improvements in Ancient Greek, Hungarian, and Czech.

Wrong decisions made by the baseline tagger often misled the pipeline parser. For adjectives, the example shown in Table 1 and Figure 1 is a typical scenario, where an accusative adjective was tagged as nominative, and was then misanalyzed by the parser as modifying a verb (as a subject) rather than modifying an accusative noun. For participles modifying a noun, the wrong noun was often chosen based on inaccurate morphological information. In these cases, the joint model, entertaining all morphological possibilities, was able to find the combination of links and morphological analyses that are collectively more likely.

The accuracy figures of our baselines are comparable, but not identical, to their counterparts reported in (Bamman and Crane, 2008). The differences may partially be attributed to the different morphological tagger used, and the different learning algorithm, namely Margin Infused Relaxed Algorithm (MIRA) in (McDonald et al., 2005) rather than maximum likelihood. More importantly, the Latin Dependency Treebank has grown from about 30K at the time of the previous work to 53K at present, resulting in significantly different training and testing material.

Gold Pipeline Parser When given perfect morphological information, the Latin parser performs at 65.28% accuracy in head selection. Despite the oracle morphology, the head selection accuracy is still below other languages. This is hardly surprising, given the relatively small training set, and that the "the most difficult languages are those that combine a relatively free word order with a high degree of inflection", as observed at the recent dependency parsing shared task (Nivre et al., 2007); both of these are characteristics of Latin.

A particularly troublesome structure is coordination; the most frequent link errors all involve either a parent or a child as a conjunction. In a list of words, all words and coordinators depend on the final coordinator. Since the factors in our model consult only one link at a time, they do not sufficiently capture this kind of structures. Higher-order features, particularly those concerned with links with grandparents and siblings, have been shown to benefit dependency

[6] The differences are statistically significant in all ($p < 0.01$ by McNemar's Test) but POS ($p = 0.5$).

[7] Significant at $p < e^{-11}$ by McNemar's Test.

parsing (Smith and Eisner, 2008) and may be able to address this issue.

7 Conclusions and Future Work

We have proposed a discriminative model that jointly infers morphological properties and syntactic structures. In evaluations on various highly-inflected languages, this joint model outperforms both a baseline tagger in morphological disambiguation, and a pipeline parser in head selection.

This model may be refined by incorporating richer features and improved decoding. In particular, we would like to experiment with higher-order features (§6), and with maximum *a posteriori* decoding, via max-product BP or (relaxed) integer linear programming. Further evaluation on other morphological systems would also be desirable.

Acknowledgments

We thank David Bamman and Gregory Crane for their feedback and support. Part of this research was performed by the first author while visiting Perseus Digital Library at Tufts University, under the grants *A Reading Environment for Arabic and Islamic Culture*, Department of Education (P017A060068-08) and *The Dynamic Lexicon: Cyberinfrastructure and the Automatic Analysis of Historical Languages*, National Endowment for the Humanities (PR-50013-08). The latter two authors were supported by Army prime contract #W911NF-07-1-0216 and University of Pennsylvania subaward #103-548106; by SRI International subcontract #27-001338 and ARFL prime contract #FA8750-09-C-0181; and by the Center for Intelligent Information Retrieval. Any opinions, findings, and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsors.

References

David Bamman and Gregory Crane. 2006. The Design and Use of a Latin Dependency Treebank. *Proc. Workshop on Treebanks and Linguistic Theories (TLT)*. Prague, Czech Republic.

David Bamman and Gregory Crane. 2008. Building a Dynamic Lexicon from a Digital Library. *Proc. 8th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL 2008)*. Pittsburgh, PA.

David Bamman, Francesco Mambrini, and Gregory Crane. 2009. An Ownership Model of Annotation: The Ancient Greek Dependency Treebank. *Proc. Workshop on Treebanks and Linguistic Theories (TLT)*.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level Annotation Scenario. In *Treebanks: Building and Using Parsed Corpora*, A. Abeillé (ed). Kluwer.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. *Proc. CoNLL*. New York, NY.

Shay B. Cohen and Noah A. Smith. 2007. Joint Morphological and Syntactic Disambiguation. *Proc. EMNLP-CoNLL*. Prague, Czech Republic.

Gregory Crane. 1991. Generating and Parsing Classical Greek. *Literary and Linguistic Computing* 6(4):243–245.

Yoav Goldberg and Reut Tsarfaty. 2008. A Single Generative Model for Joint Morphological Segmentation and Syntactic Parsing. *Proc. ACL*. Columbus, OH.

Joshua Goodman. 1996. Parsing Algorithms and Metrics. *Proc. ACL*.

J. Hajič, P. Krbec, P. Květoň, K. Oliva, and V. Petkevič. 2001. Serial Combination of Rules and Statistics: A Case Study in Czech Tagging. *Proc. ACL*.

D. Z. Hakkani-Tür, K. Oflazer, and G. Tür. 2000. Statistical Morphological Disambiguation for Agglutinative Languages. *Proc. COLING*.

Vincent Katz. 2004. *The Complete Elegies of Sextus Propertius*. Princeton University Press, Princeton, NJ.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jana Hajič. 2005. Non-projective Dependency Parsing using Spanning Tree Algorithms. *Proc. HLT/EMNLP*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online Large-Margin Training of Dependency Parsers. *Proc. ACL*.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 Shared Task on Dependency Parsing. *Proc. CoNLL Shared Task Session of EMNLP-CoNLL*. Prague, Czech Republic.

Helmut Schmid. 1994. Probabilistic Part-of-Speech Tagging using Decision Trees. *Proc. International Conference on New Methods in Language Processing*. Manchester, UK.

Noah A. Smith, David A. Smith and Roy W. Tromble. 2005. Context-Based Morphological Disambiguation with Random Fields. *Proc. HLT/EMNLP*. Vancouver, Canada.

David Smith and Jason Eisner. 2008. Dependency Parsing by Belief Propagation. *Proc. EMNLP*. Honolulu, Hawaii.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. *Proc. HLT-NAACL*. Edmonton, Canada.

Reut Tsarfaty. 2006. Integrated Morphological and Syntactic Disambiguation for Modern Hebrew. *Proc. COLING-ACL Student Research Workshop*.

Veronika Vincze, Dóra Szauter, Attila Almási, György Móra, Zoltán Alexin, and János Csirik. 2010. Hungarian Dependency Treebank. *Proc. LREC*.

Unsupervised Bilingual Morpheme Segmentation and Alignment with Context-rich Hidden Semi-Markov Models

Jason Naradowsky*
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
narad@cs.umass.edu

Kristina Toutanova
Microsoft Research
Redmond, WA 98502
kristout@microsoft.com

Abstract

This paper describes an unsupervised dynamic graphical model for morphological segmentation and bilingual morpheme alignment for statistical machine translation. The model extends Hidden Semi-Markov chain models by using factored output nodes and special structures for its conditional probability distributions. It relies on morpho-syntactic and lexical source-side information (part-of-speech, morphological segmentation) while learning a morpheme segmentation over the target language. Our model outperforms a competitive word alignment system in alignment quality. Used in a monolingual morphological segmentation setting it substantially improves accuracy over previous state-of-the-art models on three Arabic and Hebrew datasets.

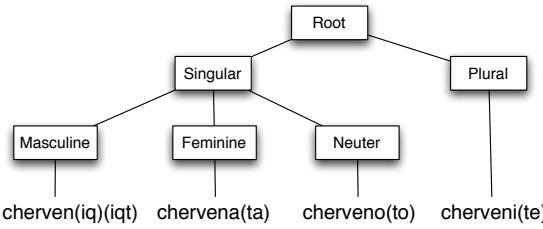
1 Introduction

An enduring problem in statistical machine translation is sparsity. The word alignment models of modern MT systems attempt to capture $p(e_i|f_j)$, the probability that token e_i is a translation of f_j. Underlying these models is the assumption that the word-based tokenization of each sentence is, if not optimal, at least appropriate for specifying a conceptual mapping between the two languages.

However, when translating between unrelated languages – a common task – disparate morphological systems can place an asymmetric conceptual burden on words, making the lexicon of one language much more coarse. This intensifies the problem of sparsity as the large number of word forms created through morphologically productive processes hinders attempts to find concise mappings between concepts.

For instance, Bulgarian adjectives may contain markings for gender, number, and definiteness. The following tree illustrates nine realized forms of the Bulgarian word for *red*, with each leaf listing the definite and indefinite markings.

Table 1: Bulgarian forms of *red*



Contrast this with English, in which this information is marked either on the modified word or by separate function words.

In comparison to a language which isn't morphologically productive on adjectives, the alignment model must observe nine times as much data (assuming uniform distribution of the inflected forms) to yield a comparable statistic. In an area of research where the amount of data available plays a large role in a system's overall performance, this sparsity can be extremely problematic. Further complications are created when lexical sparsity is compounded with the desire to build up alignments over increasingly larger contiguous phrases.

To address this issue we propose an alternative to word alignment: *morpheme alignment*, an alignment that operates over the smallest meaningful subsequences of words. By striving to keep a direct 1-to-1 mapping between corresponding semantic units across languages, we hope to find better estimates

*This research was conducted during the author's internship at Microsoft Research

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 895–904,
Portland, Oregon, June 19-24, 2011.

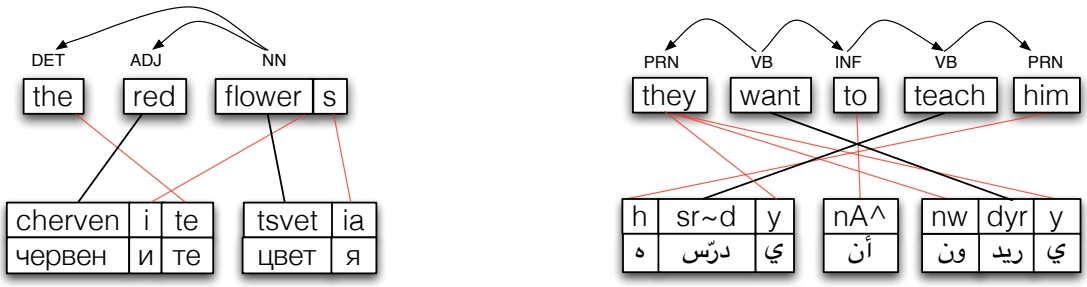


Figure 1: A depiction of morpheme-level alignment. Here dark lines indicate the more stem-focused alignment strategy of a traditional word or phrasal alignment model, while thin lines indicate a more fine-grained alignment across morphemes. In the alignment between English and Bulgarian (a) the morpheme-specific alignment reduces sparsity in the adjective and noun (*red flowers*) by isolating the stems from their inflected forms. Despite Arabic exhibiting templatic morphology, there are still phenomena which can be accounted for with a simpler segmentational approach. The Arabic alignment (b) demonstrates how the plural marker on English *they* would normally create sparsity by being marked in three additional places, two of them inflections in larger wordforms.

for the alignment statistics. Our results show that this improves alignment quality.

In the following sections we describe an unsupervised dynamic graphical model approach to monolingual morphological segmentation and bilingual morpheme alignment using a linguistically motivated statistical model. In a bilingual setting, the model relies on morpho-syntactic and lexical source-side information (part-of-speech, morphological segmentation, dependency analysis) while learning a morpheme segmentation over the target language. In a monolingual setting we introduce effective use of context by feature-rich modeling of the probabilities of morphemes, morpheme-transitions, and word boundaries. These additional sources of information provide powerful bias for unsupervised learning, without increasing the asymptotic running time of the inference algorithm.

Used as a monolingual model, our system significantly improves the state-of-the-art segmentation performance on three Arabic and Hebrew datasets. Used as a bilingual model, our system outperforms the state-of-the-art WDHMM (He, 2007) word alignment model as measured by alignment error rate (AER).

In agreement with some previous work on tokenization/morpheme segmentation for alignment (Chung and Gildea, 2009; Habash and Sadat, 2006), we find that the best segmentation for alignment does not coincide with the gold-standard segmentation and our bilingual model does not outperform our monolingual model in segmentation F-Measure.

2 Model

Our model defines the probability of a target language sequence of words (each consisting of a sequence of morphemes), and alignment from target to source morphemes, given a source language sequence of words (each consisting of a sequence of morphemes).

An example morpheme segmentation and alignment of phrases in English-Arabic and English-Bulgarian is shown in Figure 1. In our task setting, the words of the source and target language as well as the morpheme segmentation of the source (English) language are given. The morpheme segmentation of the target language and the alignments between source and target morphemes are hidden.

The source-side input, which we assume to be English, is processed with a gold morphological segmentation, part-of-speech, and dependency tree analysis. While these tools are unavailable in resource-poor languages, they are often available for at least one of the modeled languages in common translation tasks. This additional information then provides a source of features and conditioning information for the translation model.

Our model is derived from the hidden-markov model for word alignment (Vogel et al., 1996; Och and Ney, 2000). Based on it, we define a dynamic

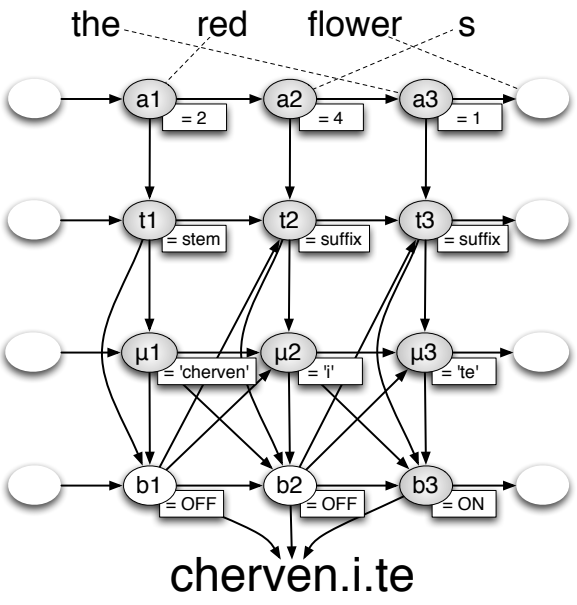


Figure 2: A graphical depiction of the model generating the transliteration of the first Bulgarian word from Figure 1. Trigram dependencies and some incoming/outgoing arcs have been omitted for clarity.

graphical model which lets us encode more linguistic intuition about morpheme segmentation and alignment: (*i*) we extend it to a hidden semi-markov model to account for hidden target morpheme segmentation; (*ii*) we introduce an additional observation layer to model observed word boundaries and thus truly represent target sentences as words composed of morphemes, instead of just a sequence of tokens; (*iii*) we employ hierarchically smoothed models and log-linear models to capture broader context and to better represent the morpho-syntactic mapping between source and target languages. (*iv*) we enrich the hidden state space of the model to encode morpheme types {prefix,suffix,stem}, in addition to morpheme alignment and segmentation information.

Before defining our model formally, we introduce some notation. Each possible morphological segmentation and alignment for a given sentence pair can be described by the following random variables:

Let $\mu_1\mu_2 \ldots \mu_I$ denote I morphemes in the segmentation of the target sentence. For the Example in Figure 1 (a) I=5 and μ_1=*cherven*, μ_2=*i* $\ldots$, and μ_5=*ia*. Let $b_1, b_2, \ldots b_I$ denote Bernoulli variables indicating whether there is a word boundary after morpheme μ_i. For our example, $b_3 = 1$, $b_5 = 1$, and the other b_i are 0. Let $c_1, c_2, \ldots, c_T$ denote the non-space characters in the target string, and $wb_1, \ldots, wb_T$ denote Bernoulli variables indicating whether there is a word boundary after the corresponding target character. For our example, $T = 14$ (for the Cyrillic version) and the only wb variables that are on are wb_9 and wb_{14}. The c and wb variables are observed. Let $s_1 s_2 \ldots s_T$ denote Bernoulli segmentation variables indicating whether there is a morpheme boundary after the corresponding character. The values of the hidden segmentation variables s together with the values of the observed c and wb variables uniquely define the values of the morpheme variables μ_i and the word boundary variables b_i. Naturally we enforce the constraint that a given word boundary $wb_t = 1$ entails a segmentation boundary $s_t = 1$. If we use bold letters to indicate a vector of corresponding variables, we have that $\mathbf{c}, \mathbf{wb}, \mathbf{s}{=}\boldsymbol{\mu}, \mathbf{b}$. We will define the assumed parametric form of the learned distribution using the $\boldsymbol{\mu}, \mathbf{b}$ but the inference algorithms are implemented in terms of the $\mathbf{s}$ and $\mathbf{wb}$ variables.

We denote the observed source language morphemes by $e_1 \ldots e_J$. Our model makes use of additional information from the source which we will mention when necessary.

The last part of the hidden model state represents the alignment between target and source morphemes and the type of target morphemes. Let $ta_i = [a_i, t_i]$, $i = 1 \ldots I$ indicate a factored state where a_i represents one of the J source words (or NULL) and t_i represents one of the three morpheme types {prefix,suffix,stem}. a_i is the source morpheme aligned to μ_i and t_i is the type of μ_i.

We are finally ready to define the desired probability of target morphemes, morpheme types, alignments, and word boundaries given source:

$$P(\boldsymbol{\mu}, \mathbf{ta}, \mathbf{b}|\mathbf{e}) = \prod_{i=1}^{I} P_T(\mu_i|ta_i, b_{i-1}, b_{i-2}, \mu_{i-1}, \mathbf{e}) \cdot P_B(b_i|\mu_i, \mu_{i-1}, ta_i, b_{i-1}, b_{i-2}, \mathbf{e}) \cdot P_D(ta_i|ta_{i-1}, b_{i-1}, \mathbf{e}) \cdot LP(|\mu_i|)$$

We now describe each of the factors used by our model in more detail. The formulation makes explicit the full extent of dependencies we have explored in this work. By simplifying the factors

we can recover several previously used models for monolingual segmentation and bilingual joint segmentation and alignment. We discuss the relationship of this model to prior work and study the impact of the novel components in our experiments.

When the source sentence is assumed to be empty (and thus contains no morphemes to align to) our model turns into a monolingual morpheme segmentation model, which we show exceeds the performance of previous state-of-the-art models. When we remove the word boundary component, reduce the order of the alignment transition, omit the morphological type component of the state space, and retain only minimal dependencies in the morpheme translation model, we recover the joint tokenization and alignment model based on IBM Model-1 proposed by (Chung and Gildea, 2009).

2.1 Morpheme Translation Model

In the model equation, P_T denotes the morpheme translation probability. The standard dependence on the aligned source morpheme is represented as a dependence on the state ta_i and the whole annotated source sentence $\mathbf{e}$. We experimented with multiple options for the amount of conditioning context to be included. When most context is used, there is a bigram dependency of target language morphemes as well as dependence on two previous boundary variables and dependence on the aligned source morpheme e_{a_i} as well as its POS tag.

When multiple conditioning variables are used we assume a special linearly interpolated backoff form of the model, similar to models routinely used in language modeling.

As an example, suppose we estimate the morpheme translation probability as $P_T(\mu_i|e_{a_i}, t_i)$. We estimate this in the M-step, given expected joint counts $c(\mu_i, e_{a_i}, t_i)$ and marginal counts derived from these as follows:

$P_T(\mu_i|e_{a_i}, t_i) = \frac{c(\mu_i, e_{a_i}, t_i) + \alpha_2 P_2(\mu_i|t_i)}{c(e_{a_i}, t_i) + \alpha_2}$

The lower order distributions are estimated recursively in a similar way:

$P_2(\mu_i|t_i) = \frac{c(\mu_i, t_i) + \alpha_1 P_1(\mu_i)}{c(t_i) + \alpha_1}$

$P_1(\mu_i) = \frac{c(\mu_i) + \alpha_0 P_0(\mu_i)}{c(.) + \alpha_0}$

For P_0 we used a unigram character language model. This hierarchical smoothing can be seen as an approximation to hierarchical Dirichlet priors with maximum aposteriori estimation.

Note how our explicit treatment of word boundary variables b_i allows us to use a higher order dependence on these variables. If word boundaries are treated as morphemes on their own, we would need to have a four-gram model on target morphemes to represent this dependency which we are now representing using only a bigram model on hidden morphemes.

2.2 Word Boundary Generation Model

The P_B distribution denotes the probability of generating word boundaries. As a sequence model of sentences the basic hidden semi-markov model completely ignores word boundaries. However, they can be powerful predictors of morpheme segments (by for example, indicating that common prefixes follow word boundaries, or that common suffixes precede them). The log-linear model of (Poon et al., 2009) uses word boundaries as observed left and right context features, and Morfessor (Creutz and Lagus, 2007) includes boundaries as special boundary symbols which can inform about the morpheme state of a morpheme (but not its identity).

Our model includes a special generative process for boundaries which is conditioned not only on the previous morpheme state but also the previous two morphemes and other boundaries. Due to the fact that boundaries are observed their inclusion in the model does not increase the complexity of inference.

The inclusion of this distribution lets us estimate the likelihood of a word consisting of one,two,three, or more morphemes. It also allows the estimation of likelihood that particular morphemes are in the beginning/middle/end of words. Through the included factored state variable ta_i word boundaries can also inform about the likelihood of a morpheme aligned to a source word of a particular pos tag to end a word. We discuss the particular conditioning context for this distribution we found most helpful in our experiments.

Similarly to the P_T distribution, we make use of multiple context vectors by hierarchical smoothing of distributions of different granularities.

2.3 Distortion Model

P_D indicates the distortion modeling distribution we use. [1] Traditional distortion models represent $P(a_j|a_{j-1}, \mathbf{e})$, the probability of an alignment given the previous alignment, to bias the model away from placing large distances between the aligned tokens of consecutively sequenced tokens. In addition to modeling a larger state space to also predict morpheme types, we extend this model by using a special log-linear model form which allows the integration of rich morpho-syntactic context. Log-linear models have been previously used in unsupervised learning for local multinomial distributions like this one in e.g. (Berg-Kirkpatrick et al., 2010), and for global distributions in (Poon et al., 2009).

The special log-linear form allows the inclusion of features targeted at learning the transitions among morpheme types and the transitions between corresponding source morphemes. The set of features with example values for this model is depicted in Table 3. The example is focussed on the features firing for the transition from the Bulgarian suffix *te* aligned to the first English morpheme $\mu_{i-1} = te$, t_{i-1}=suffix, a_{i-1}=1, to the Bulgarian root *tsvet* aligned to the third English morpheme $\mu_i = tsvet$, t_i=root, a_i=3. The first feature is the absolute difference between a_i and $a_{i-1} + 1$ and is similar to information used in other HMM word alignment models (Och and Ney, 2000) as well as phrase-translation models (Koehn, 2004). The alignment positions a_i are defined as indices of the aligned source morphemes. We additionally compute distortion in terms of distance in number of source words that are skipped. This distance corresponds to the feature name WORD DISTANCE. Looking at both kinds of distance is useful to capture the intuition that consecutive morphemes in the same target word should prefer to have a higher proximity of their aligned source words, as compared to consecutive morphemes which are not part of the same target word. The binned distances look at the sign of the distortion and bin the jumps into 5 bins, pooling the distances greater than 2 together. The feature SAME TARGET WORD indicates whether the two consecutive morphemes are part of the same word. In this case, they are not. This feature is not useful on its own because it does not distinguish between different alignment possibilities for ta_i, but is useful in conjunction with other features to differentiate the transition behaviors within and across target words. The DEP RELATION feature indicates the direct dependency relation between the source words containing the aligned source morphemes, if such relationship exists. We also represent alignments to null and have one null for each source word, similarly to (Och and Ney, 2000) and have a feature to indicate null. Additionally, we make use of several feature conjunctions involving the null, same target word, and distance features.

Feature	Value
MORPH DISTANCE	*1*
WORD DISTANCE	*1*
BINNED MORPH DISTANCE	*fore1*
BINNED WORD DISTANCE	*fore1*
MORPH STATE TRANSITION	suffix-root
SAME TARGET WORD	*False*
POS TAG TRANSITION	*DET-NN*
DEP RELATION	*DET←NN*
NULL ALIGNMENT	*False*
conjunctions	...

Figure 3: Features in log-linear distortion model firing for the transition from *te*:suffix:1 to *tsvet*:root:3 in the example sentence pair in Figure 1a.

2.4 Length Penalty

Following (Chung and Gildea, 2009) and (Liang and Klein, 2009) we use an exponential length penalty on morpheme lengths to bias the model away from the maximum likelihood under-segmentation solution. The form of the penalty is:

$LP(|\mu_i|) = \frac{1}{e^{|\mu_i|^{lp}}}$

Here lp is a hyper-parameter indicating the power that the morpheme length is raised to. We fit this parameter using an annotated development set, to optimize morpheme-segmentation F1. The model is extremely sensitive to this value and performs quite poorly if such penalty is not used.

2.5 Inference

We perform inference by EM training on the aligned sentence pairs. In the E-step we compute expected

[1] To reduce complexity of exposition we have omitted the final transition to a special state beyond the source sentence end after the last target morpheme.

counts of all hidden variable configurations that are relevant for our model. In the M-step we re-estimate the model parameters (using LBFGS in the M-step for the distortion model and using count interpolation for the translation and word-boundary models).

The computation of expectations in the E-step is of the same order as an order two semi-markov chain model using hidden state labels of cardinality ($J \times 3$ = number of source morphemes times number of target morpheme types). The running time of the forward and backward dynamic programming passes is $T \times l^2 \times (3J)^2$, where T is the length of the target sentence in characters, J is the number of source morphemes, and l is the maximum morpheme length. Space does not permit the complete listing of the dynamic programming solution but it is not hard to derive by starting from the dynamic program for the IBM-1 like tokenization model of (Chung and Gildea, 2009) and extending it to account for the higher order on morphemes and the factored alignment state space.

Even though the inference algorithm is low polynomial it is still much more expensive than the inference for an HMM model for word-alignment without segmentation. To reduce the running time of the model we limit the space of considered morpheme boundaries as follows:

Given the target side of the corpus, we derive a list of K most frequent prefixes and suffixes using a simple trie-based method proposed by (Schone and Jurafsky, 2000).[2] After we determine a list of allowed prefixes and suffixes we restrict our model to allow only segmentations of the form : ((*p**)*r*(*s**))+ where *p* and *s* belong to the allowed prefixes and suffixes and *r* can match any substring.

We determine the number of prefixes and suffixes to consider using the maximum recall achievable by limiting the segmentation points in this way. Restricting the allowable segmentations in this way not only improves the speed of inference but also leads to improvements in segmentation accuracy.

[2]Words are inserted into a trie with each complete branch naturally identifying a potential suffix, inclusive of its sub-branches. The list comprises of the K most frequent of these complete branches. Inserting the reversed words will then yield potential *prefixes*.

3 Evaluation

For a majority of our testing we borrow the parallel phrases corpus used in previous work (Snyder and Barzilay, 2008), which we refer to as S&B. The corpus consists of 6,139 short phrases drawn from English, Hebrew, and Arabic translations of the Bible. We use an unmodified version of this corpus for the purpose of comparing morphological segmentation accuracy. For evaluating morpheme alignment accuracy, we have also augmented the English/Arabic subset of the corpus with a gold standard alignment between morphemes. Here morphological segmentations were obtained using the previously-annotated gold standard Arabic morphological segmentation, while the English was preprocessed with a morphological analyzer and then further hand annotated with corrections by two native speakers. Morphological alignments were manually annotated. Additionally, we evaluate monolingual segmentation models on the full Arabic Treebank (ATB), also used for unsupervised morpheme segmentation in (Poon et al., 2009).

4 Results

4.1 Morpheme Segmentation

We begin by evaluating a series of models which are simplifications of our complete model, to assess the impact of individual modeling decisions. We focus first on a monolingual setting, where the source sentence aligned to each target sentence is empty.

Unigram Model with Length Penalty

The first model we study is the unigram monolingual segmentation model using an exponential length penalty as proposed by (Liang and Klein, 2009; Chung and Gildea, 2009), which has been shown to be quite accurate. We refer to this model as Model-UP (for unigram with penalty). It defines the probability of a target morpheme sequence as follows: $(\mu_1 \dots \mu_I) = (1-\theta) \prod_{i=1}^{I} \theta P_T(\mu_i) LP(|\mu_i|)$

This model can be (almost) recovered as a special case of our full model, if we drop the transition and word boundary probabilities, do not model morpheme types, and use no conditioning for the morpheme translation model. The only parameter not present in our model is the probability θ of generating a morpheme as opposed to stopping to gener-

ate morphemes (with probability $1 - \theta$). We experimented with this additional parameter, but found it had no significant impact on performance, and so we do not report results including it.

We select the value of the length penalty power by a gird search in the range 1.1 to 2.0, using .1 increments and choosing the values resulting in best performance on a development set containing 500 phrase pairs for each language. We also select the optimal number of prefixes/suffixes to consider by measuring performance on the development set. [3]

Morpheme Type Models

The next model we consider is similar to the unigram model with penalty, but introduces the use of the hidden ta states which indicate only morpheme types in the monolingual setting. We use the ta states and test different configurations to derive the best set of features that can be used in the distortion model utilizing these states, and the morpheme translation model. We consider two variants: (1) Model-HMMP-basic (for HMM model with length penalty), which includes the hidden states but uses them with a simple uniform transition matrix $P(ta_i|ta_{i-1}, b_{i-1})$ (uniform over allowable transitions but forbidding the prefixes from transitioning directly to suffixes, and preventing suffixes from immediately following a word boundary), and (2) a richer model Model-HMMP which is allowed to learn a log-linear distortion model and a feature rich translation model as detailed in the model definition. This model is allowed to use word boundary information for conditioning (because word boundaries are observed), but does not include the P_B predictive word boundary distribution.

Full Model with Word Boundaries

Finally we consider our full monolingual model which also includes the distribution predicting word boundary variables b_i. We term this model Model-FullMono. We detail the best context features for the conditional P_D distribution for each language. We initialize this model with the morpheme translation unigram distribution of ModelHMMP-basic, trained for 5 iterations.

Table 4 details the test set results of the different model configurations, as well as previously reported results on these datasets. For our main results we use the automatically derived list of prefixes and suffixes to limit segmentation points. The names of models that use such limited lists are prefixed by Dict in the Table. For comparison, we also report the results achieved by models that do not limit the segmentation points in this way.

As we can see the unigram model with penalty, Dict-Model-UP, is already very strong, especially on the S&B Arabic dataset. When the segmentation points are not limited, its performance is much worse. The introduction of hidden morpheme states in Dict-HMMP-basic gives substantial improvement on Arabic and does not change results much on the other datasets. A small improvement is observed for the unconstrained models.[4] When our model includes all components except word boundary prediction, Dict-Model-HMMP, the results are substantially improved on all languages. Model-HMMP is also the first unconstrained model in our sequence to approach or surpass previous state-of-the-art segmentation performance.

Finally, when the full model Dict-MonoFull is used, we achieve a substantial improvement over the previous state-of-the-art results on all three corpora, a 6.5 point improvement on Arabic, 6.2 point improvement on Hebrew, and a 9.3 point improvement on ATB. The best configuration of this model uses the same distortion model for all languages: using the morph state transition and boundary features. The translation models used only t_i for Hebrew and ATB and t_i and μ_{i-1} for Arabic. Word boundary was predicted using t_i in Arabic and Hebrew, and additionally using b_{i-1} and b_{i-2} for ATB. The unconstrained models without affix dictionaries are also very strong, outperforming previous state-of-the-art models. For ATB, the unconstrained model slightly outperforms the constrained one.

The segmentation errors made by this system shed light on how it might be improved. We find the dis-

[3] For the S&B Arabic dataset, we selected to use seven prefixes and seven suffixes, which correspond to maximum achievable recall of 95.3. For the S&B Hebrew dataset, we used six prefixes and six suffixes, for a maximum recall of 94.3. The Arabic treebank data required a larger number of affixes: we used seven prefixes and 20 suffixes, for a maximum recall of 98.3.

[4] Note that the inclusion of states in HMMP-basic only serves to provide a different distribution over the number of morphemes in a word, so it is interesting it can have a positive impact.

	Arabic			Hebrew			ATB		
	P	R	F1	P	R	F1	P	R	F1
UP	88.1	55.1	67.8	43.2	87.6	57.9	79.0	54.6	64.6
Dict-UP	85.8	73.1	78.9	57.0	79.4	66.3	61.6	91.0	73.5
HMMP-basic	83.3	58.0	68.4	43.5	87.8	58.2	79.0	54.9	64.8
Dict-HMMP-basic	84.8	76.3	80.3	56.9	78.8	66.1	69.3	76.2	72.6
HMMP	73.6	76.9	75.2	70.2	73.0	71.6	94.0	76.1	84.1
Dict-HMMP	82.4	81.3	81.8	62.7	77.6	69.4	85.2	85.8	85.5
MonoFull	80.5	87.3	83.8	72.2	71.7	72.0	86.2	88.5	**87.4**
Dict-MonoFull	86.1	83.2	**84.6**	73.7	72.5	**73.1**	92.9	81.8	87.0
Poon et. al	76.0	80.2	78.1	67.6	66.1	66.9	88.5	69.2	77.7
S&B-Best	67.8	77.3	72.2	64.9	62.9	63.9	–	–	–
Morfessor	71.1	60.5	65.4	65.4	57.7	61.3	77.4	72.6	74.9

Figure 4: Results on morphological segmentation achieved by monolingual variants of our model (top) with results from prior work are included for comparison (bottom). Results from models with a small, automatically-derived list of possible prefixes and suffixes are labeled as "Dict-" followed by the model name.

tributions over the frequencies of particular errors follow a Zipfian skew across both S&B datasets, with the Arabic being more pronounced (the most frequent error being made 27 times, with 627 errors being made just once) in comparison with the Hebrew (with the most frequent error being made 19 times, and with 856 isolated errors). However, in both the Arabic and Hebrew S&B tasks we find that a tendency to over-segment certain characters off of their correct morphemes and on to other frequently occurring, yet incorrect, particles is actually the cause of many of these isolated errors. In Arabic the system tends to over segment the character *aleph* (totally about 300 errors combined). In Hebrew the source of error is not as overwhelmingly directed at a single character, but *yod* and *he*, the latter functioning quite similarly to the problematic Arabic character and frequently turn up in the corresponding places of cognate words in Biblical texts.

We should note that our models select a large number of hyper-parameters on an annotated development set, including length penalty, hierarchical smoothing parameters α, and the subset of variables to use in each of three component sub-models. This might in part explain their advantage over previous-state-of-the-art models, which might use fewer (e.g. (Poon et al., 2009) and (Snyder and Barzilay, 2008)) or no specifically tuned for these datasets hyper-parameters (Morfessor (Creutz and Lagus, 2007)).

4.2 Alignment

Next we evaluate our full bilingual model and a simpler variant on the task of word alignment. We use the morpheme-level annotation of the S&B English-Arabic dataset and project the morpheme alignments to word alignments. We can thus compare alignment performance of the results of different segmentations. Additionally, we evaluate against a state-of-the-art word alignment system WDHMM (He, 2007), which performs comparably or better than IBM-Model4. The table in Figure 5 presents the results. In addition to reporting alignment error rate for different segmentation models, we report their morphological segmentation F1.

The word-alignment WDHMM model performs best when aligning English words to Arabic words (using Arabic as source). In this direction it is able to capture the many-to-one correspondence between English words and arabic morphemes. When we combine alignments in both directions using the standard grow-diag-final method, the error goes up.

We compare the (Chung and Gildea, 2009) model (termed Model-1) to our full bilingual model. We can recover Model-1 similarly to Model-UP, except now every morpheme is conditioned on an aligned source morpheme. Our full bilingual model outperforms Model-1 in both AER and segmentation F1. The specific form of the full model was selected as in the previous experiments, by choosing the model with best segmentations of the development set.

For Arabic, the best model conditions target mor-

	Arabic						Hebrew		
	Align P	Align R	AER	P	R	F1	P	R	F1
Model-1 (C&G 09)	91.6	81.2	13.9	72.4	76.2	74.3	61.0	71.8	65.9
Bilingual full	91.0	88.3	10.3	90.0	72.0	80.0	63.3	71.2	67.0
WDHMM E-to-A	82.4	96.7	11.1						
WDHMM GDF	82.1	94.6	12.1						

Figure 5: Alignment Error Rate (AER) and morphological segmentation F1 achieved by bilingual variants of our model. AER performance of WDHMM is also reported. Gold standard alignments are not available for the Hebrew data set.

phemes on source morphemes only, uses the boundary model with conditioning on number of morphemes in the word, aligned source part-of-speech, and type of target morpheme. The distortion model uses both morpheme and word-based absolute distortion, binned distortion, morpheme types of states, and aligned source-part-of-speech tags. Our best model for Arabic outperforms WDHMM in word alignment error rate. For Hebrew, the best model uses a similar boundary model configuration but a simpler uniform transition distortion distribution.

Note that the bilingual models perform worse than the monolingual ones in segmentation F1. This finding is in line with previous work showing that the best segmentation for MT does not necessarily agree with a particular linguistic convention about what morphemes should contain (Chung and Gildea, 2009; Habash and Sadat, 2006), but contradicts other results (Snyder and Barzilay, 2008). Further experimentation is required to make a general claim.

We should note that the Arabic dataset used for word-alignment evaluation is unconventionally small and noisy (the sentences are very short phrases, automatically extracted using GIZA++). Thus the phrases might not be really translations, and the sentence length is much smaller than in standard parallel corpora. This warrants further model evaluation in a large-scale alignment setting.

5 Related Work

This work is most closely related to the unsupervised tokenization and alignment models of Chung and Gildea (2009), Xu et al. (2008), Snyder and Barzilay (2008), and Nguyen et al. (2010).

Chung & Gildea (2009) introduce a unigram model of tokenization based on IBM Model-1,which is a special case of our model. Snyder and Barzilay (2008) proposes a hierarchical Bayesian model that combines the learning of monolingual segmentations and a cross-lingual alignment; their model is very different from ours.

Incorporating morphological information into MT has received reasonable attention. For example, Goldwater & McClosky (2005) show improvements when preprocessing Czech input to reflect a morphological decomposition using combinations of lemmatization, pseudowords, and morphemes. Yeniterzi and Oflazer (2010) bridge the morphological disparity between languages in a unique way by effectively aligning English syntactic elements (function words connected by dependency relations) to Turkish morphemes, using rule-based postprocessing of standard word alignment. Our work is partly inspired by that work and attempts to automate both the morpho-syntactic alignment and morphological analysis tasks.

6 Conclusion

We have described an unsupervised model for morpheme segmentation and alignment based on Hidden Semi-Markov Models. Our model makes use of linguistic information to improve alignment quality. On the task of monolingual morphological segmentation it produces a new state-of-the-art level on three datasets. The model shows quantitative improvements in both word segmentation and word alignment, but its true potential lies in its finer-grained interpretation of word alignment, which will hopefully yield improvements in translation quality.

Acknowledgements

We thank the ACL reviewers for their valuable comments on earlier versions of this paper, and Michael J. Burling for his contributions as a corpus annotator and to the Arabic aspects of this paper.

References

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Cote, John DeNero, and Dan Klein. 2010. Unsupervised learning with features. In *Proceedings of the North American chapter of the Association for Computational Linguistics (NAACL)*.

Tagyoung Chung and Daniel Gildea. 2009. Unsupervised tokenization for machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process*.

Nizar Habash and Fatiha Sadat. 2006. Arabic preprocessing schemes for statistical machine translation. In *North American Chapter of the Association for Computational Linguistics*.

Xiaodong He. 2007. Using word-dependent transition models in HMM based word alignment for statistical machine translation. In *ACL 2nd Statistical MT workshop*, pages 80–87.

Philip Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *AMTA*.

P. Liang and D. Klein. 2009. Online EM for unsupervised models. In *North American Association for Computational Linguistics (NAACL)*.

ThuyLinh Nguyen, Stephan Vogel, and Noah A. Smith. 2010. Nonparametric word segmentation for machine translation. In *Proceedings of the International Conference on Computational Linguistics*.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *North American Chapter of the Association for Computation Linguistics - Human Language Technologies 2009 conference (NAACL/HLT-09)*.

Patrick Schone and Daniel Jurafsky. 2000. Knowlege-free induction of morphology using latent semantic analysis. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-2000)*.

Benjamin Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *ACL*.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *In COLING 96: The 16th Int. Conf. on Computational Linguistics*.

Jia Xu, Jianfeng Gao, Kristina Toutanova, and Hermann Ney. 2008. Bayesian semi-supervised chinese word segmentation for statistical machine translation. In *COLING*.

Reyyan Yeniterzi and Kemal Oflazer. 2010. Syntax-to-morphology mapping in factored phrase-based statistical machine translation from english to turkish. In *Proceedings of Association of Computational Linguistics*.

A Graph Approach to Spelling Correction in Domain-Centric Search

Zhuowei Bao
University of Pennsylvania
Philadelphia, PA 19104, USA
zhuowei@cis.upenn.edu

Benny Kimelfeld
IBM Research–Almaden
San Jose, CA 95120, USA
kimelfeld@us.ibm.com

Yunyao Li
IBM Research–Almaden
San Jose, CA 95120, USA
yunyaoli@us.ibm.com

Abstract

Spelling correction for keyword-search queries is challenging in restricted domains such as personal email (or desktop) search, due to the scarcity of query logs, and due to the specialized nature of the domain. For that task, this paper presents an algorithm that is based on statistics from the corpus data (rather than the query log). This algorithm, which employs a simple graph-based approach, can incorporate different types of data sources with different levels of reliability (e.g., email subject vs. email body), and can handle complex spelling errors like splitting and merging of words. An experimental study shows the superiority of the algorithm over existing alternatives in the email domain.

1 Introduction

An abundance of applications require *spelling correction*, which (at the high level) is the following task. The user intends to type a chunk $\mathbf{q}$ of text, but types instead the chunk $\mathbf{s}$ that contains *spelling errors* (which we discuss in detail later), due to uncareful typing or lack of knowledge of the exact spelling of $\mathbf{q}$. The goal is to restore $\mathbf{q}$, when given $\mathbf{s}$. Spelling correction has been extensively studied in the literature, and we refer the reader to comprehensive summaries of prior work (Peterson, 1980; Kukich, 1992; Jurafsky and Martin, 2000; Mitton, 2010). The focus of this paper is on the special case where $\mathbf{q}$ is a *search query*, and where $\mathbf{s}$ instead of $\mathbf{q}$ is submitted to a search engine (with the goal of retrieving documents that match the search query $\mathbf{q}$). Spelling correction for search queries is important, because a significant portion of posed queries may be misspelled (Cucerzan and Brill, 2004). Effective spelling correction has a major effect on the experience and effort of the user, who is otherwise required to ensure the exact spellings of her queries. Furthermore, it is critical when the exact spelling is unknown (e.g., person names like *Schwarzenegger*).

1.1 Spelling Errors

The more common and studied type of spelling error is *word-to-word* error: a single word w is misspelled into another single word w'. The specific spelling errors involved include omission of a character (e.g., `atachment`), inclusion of a redundant character (e.g., `attachement`), and replacement of characters (e.g., `attachemnt`). The fact that w' is a misspelling of (and should be corrected to) w is denoted by $w' \rightarrow w$ (e.g., `atachment` $\rightarrow$ `attachment`).

Additional common spelling errors are *splitting* of a word, and *merging* two (or more) words:

- `attach ment` $\rightarrow$ `attachment`
- `emailattachment` $\rightarrow$ `email attachment`

Part of our experiments, as well as most of our examples, are from the domain of (personal) email search. An email from the *Enron email collection* (Klimt and Yang, 2004) is shown in Figure 1. Our running example is the following misspelling of a search query, involving multiple types of errors.

`sadeep kohli excellatach ment` $\rightarrow$
`sandeep kohli excel attachment` (1)

In this example, correction entails fixing `sadeep`, splitting `excellatach`, fixing `excell`, merging `atach ment`, and fixing `atachment`. Beyond the complexity of errors, this example also illustrates other challenges in spelling correction for search. We need to identify not only that `sadeep` is misspelled, but also that `kohli` is correctly spelled. Just having `kohli` in a dictionary is not enough.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 905–914,
Portland, Oregon, June 19-24, 2011.

Subject: Follow-Up on Captive Generation
From: sandeep.kohli@enron.com
X-From: Sandeep Kohli
X-To: Stinson Gibner@ECT, Vince J Kaminski@ECT

Vince/Stinson,

Please find below two attachemnts. The Excell spreadsheet shows some calculations... The seond attachement (Word) has the wordings that I think we can send in to the press...

I am availabel on mobile if you have questions o clarifications...

Regards,
Sandeep.

Figure 1: Enron email (misspelled words are underlined)

For example, in `kohli coupons` the user may very well mean `kohls coupons` if Sandeep Kohli has nothing to do with coupons (in contrast to the store chain Kohl's). A similar example is the word `nail`, which is a legitimate English word, but in the context of email the query `nail box` is likely to be a misspelling of `mail box` (unless nail boxes are indeed relevant to the user's email collection). Finally, while the word `kohli` is relevant to some email users (e.g., Kohli's colleagues), it may have no meaning at all to other users.

1.2 Domain Knowledge

The common approach to spelling correction utilizes statistical information (Kernighan et al., 1990; Schierle et al., 2007; Mitton, 2010). As a simple example, if we want to avoid maintaining a manually-crafted dictionary to accommodate the wealth of new terms introduced every day (e.g., `ipod` and `ipad`), we may decide that `atachment` is a misspelling of `attachment` due to both the (relative) proximity between the words, and the fact that `attachment` is significantly more popular than `atachment`. As another example, the fact that the expression `sandeep kohli` is frequent in the domain increases our confidence in `sadeep kohli` $\rightarrow$ `sandeep kohli` (rather than, e.g., `sadeep kohli` $\rightarrow$ `sudeep kohli`). One can further note that, in email search, the fact that Sandeep Kohli sent multiple excel attachments increases our confidence in `excell` $\rightarrow$ `excel`.

A source of statistics widely used in prior work is the query log (Cucerzan and Brill, 2004; Ahmad and Kondrak, 2005; Li et al., 2006a; Chen et al., 2007; Sun et al., 2010). However, while query logs are abundant in the context of Web search, in many other search applications (e.g. email search, desktop search, and even small-enterprise search) query logs are too scarce to provide statistical information that is sufficient for effective spelling correction. Even an email provider of a massive scale (such as GMail) may need to rely on the (possibly tiny) query log of the single user at hand, due to privacy or security concerns; moreover, as noted earlier about `kohli`, the statistics of one user may be relevant to one user, while irrelevant to another.

The focus of this paper is on spelling correction for search applications like the above, where query-log analysis is impossible or undesirable (with email search being a prominent example). Our approach relies mainly on the corpus data (e.g., the collection of emails of the user at hand) and external, generic dictionaries (e.g., English). As shown in Figure 1, the corpus data may very well contain misspelled words (like query logs do), and such noise is a part of the challenge. Relying on the corpus has been shown to be successful in spelling correction for *text cleaning* (Schierle et al., 2007). Nevertheless, as we later explain, our approach can still incorporate query-log data as features involved in the correction, as well as means to refine the parameters.

1.3 Contribution and Outline

As said above, our goal is to devise spelling correction that relies on the corpus. The corpus often contains various types of information, with different levels of reliability (e.g., n-grams from email subjects and sender information, vs. those from email bodies). The major question is how to effectively exploit that information while addressing the various types of spelling errors such as those discussed in Section 1.1. The key contribution of this work is a novel graph-based algorithm, MaxPaths, that handles the different types of errors and incorporates the corpus data in a uniform (and simple) fashion. We describe MaxPaths in Section 2. We evaluate the effectiveness of our algorithm via an experimental study in Section 3. Finally, we make concluding remarks and discuss future directions in Section 4.

2 Spelling-Correction Algorithm

In this section, we describe our algorithm for spelling correction. Recall that given a search query

$\mathbf{s}$ of a user who intends to phrase $\mathbf{q}$, the goal is to find $\mathbf{q}$. Our corpus is essentially a collection $\mathbf{D}$ of unstructured or *semistructured documents*. For example, in email search such a document is an email with a title, a body, one or more recipients, and so on. As conventional in spelling correction, we devise a scoring function $\text{score}_{\mathbf{D}}(\mathbf{r} \mid \mathbf{s})$ that estimates our confidence in $\mathbf{r}$ being the correction of $\mathbf{s}$ (i.e., that $\mathbf{r}$ is equal to $\mathbf{q}$). Eventually, we suggest a sequence $\mathbf{r}$ from a set $\mathcal{C}_{\mathbf{D}}(\mathbf{s})$ of *candidates*, such that $\text{score}_{\mathbf{D}}(\mathbf{r} \mid \mathbf{s})$ is maximal among all the candidates in $\mathcal{C}_{\mathbf{D}}(\mathbf{s})$. In this section, we describe our graph-based approach to finding $\mathcal{C}_{\mathbf{D}}(\mathbf{s})$ and to determining $\text{score}_{\mathbf{D}}(\mathbf{r} \mid \mathbf{s})$.

We first give some basic notation. We fix an alphabet $\mathbf{\Sigma}$ of *characters* that does not include any of the conventional whitespace characters. By $\mathbf{\Sigma}^*$ we denote the set of all the *words*, namely, finite sequences over $\mathbf{\Sigma}$. A *search query* $\mathbf{s}$ is a sequence $w_1, \ldots, w_n$, where each w_i is a word. For convenience, in our examples we use whitespace instead of comma (e.g., `sandeep kohli` instead of `sandeep, kohli`). We use the *Damerau-Levenshtein* edit distance (as implemented by the *Jazzy* tool) as our primary edit distance between two words $r_1, r_2 \in \mathbf{\Sigma}^*$, and we denote this distance by $ed(r_1, r_2)$.

2.1 Word-Level Correction

We first handle a restriction of our problem, where the search query is a single word w (rather than a general sequence $\mathbf{s}$ of words). Moreover, we consider only candidate suggestions that are words (rather than sequences of words that account for the case where w is obtained by merging keywords). Later, we will use the solution for this restricted problem as a basic component in our algorithm for the general problem.

Let $\mathbf{U}_{\mathbf{D}} \subseteq \mathbf{\Sigma}^*$ be a finite *universal lexicon*, which (conceptually) consists of all the words in the corpus $\mathbf{D}$. (In practice, one may want add to $\mathbf{D}$ words of auxiliary sources, like English dictionary, and to filter out noisy words; we did so in the site-search domain that is discussed in Section 3.) The set $\mathcal{C}_{\mathbf{D}}(w)$ of candidates is defined by

$$\mathcal{C}_{\mathbf{D}}(w) \stackrel{\text{def}}{=} \{w\} \cup \{w' \in \mathbf{U}_{\mathbf{D}} \mid ed(w, w') \leq \delta\}\,.$$

for some fixed number δ. Note that $\mathcal{C}_{\mathbf{D}}(w)$ contains w even if w is misspelled; furthermore, $\mathcal{C}_{\mathbf{D}}(w)$ may contain other misspelled words (with a small edit distance to w) that appear in $\mathbf{D}$.

Table 1: Feature set $\mathsf{WF}_{\mathbf{D}}$ in email search

Basic Features
$ed(w, w')$: weighted Damerau-Levenshtein edit distance
$ph(w, w')$: 1 if w and w' are phonetically equal, 0 otherwise
$\text{english}(w')$: 1 is w' is in English, 0 otherwise
Corpus-Based Features
$logfreq(w')$): logarithm of #occurrences of w' in the corpus
Domain-Specific Features
$\text{subject}(w')$: 1 if w' is in some "Subject" field, 0 otherwise
$\text{from}(w')$: 1 if w' is in some "From" field, 0 otherwise
$\text{xfrom}(w')$: 1 if w' is in some "X-From" field, 0 otherwise

We now define $\text{score}_{\mathbf{D}}(w' \mid w)$. Here, our corpus $\mathbf{D}$ is translated into a set $\mathsf{WF}_{\mathbf{D}}$ of *word features*, where each feature $f \in \mathsf{WF}_{\mathbf{D}}$ gives a scoring function $\text{score}_f(w' \mid w)$. The function $\text{score}_{\mathbf{D}}(w' \mid w)$ is simply a linear combination of the $\text{score}_f(w' \mid w)$:

$$\text{score}_{\mathbf{D}}(w' \mid w) \stackrel{\text{def}}{=} \sum_{f \in \mathsf{WF}_{\mathbf{D}}} a_f \cdot \text{score}_f(w' \mid w)$$

As a concrete example, the features of $\mathsf{WF}_{\mathbf{D}}$ we used in the email domain are listed in Table 1; the resulting $\text{score}_f(w'|w)$ is in the spirit of the *noisy channel model* (Kernighan et al., 1990). Note that additional features could be used, like ones involving the *stems* of w and w', and even query-log statistics (when available). Rather than manually tuning the parameters a_f, we learned them using the well known *Support Vector Machine*, abbreviated *SVM* (Cortes and Vapnik, 1995), as also done by Schaback and Li (2007) for spelling correction. We further discuss this learning step in Section 3.

We fix a natural number k, and in the sequel we denote by $\text{top}_{\mathbf{D}}(w)$ a set of k words $w' \in \mathcal{C}_{\mathbf{D}}(w)$ with the highest $\text{score}_{\mathbf{D}}(w' \mid w)$. If $|\mathcal{C}_{\mathbf{D}}(w)| < k$, then $\text{top}_{\mathbf{D}}(w)$ is simply $\mathcal{C}_{\mathbf{D}}(w)$.

2.2 Query-Level Correction: MaxPaths

We now describe our algorithm, MaxPaths, for spelling correction. The input is a (possibly misspelled) search query $\mathbf{s} = s_1, \ldots, s_n$. As done in the word-level correction, the algorithm produces a set $\mathcal{C}_{\mathbf{D}}(\mathbf{s})$ of suggestions and determines the values

Algorithm 1 MaxPaths

Input: a search query $\mathbf{s}$
Output: a set $\mathcal{C}_\mathbf{D}(\mathbf{s})$ of candidate suggestions $\mathbf{r}$, ranked by $\text{score}_\mathbf{D}(\mathbf{r} \mid \mathbf{s})$

1: Find the *strongly plausible tokens*
2: Construct the *correction graph*
3: Find top-k *full paths* (with the largest weights)
4: Re-rank the paths by *word correlation*

$\text{score}_\mathbf{D}(\mathbf{r} \mid \mathbf{s})$, for all $\mathbf{r} \in \mathcal{C}_\mathbf{D}(\mathbf{s})$, in order to rank $\mathcal{C}_\mathbf{D}(\mathbf{s})$. A high-level overview of MaxPaths is given in the pseudo-code of Algorithm 1. In the rest of this section, we will detail each of the four steps in Algorithm 1. The name MaxPaths will become clear towards the end of this section.

We use the following notation. For a word $w = c_1 \cdots c_m$ of m characters c_i and integers $i < j$ in $\{1, \ldots, m+1\}$, we denote by $w_{[i,j)}$ the word $c_i \cdots c_{j-1}$. For two words $w_1, w_2 \in \mathbf{\Sigma}^*$, the word $w_1 w_2 \in \mathbf{\Sigma}^*$ is obtained by concatenating w_1 and w_2. Note that for the search query $\mathbf{s} = s_1, \ldots, s_n$ it holds that $s_1 \cdots s_n$ is a single word (in $\mathbf{\Sigma}^*$). We denote the word $s_1 \cdots s_n$ by $\lfloor \mathbf{s} \rfloor$. For example, if $s_1 =$ `sadeep` and $s_2 =$ `kohli`, then $\mathbf{s}$ corresponds to the query `sadeep kohli` while $\lfloor \mathbf{s} \rfloor$ is the word `sadeepkohli`; furthermore, $\lfloor \mathbf{s} \rfloor_{[1,7)} =$ `sadeep`.

2.2.1 Plausible Tokens

To support merging and splitting, we first identify the possible *tokens* of the given query $\mathbf{s}$. For example, in `excellatach ment` we would like to identify `excell` and `atach ment` as tokens, since those are indeed the tokens that the user has in mind. Formally, suppose that $\lfloor \mathbf{s} \rfloor = c_1 \cdots c_m$. A *token* is a word $\lfloor \mathbf{s} \rfloor_{[i,j)}$ where $1 \leq i < j \leq m+1$. To simplify the presentation, we make the (often false) assumption that a token $\lfloor \mathbf{s} \rfloor_{[i,j)}$ uniquely identifies i and j (that is, $\lfloor \mathbf{s} \rfloor_{[i,j)} \neq \lfloor \mathbf{s} \rfloor_{[i',j')}$ if $i \neq i'$ or $j \neq j'$); in reality, we should define a token as a triple $(\lfloor \mathbf{s} \rfloor_{[i,j)}, i, j)$. In principle, every token $\lfloor \mathbf{s} \rfloor_{[i,j)}$ could be viewed as a possible word that user meant to phrase. However, such liberty would require our algorithm to process a search space that is too large to manage in reasonable time. Instead, we restrict to *strongly plausible tokens*, which we define next.

A token $w = \lfloor \mathbf{s} \rfloor_{[i,j)}$ is *plausible* if w is a word of $\mathbf{s}$, or there is a word $w' \in \mathcal{C}_\mathbf{D}(w)$ (as defined in Section 2.1) such that $\text{score}_\mathbf{D}(w' \mid w) > \epsilon$ for some fixed number ϵ. Intuitively, w is plausible if it is an original token of $\mathbf{s}$, or we have a high confidence in our word-level suggestion to correct w (note that the suggested correction for w can be w itself). Recall that $\lfloor \mathbf{s} \rfloor = c_1 \cdots c_m$. A *tokenization* of $\mathbf{s}$ is a sequence $j_1, \ldots, j_l$, such that $j_1 = 1$, $j_l = m+1$, and $j_i < j_{i+1}$ for $1 \leq i < l$. The tokenization $j_1, \ldots, j_l$ *induces* the tokens $\lfloor \mathbf{s} \rfloor_{[j_1,j_2)}, \ldots, \lfloor \mathbf{s} \rfloor_{[j_{l-1},j_l)}$. A tokenization is *plausible* if each of its induced tokens is plausible. Observe that a plausible token is not necessarily induced by any plausible tokenization; in that case, the plausible token is useless to us. Thus, we define a *strongly plausible token*, abbreviated *sp-token*, which is a token that is induced by some plausible tokenization. As a concrete example, for the query `excellatach ment`, the sp-tokens in our implementation include `excellatach`, `ment`, `excell`, and `atachment`.

As the first step (line 1 in Algorithm 1), we find the sp-tokens by employing an efficient (and fairly straightforward) dynamic-programming algorithm.

2.2.2 Correction Graph

In the next step (line 2 in Algorithm 1), we construct the *correction graph*, which we denote by $\mathcal{G}_\mathbf{D}(\mathbf{s})$. The construction is as follows.

We first find the set $\text{top}_\mathbf{D}(w)$ (defined in Section 2.1) for each sp-token w. Table 2 shows the sp-tokens and suggestions thereon in our running example. This example shows the actual execution of our implementation within email search, where $\mathbf{s}$ is the query `sadeep kohli excellatach ment`; for clarity of presentation, we omitted a few sp-tokens and suggested corrections. Observe that some of the corrections in the table are actually misspelled words (as those naturally occur in the corpus).

A node of the graph $\mathcal{G}_\mathbf{D}(\mathbf{s})$ is a pair $\langle w, w' \rangle$, where w is an sp-token and $w' \in \text{top}_\mathbf{D}(w)$. Recall our simplifying assumption that a token $\lfloor \mathbf{s} \rfloor_{[i,j)}$ uniquely identifies the indices i and j. The graph $\mathcal{G}_\mathbf{D}(\mathbf{s})$ contains a (directed) edge from a node $\langle w_1, w_1' \rangle$ to a node $\langle w_2, w_2' \rangle$ if w_2 immediately follows w_1 in $\lfloor \mathbf{q} \rfloor$; in other words, $\mathcal{G}_\mathbf{D}(\mathbf{s})$ has an edge from $\langle w_1, w_1' \rangle$ to $\langle w_2, w_2' \rangle$ whenever there exist indices i, j and k, such that $w_1 = \lfloor \mathbf{s} \rfloor_{[i,j)}$ and $w_2 = \lfloor \mathbf{s} \rfloor_{[j,k)}$. Observe that $\mathcal{G}_\mathbf{D}(\mathbf{s})$ is a *directed acyclic graph* (DAG).

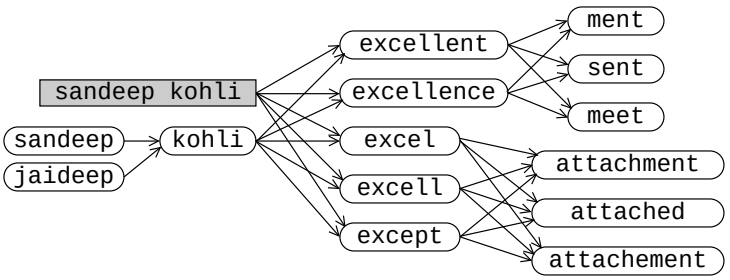


Figure 2: The graph $\mathcal{G}_{\mathbf{D}}(\mathbf{s})$

For example, Figure 2 shows $\mathcal{G}_{\mathbf{D}}(\mathbf{s})$ for the query `sadeep kohli excellatach ment`, with the sp-tokens w and the sets $\text{top}_{\mathbf{D}}(w)$ being those of Table 2. For now, the reader should ignore the node in the grey box (containing `sandeep kohli`) and its incident edges. For simplicity, in this figure we depict each node $\langle w, w' \rangle$ by just mentioning w'; the word w is in the first row of Table 2, above w'.

2.2.3 Top-k Paths

Let $P = \langle w_1, w'_1 \rangle \rightarrow \cdots \rightarrow \langle w_k, w'_k \rangle$ be a path in $\mathcal{G}_{\mathbf{D}}(\mathbf{s})$. We say that P is *full* if $\langle w_1, w'_1 \rangle$ has no incoming edges in $\mathcal{G}_{\mathbf{D}}(\mathbf{s})$, and $\langle w_k, w'_k \rangle$ has no outgoing edges in $\mathcal{G}_{\mathbf{D}}(\mathbf{s})$. An easy observation is that, since we consider only strongly plausible tokens, if P is full then $w_1 \cdots w_k = \lfloor \mathbf{s} \rfloor$; in that case, the sequence $w'_1, \ldots, w'_k$ is a suggestion for spelling correction, and we denote it by $crc(P)$. As an example, Figure 3 shows two full paths P_1 and P_2 in the graph $\mathcal{G}_{\mathbf{D}}(\mathbf{s})$ of Figure 2. The corrections $crc(P_i)$, for $i = 1, 2$, are `jaideep kohli excellent ment` and `sandeep kohli excel attachement`, respectively.

To obtain corrections $crc(P)$ with high quality, we produce a set of k full paths with the largest weights, for some fixed k; we denote this set by $\text{topPaths}_{\mathbf{D}}(\mathbf{s})$. The *weight* of a path P, denoted $\text{weight}(P)$, is the sum of the weights of all the nodes and edges in P, and we define the weights of nodes and edges next. To find these paths, we use a well known efficient algorithm (Eppstein, 1994).

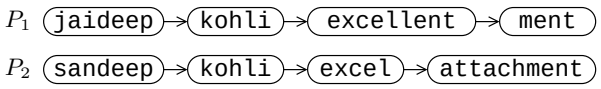


Figure 3: Full paths in the graph $\mathcal{G}_{\mathbf{D}}(\mathbf{s})$ of Figure 2

Consider a node $u = \langle w, w' \rangle$ of $\mathcal{G}_{\mathbf{D}}(\mathbf{s})$. In the construction of $\mathcal{G}_{\mathbf{D}}(\mathbf{s})$, zero or more merges of (part of) original tokens have been applied to obtain the token w; let $\#\text{merges}(w)$ be that number. Consider an edge e of $\mathcal{G}_{\mathbf{D}}(\mathbf{s})$ from a node $u_1 = \langle w_1, w'_1 \rangle$ to $u_2 = \langle w_2, w'_2 \rangle$. In $\mathbf{s}$, either w_1 and w_2 belong to different words (i.e., there is a whitespace between them) or not; in the former case define $\#\text{splits}(e) = 0$, and in the latter $\#\text{splits}(e) = 1$. We define:

$$\text{weight}(u) \stackrel{\text{def}}{=} \text{score}_{\mathbf{D}}(w' \mid w) + a_m \cdot \#\text{merges}(w)$$
$$\text{weight}(e) \stackrel{\text{def}}{=} a_s \cdot \#\text{splits}(e)$$

Note that a_m and a_s are negative, as they penalize for merges and splits, respectively. Again, in our implementations, we learned a_m and a_s by means of SVM.

Recall that $\text{topPaths}_{\mathbf{D}}(\mathbf{s})$ is the set of k full paths (in the graph $\mathcal{G}_{\mathbf{D}}(\mathbf{s})$) with the largest weights. From $\text{topPaths}_{\mathbf{D}}(\mathbf{s})$ we get the set $\mathcal{C}_{\mathbf{D}}(\mathbf{s})$ of candidate suggestions:

$$\mathcal{C}_{\mathbf{D}}(\mathbf{s}) \stackrel{\text{def}}{=} \{crc(P) \mid P \in \text{topPaths}_{\mathbf{D}}(\mathbf{s})\}.$$

2.2.4 Word Correlation

To compute $\text{score}_{\mathbf{D}}(\mathbf{r}|\mathbf{s})$ for $\mathbf{r} \in \mathcal{C}_{\mathbf{D}}(\mathbf{s})$, we incorporate *correlation* among the words of $\mathbf{r}$. Intuitively, we would like to reward a candidate with pairs of words that are likely to co-exist in a query. For that, we assume a (symmetric) numerical function $crl(w'_1, w'_2)$ that estimates the extent to which the words w'_1 and w'_2 are correlated. As an example, in the email domain we would like $crl(\texttt{kohli}, \texttt{excel})$ to be high if Kohli sent many emails with excel attachments. Our implementation of $crl(w'_1, w'_2)$ essentially employs *pointwise mutual information* that has also been used in (Schierle et al., 2007), and that

Table 2: $\text{top}_{\mathbf{D}}(w)$ for sp-tokens w

sadeep	kohli	excellatach	ment	excell	atachment
sandeep	kohli	excellent	ment	excel	attachment
jaideep		excellence	sent	excell	attached
			meet	except	attachement

compares the number of documents (emails) containing w'_1 and w'_2 separately and jointly.

Let $P \in \text{topPaths}_{\mathbf{D}}(\mathbf{s})$ be a path. We denote by $crl(P)$ a function that aggregates the numbers $crl(w'_1, w'_2)$ for nodes $\langle w_1, w'_1 \rangle$ and $\langle w_2, w'_2 \rangle$ of P (where $\langle w_1, w'_1 \rangle$ and $\langle w_2, w'_2 \rangle$ are not necessarily neighbors in P). Over the email domain, our $crl(P)$ is the minimum of the $crl(w'_1, w'_2)$. We define $\text{score}_{\mathbf{D}}(P) = \text{weight}(P) + crl(P)$. To improve the performance, in our implementation we learned again (re-trained) all the parameters involved in $\text{score}_{\mathbf{D}}(P)$.

Finally, as the top suggestions we take $crc(P)$ for full paths P with highest $\text{score}_{\mathbf{D}}(P)$. Note that $crc(P)$ is not necessarily injective; that is, there can be two full paths $P_1 \neq P_2$ satisfying $crc(P_1) = crc(P_2)$. Thus, in effect, $\text{score}_{\mathbf{D}}(\mathbf{r} \mid \mathbf{s})$ is determined by the best evidence of $\mathbf{r}$; that is,

$$\text{score}_{\mathbf{D}}(\mathbf{r} \mid \mathbf{s}) \stackrel{\text{def}}{=} \max\{\text{score}_{\mathbf{D}}(P) \mid crc(P) = \mathbf{r} \wedge P \in \text{topPaths}_{\mathbf{D}}(\mathbf{s})\}.$$

Note that our final scoring function essentially views P as a *clique* rather than a path. In principle, we could define $\mathcal{G}_{\mathbf{D}}(\mathbf{s})$ in a way that we would extract the maximal cliques directly without finding $\text{topPaths}_{\mathbf{D}}(\mathbf{s})$ first. However, we chose our method (finding top paths first, and then re-ranking) to avoid the inherent computational hardness involved in finding maximal cliques.

2.3 Handling Expressions

We now briefly discuss our handling of frequent n-grams (expressions). We handle n-grams by introducing new nodes to the graph $\mathcal{G}_{\mathbf{D}}(\mathbf{s})$; such a new node u is a pair $\langle \mathbf{t}, \mathbf{t}' \rangle$, where $\mathbf{t}$ is a sequence of n consecutive sp-tokens and $\mathbf{t}'$ is a n-gram. The weight of such a node u is rewarded for constituting a frequent or important n-gram. An example of such a node is in the grey box of Figure 2, where `sandeep kohli` is a bigram. Observe that `sandeep kohli` may be deemed an important bigram because it occurs as a sender of an email, and not necessarily because it is frequent.

An advantage of our approach is avoidance of over-scoring due to *conflicting n-grams*. For example, consider the query `textile import expert`, and assume that both `textile import` and `import export` (with an "o" rather than an "e") are frequent bigrams. If the user referred to the bigram `textile import`, then `expert` is likely to be correct. But if she meant for `import export`, then `expert` is misspelled. However, only one of these two options can hold true, and we would like `textile import export` to be rewarded only once—for the bigram `import export`. This is achieved in our approach, since a full path in $\mathcal{G}_{\mathbf{D}}(\mathbf{s})$ may contain either a node for `textile import` or a node for `import export`, but it cannot contain nodes for both of these bigrams.

Finally, we note that our algorithm is in the spirit of that of Cucerzan and Brill (2004), with a few inherent differences. In essence, a node in the graph they construct corresponds to what we denote here as $\langle w, w' \rangle$ in the special case where w is an actual word of the query; that is, no re-tokenization is applied. They can split a word by comparing it to a bigram. However, it is not clear how they can split into non-bigrams (without a huge index) and to handle simultaneous merging and splitting as in our running example (1). Furthermore, they translate bigram information into edge weights, which implies that the above problem of over-rewarding due to conflicting bigrams occurs.

3 Experimental Study

Our experimental study aims to investigate the effectiveness of our approach in various settings, as we explain next.

3.1 Experimental Setup

We first describe our experimental setup, and specifically the datasets and general methodology.

Datasets. The focus of our experimental study is on *personal email search*; later on (Section 3.6), we will consider (and give experimental results for) a totally different setting—site search over `www.ibm.com`, which is a massive and open domain. Our dataset (for the email domain) is obtained from

the Enron email collection (Bekkerman et al., 2004; Klimt and Yang, 2004). Specifically, we chose the three users with the largest number of emails. We refer to the three email collections by the last names of their owners: *Farmer*, *Kaminski* and *Kitchen*. Each user mailbox is a separate domain, with a separate corpus $\mathbf{D}$, that one can search upon. Due to the absence of real user queries, we constructed our dataset by conducting a user study, as described next.

For each user, we randomly sampled 50 emails and divided them into 5 disjoint sets of 10 emails each. We gave each 10-email set to a unique human subject that was asked to phrase two search queries for each email: one for the entire email content (*general query*), and the other for the `From` and `X-From` fields (*sender query*). (Figure 1 shows examples of the `From` and `X-From` fields.) The latter represents queries posed against a specific field (e.g., using "advanced search"). The participants were not told about the goal of this study (i.e., spelling correction), and the collected queries have no spelling errors. For generating spelling errors, we implemented a typo generator.[1] This generator extends an online typo generator (Seobook, 2010) that produces a variety of spelling errors, including skipped letter, doubled letter, reversed letter, skipped space (merge), missed key and inserted key; in addition, our generator produces inserted space (split). When applied to a search query, our generator adds random typos to each word, independently, with a specified probability p that is 50% by default. For each collected query (and for each considered value of p) we generated 5 misspelled queries, and thereby obtained 250 instances of misspelled general queries and 250 instances of misspelled sender queries.

Methodology. We compared the accuracy of MaxPaths (Section 2) with three alternatives. The first alternative is the open-source *Jazzy*, which is a widely used spelling-correction tool based on (weighted) edit distance. The second alternative is the spelling correction provided by *Google*. We provided Jazzy with our unigram index (as a dictionary). However, we were not able to do so with Google, as we used remote access via its Java API (Google, 2010); hence, the Google tool is unaware of our domain, but is rather based on its own statistics (from the World Wide Web). The third alternative is what we call WordWise, which applies word-level correction (Section 2.1) to each input query term, independently. More precisely, WordWise is a simplified version of MaxPaths, where we forbid splitting and merging of words (i.e., only the original tokens are considered), and where we do not take correlation into account.

Our emphasis is on *correcting misspelled queries*, rather than *recognizing correctly spelled queries*, due to the role of spelling in a search engine: we wish to provide the user with the correct query upon misspelling, but there is no harm in making a suggestion for correctly spelled queries, except for visual discomfort. Hence, by default *accuracy* means the number of properly corrected queries (within the top-k suggestions) divided by the number of the misspelled queries. An exception is in Section 3.5, where we study the accuracy on correct queries.

Since MaxPaths and WordWise involve parameter learning (SVM), the results for them are consistently obtained by performing 5-folder *cross validation* over each collection of misspelled queries.

3.2 Fixed Error Probability

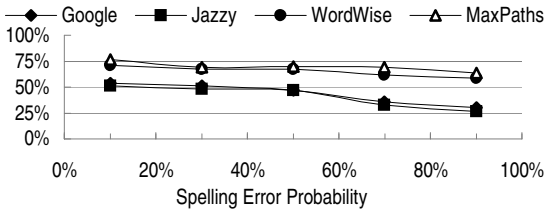
Here, we compare MaxPaths to the alternatives when the error probability p is fixed (0.5). We consider only the Kaminski dataset; the results for the other two datasets are similar. Figure 4(a) shows the accuracy, for general queries, of top-k suggestions for $k = 1$, $k = 3$ and $k = 10$. Note that we can get only one (top-1) suggestion from Google. As can be seen, MaxPaths has the highest accuracy in all cases. Moreover, the advantage of MaxPaths over the alternatives increases as k increases, which indicates potential for further improving MaxPaths.

Figure 4(b) shows the accuracy of top-k suggestions for sender queries. Overall, the results are similar to those of Figure 4(a), except that top-1 of both WordWise and MaxPaths has a higher accuracy in sender queries than in general queries. This is due to the fact that the dictionaries of person names and email addresses extracted from the `X-From` and `From` fields, respectively, provide strong features for the scoring function, since a sender query refers to these two fields. In addition, the accuracy of MaxPaths is further enhanced by exploiting the cor-

[1] The queries and our typo generator are publicly available at `https://dbappserv.cis.upenn.edu/spell/`.

(a) General queries (Kaminski)

(b) Sender queries (Kaminski)



(c) Varying error probability (Kaminski)

Figure 4: Accuracy for Kaminski (misspelled queries)

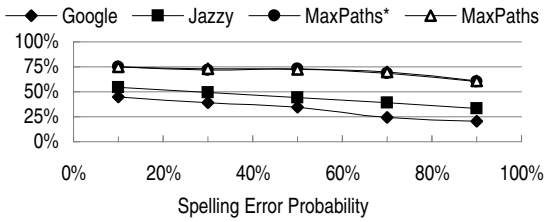
relation between the first and last name of a person.

3.3 Impact of Error Probability

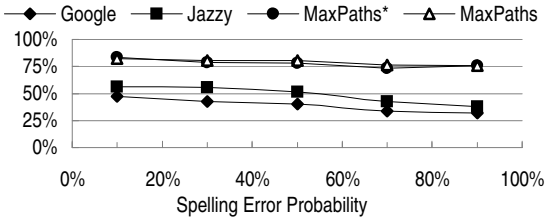
We now study the impact of the complexity of spelling errors on our algorithm. For that, we measure the accuracy while the error probability p varies from 10% to 90% (with gaps of 20%). The results are in Figure 4(c). Again, we show the results only for Kaminski, since we get similar results for the other two datasets. As expected, in all examined methods the accuracy decreases as p increases. Now, not only does MaxPaths outperform the alternatives, its decrease (as well as that of WordWise) is the mildest—13% as p increases from 10% to 90% (while Google and Jazzy decrease by 23% or more). We got similar results for the sender queries (and for each of the three users).

3.4 Adaptiveness of Parameters

Obtaining the labeled data needed for parameter learning entails a nontrivial manual effort. Ideally, we would like to learn the parameters of MaxPaths in one domain, and use them in similar domains.



(a) General queries (Farmer)



(b) Sender queries (Farmer)

Figure 5: Accuracy for Farmer (misspelled queries)

More specifically, our desire is to use the parameters learned over one corpus (e.g., the email collection of one user) on a second corpus (e.g., the email collection of another user), rather than learning the parameters again over the second corpus. In this set of experiments, we examine the feasibility of that approach. Specifically, we consider the user Farmer and observe the accuracy of our algorithm with two sets of parameters: the first, denoted by MaxPaths in Figures 5(a) and 5(b), is learned within the Farmer dataset, and the second, denoted by MaxPaths*, is learned within the Kaminski dataset. Figures 5(a) and 5(b) show the accuracy of the top-1 suggestion for general queries and sender queries, respectively, with varying error probabilities. As can be seen, these results mean good news—the accuracies of MaxPaths* and MaxPaths are extremely close (their curves are barely distinguishable, as in most cases the difference is smaller than 1%). We repeated this experiment for Kitchen and Kaminski, and got similar results.

3.5 Accuracy for Correct Queries

Next, we study the accuracy on *correct* queries, where the task is to recognize the given query as correct by returning it as the top suggestion. For each of the three users, we considered the $50 + 50$ (general + sender) collected queries (having no spelling errors), and measured the *accuracy*, which is the percentage of queries that are equal to the top sug-

Table 3: Accuracy for Correct Queries

Dataset	Google	Jazzy	MaxPaths
Kaminski (general)	90%	98%	94%
Kaminski (sender)	94%	98%	94%
Farmer (general)	96%	98%	96%
Farmer (sender)	96%	96%	92%
Kitchen (general)	86%	100%	92%
Kitchen (sender)	94%	100%	98%

gestion. Table 3 shows the results. Since Jazzy is based on edit distance, it almost always gives the input query as the top suggestion; the misses of Jazzy are for queries that contain a word that is not the corpus. MaxPaths is fairly close to the upper bound set by Jazzy. Google (having no access to the domain) also performs well, partly because it returns the input query if no reasonable suggestion is found.

3.6 Applicability to Large-Scale Site Search

Up to now, our focus has been on email search, which represents a restricted (closed) domain with specialized knowledge (e.g., sender names). In this part, we examine the effectiveness of our algorithm in a totally different setting—*large-scale site search* within `www.ibm.com`, a domain that is popular on a world scale. There, the accuracy of Google is very high, due to this domain's popularity, scale, and full accessibility on the Web. We crawled 10 million documents in that domain to obtain the corpus. We manually collected 1348 misspelled queries from the log of search issued against developerWorks (`www.ibm.com/developerworks/`) during a week. To facilitate the manual collection of these queries, we inspected each query with two or fewer search results, after applying a random permutation to those queries. Figure 6 shows the accuracy of top-k suggestions. Note that the performance of MaxPaths is very close to that of Google—only 2% lower for top-1. For $k = 3$ and $k = 10$, MaxPaths outperforms Jazzy and the top-1 of Google (from which we cannot obtain top-k for $k > 1$).

Figure 6: Accuracy for site search

3.7 Summary

To conclude, our experiments demonstrate various important qualities of MaxPaths. First, it outperforms its alternatives, in both accuracy (Section 3.2) and robustness to varying error complexities (Section 3.3). Second, the parameters learned in one domain (e.g., an email user) can be applied to similar domains (e.g., other email users) with essentially no loss in performance (Section 3.4). Third, it is highly accurate in recognition of correct queries (Section 3.5). Fourth, even when applied to large (open) domains, it achieves a comparable performance to the state-of-the-art Google spelling correction (Section 3.6). Finally, the higher performance of MaxPaths on top-3 and top-10 corrections suggests a potential for further improvement of top-1 (which is important since search engines often restrict their interfaces to only one suggestion).

4 Conclusions

We presented the algorithm MaxPaths for spelling correction in domain-centric search. This algorithm relies primarily on corpus statistics and domain knowledge (rather than on query logs). It can handle a variety of spelling errors, and can incorporate different levels of spelling reliability among different parts of the corpus. Our experimental study demonstrates the superiority of MaxPaths over existing alternatives in the domain of email search, and indicates its effectiveness beyond that domain.

In future work, we plan to explore how to utilize additional domain knowledge to better estimate the correlation between words. Particularly, from available auxiliary data (Fagin et al., 2010) and tools like *information extraction* (Chiticariu et al., 2010), we can infer and utilize *type* information from the corpus (Li et al., 2006b; Zhu et al., 2007). For instance, if `kohli` is of type person, and `phone` is highly correlated with person instances, then `phone` is highly correlated with `kohli` even if the two words do not frequently co-occur. We also plan to explore aspects of corpus maintenance in dynamic (constantly changing) domains.

References

F. Ahmad and G. Kondrak. 2005. Learning a spelling error model from search query logs. In *HLT/EMNLP*.

R. Bekkerman, A. Mccallum, and G. Huang. 2004. Automatic categorization of email into folders: Benchmark experiments on Enron and Sri Corpora. Technical report, University of Massachusetts - Amherst.

Q. Chen, M. Li, and M. Zhou. 2007. Improving query spelling correction using Web search results. In *EMNLP-CoNLL*, pages 181–189.

L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, and S. Vaithyanathan. 2010. SystemT: An algebraic approach to declarative information extraction. In *ACL*, pages 128–137.

C. Cortes and V. Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

S. Cucerzan and E. Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of Web users. In *EMNLP*, pages 293–300.

D. Eppstein. 1994. Finding the k shortest paths. In *FOCS*, pages 154–165.

R. Fagin, B. Kimelfeld, Y. Li, S. Raghavan, and S. Vaithyanathan. 2010. Understanding queries in a search database system. In *PODS*, pages 273–284.

Google. 2010. A Java API for Google spelling check service. http://code.google.com/p/google-api-spelling-java/.

D. Jurafsky and J. H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR.

M. D. Kernighan, K. W. Church, and W. A. Gale. 1990. A spelling correction program based on a noisy channel model. In *COLING*, pages 205–210.

B. Klimt and Y. Yang. 2004. Introducing the Enron corpus. In *CEAS*.

K. Kukich. 1992. Techniques for automatically correcting words in text. *ACM Comput. Surv.*, 24(4):377–439.

M. Li, M. Zhu, Y. Zhang, and M. Zhou. 2006a. Exploring distributional similarity based models for query spelling correction. In *ACL*.

Y. Li, R. Krishnamurthy, S. Vaithyanathan, and H. V. Jagadish. 2006b. Getting work done on the web: supporting transactional queries. In *SIGIR*, pages 557–564.

R. Mitton. 2010. Fifty years of spellchecking. *Wring Systems Research*, 2:1–7.

J. L. Peterson. 1980. *Computer Programs for Spelling Correction: An Experiment in Program Design*, volume 96 of *Lecture Notes in Computer Science*. Springer.

J. Schaback and F. Li. 2007. Multi-level feature extraction for spelling correction. In *AND*, pages 79–86.

M. Schierle, S. Schulz, and M. Ackermann. 2007. From spelling correction to text cleaning - using context information. In *GfKl*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 397–404.

Seobook. 2010. Keyword typo generator. http://tools.seobook.com/spelling/keywords-typos.cgi.

X. Sun, J. Gao, D. Micol, and C. Quirk. 2010. Learning phrase-based spelling error models from clickthrough data. In *ACL*, pages 266–274.

H. Zhu, S. Raghavan, S. Vaithyanathan, and A. Löser. 2007. Navigating the intranet with high precision. In *WWW*, pages 491–500.

Grammatical Error Correction with Alternating Structure Optimization

Daniel Dahlmeier[1] and **Hwee Tou Ng**[1,2]
[1]NUS Graduate School for Integrative Sciences and Engineering
[2]Department of Computer Science, National University of Singapore
{danielhe,nght}@comp.nus.edu.sg

Abstract

We present a novel approach to grammatical error correction based on Alternating Structure Optimization. As part of our work, we introduce the *NUS Corpus of Learner English (NUCLE)*, a fully annotated one million words corpus of learner English available for research purposes. We conduct an extensive evaluation for article and preposition errors using various feature sets. Our experiments show that our approach outperforms two baselines trained on non-learner text and learner text, respectively. Our approach also outperforms two commercial grammar checking software packages.

1 Introduction

Grammatical error correction (GEC) has been recognized as an interesting as well as commercially attractive problem in natural language processing (NLP), in particular for learners of English as a foreign or second language (EFL/ESL).

Despite the growing interest, research has been hindered by the lack of a large annotated corpus of learner text that is available for research purposes. As a result, the standard approach to GEC has been to train an off-the-shelf classifier to re-predict words in non-learner text. Learning GEC models directly from annotated learner corpora is not well explored, as are methods that combine learner and non-learner text. Furthermore, the evaluation of GEC has been problematic. Previous work has either evaluated on artificial test instances as a substitute for real learner errors or on proprietary data that is not available to other researchers. As a consequence, existing methods have not been compared on the same test set, leaving it unclear where the current state of the art really is.

In this work, we aim to overcome both problems. First, we present a novel approach to GEC based on Alternating Structure Optimization (ASO) (Ando and Zhang, 2005). Our approach is able to train models on annotated learner corpora while still taking advantage of large non-learner corpora. Second, we introduce the *NUS Corpus of Learner English (NUCLE)*, a fully annotated one million words corpus of learner English available for research purposes. We conduct an extensive evaluation for article and preposition errors using six different feature sets proposed in previous work. We compare our proposed ASO method with two baselines trained on non-learner text and learner text, respectively. To the best of our knowledge, this is the first extensive comparison of different feature sets on real learner text which is another contribution of our work. Our experiments show that our proposed ASO algorithm significantly improves over both baselines. It also outperforms two commercial grammar checking software packages in a manual evaluation.

The remainder of this paper is organized as follows. The next section reviews related work. Section 3 describes the tasks. Section 4 formulates GEC as a classification problem. Section 5 extends this to the ASO algorithm. The experiments are presented in Section 6 and the results in Section 7. Section 8 contains a more detailed analysis of the results. Section 9 concludes the paper.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 915–923,
Portland, Oregon, June 19-24, 2011.

2 Related Work

In this section, we give a brief overview on related work on article and preposition errors. For a more comprehensive survey, see (Leacock et al., 2010).

The seminal work on grammatical error correction was done by Knight and Chander (1994) on article errors. Subsequent work has focused on designing better features and testing different classifiers, including memory-based learning (Minnen et al., 2000), decision tree learning (Nagata et al., 2006; Gamon et al., 2008), and logistic regression (Lee, 2004; Han et al., 2006; De Felice, 2008). Work on preposition errors has used a similar classification approach and mainly differs in terms of the features employed (Chodorow et al., 2007; Gamon et al., 2008; Lee and Knutsson, 2008; Tetreault and Chodorow, 2008; Tetreault et al., 2010; De Felice, 2008). All of the above works only use non-learner text for training.

Recent work has shown that training on annotated learner text can give better performance (Han et al., 2010) and that the observed word used by the writer is an important feature (Rozovskaya and Roth, 2010b). However, training data has either been small (Izumi et al., 2003), only partly annotated (Han et al., 2010), or artificially created (Rozovskaya and Roth, 2010b; Rozovskaya and Roth, 2010a).

Almost no work has investigated ways to combine learner and non-learner text for training. The only exception is Gamon (2010), who combined features from the output of logistic-regression classifiers and language models trained on non-learner text in a meta-classifier trained on learner text. In this work, we show a more direct way to combine learner and non-learner text in a single model.

Finally, researchers have investigated GEC in connection with web-based models in NLP (Lapata and Keller, 2005; Bergsma et al., 2009; Yi et al., 2008). These methods do not use classifiers, but rely on simple n-gram counts or page hits from the Web.

3 Task Description

In this work, we focus on article and preposition errors, as they are among the most frequent types of errors made by EFL learners.

3.1 Selection vs. Correction Task

There is an important difference between training on annotated learner text and training on non-learner text, namely whether the observed word can be used as a feature or not. When training on non-learner text, the observed word cannot be used as a feature. The word choice of the writer is "blanked out" from the text and serves as the correct class. A classifier is trained to re-predict the word given the surrounding context. The *confusion set* of possible classes is usually pre-defined. This *selection task* formulation is convenient as training examples can be created "for free" from any text that is assumed to be free of grammatical errors. We define the more realistic *correction task* as follows: given a particular word and its context, propose an appropriate correction. The proposed correction can be identical to the observed word, i.e., no correction is necessary. The main difference is that the word choice of the writer can be encoded as part of the features.

3.2 Article Errors

For article errors, the classes are the three articles *a*, *the*, and the *zero-article*. This covers article insertion, deletion, and substitution errors. During training, each noun phrase (NP) in the training data is one training example. When training on learner text, the correct class is the article provided by the human annotator. When training on non-learner text, the correct class is the observed article. The context is encoded via a set of feature functions. During testing, each NP in the test set is one test example. The correct class is the article provided by the human annotator when testing on learner text or the observed article when testing on non-learner text.

3.3 Preposition Errors

The approach to preposition errors is similar to articles but typically focuses on preposition substitution errors. In our work, the classes are 36 frequent English prepositions (*about, along, among, around, as, at, beside, besides, between, by, down, during, except, for, from, in, inside, into, of, off, on, onto, outside, over, through, to, toward, towards, under, underneath, until, up, upon, with, within, without*), which we adopt from previous work. Every prepositional phrase (PP) that is governed by one of the

36 prepositions is one training or test example. We ignore PPs governed by other prepositions.

4 Linear Classifiers for Grammatical Error Correction

In this section, we formulate GEC as a classification problem and describe the feature sets for each task.

4.1 Linear Classifiers

We use classifiers to approximate the unknown relation between articles or prepositions and their contexts in learner text, and their valid corrections. The articles or prepositions and their contexts are represented as feature vectors $\mathbf{X} \in \mathcal{X}$. The corrections are the classes $Y \in \mathcal{Y}$.

In this work, we employ binary linear classifiers of the form $\mathbf{u}^T\mathbf{X}$ where $\mathbf{u}$ is a weight vector. The outcome is considered $+1$ if the score is positive and -1 otherwise. A popular method for finding $\mathbf{u}$ is *empirical risk minimization* with *least square regularization*. Given a training set $\{\mathbf{X}_i, Y_i\}_{i=1,\ldots,n}$, we aim to find the weight vector that minimizes the empirical loss on the training data

$$\hat{\mathbf{u}} = \arg\min_{\mathbf{u}} \left(\frac{1}{n} \sum_{i=1}^{n} L(\mathbf{u}^T\mathbf{X}_i, Y_i) + \lambda \, ||\mathbf{u}||^2 \right), \quad (1)$$

where L is a loss function. We use a modification of Huber's robust loss function. We fix the regularization parameter λ to 10^{-4}. A multi-class classification problem with m classes can be cast as m binary classification problems in a *one-vs-rest* arrangement. The prediction of the classifier is the class with the highest score $\hat{Y} = \arg\max_{Y \in \mathcal{Y}} (\mathbf{u}_Y^T\mathbf{X})$. In earlier experiments, this linear classifier gave comparable or superior performance compared to a logistic regression classifier.

4.2 Features

We re-implement six feature extraction methods from previous work, three for articles and three for prepositions. The methods require different linguistic pre-processing: chunking, CCG parsing, and constituency parsing.

4.2.1 Article Errors

- **DeFelice** The system in (De Felice, 2008) for article errors uses a CCG parser to extract a rich set of syntactic and semantic features, including part of speech (POS) tags, hypernyms from WordNet (Fellbaum, 1998), and named entities.
- **Han** The system in (Han et al., 2006) relies on shallow syntactic and lexical features derived from a chunker, including the words before, in, and after the NP, the head word, and POS tags.
- **Lee** The system in (Lee, 2004) uses a constituency parser. The features include POS tags, surrounding words, the head word, and hypernyms from WordNet.

4.2.2 Preposition Errors

- **DeFelice** The system in (De Felice, 2008) for preposition errors uses a similar rich set of syntactic and semantic features as the system for article errors. In our re-implementation, we do not use a subcategorization dictionary, as this resource was not available to us.
- **TetreaultChunk** The system in (Tetreault and Chodorow, 2008) uses a chunker to extract features from a two-word window around the preposition, including lexical and POS n-grams, and the head words from neighboring constituents.
- **TetreaultParse** The system in (Tetreault et al., 2010) extends (Tetreault and Chodorow, 2008) by adding additional features derived from a constituency and a dependency parse tree.

For each of the above feature sets, we add the observed article or preposition as an additional feature when training on learner text.

5 Alternating Structure Optimization

This section describes the ASO algorithm and shows how it can be used for grammatical error correction.

5.1 The ASO algorithm

Alternating Structure Optimization (Ando and Zhang, 2005) is a multi-task learning algorithm that takes advantage of the *common structure* of multiple related problems. Let us assume that we have m binary classification problems. Each classifier $\mathbf{u}_i$ is a

weight vector of dimension p. Let Θ be an orthonormal $h \times p$ matrix that captures the common structure of the m weight vectors. We assume that each weight vector can be decomposed into two parts: one part that models the particular i-th classification problem and one part that models the common structure

$$\mathbf{u}_i = \mathbf{w}_i + \Theta^T \mathbf{v}_i. \quad (2)$$

The parameters $[\{\mathbf{w}_i, \mathbf{v}_i\}, \Theta]$ can be learned by *joint empirical risk minimization*, i.e., by minimizing the joint empirical loss of the m problems on the training data

$$\sum_{l=1}^{m} \left(\frac{1}{n} \sum_{i=1}^{n} L\left(\left(\mathbf{w}_l + \Theta^T \mathbf{v}_l\right)^T \mathbf{X}_i^l, Y_i^l \right) + \lambda \left|\left|\mathbf{w}_l\right|\right|^2 \right). \quad (3)$$

The key observation in ASO is that the problems used to find Θ do not have to be same as the *target problems* that we ultimately want to solve. Instead, we can automatically create *auxiliary problems* for the sole purpose of learning a better Θ.

Let us assume that we have k target problems and m auxiliary problems. We can obtain an approximate solution to Equation 3 by performing the following algorithm (Ando and Zhang, 2005):

1. Learn m linear classifiers $\mathbf{u}_i$ independently.
2. Let $U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m]$ be the $p \times m$ matrix formed from the m weight vectors.
3. Perform Singular Value Decomposition (SVD) on U: $U = V_1 D V_2^T$. The first h column vectors of V_1 are stored as rows of Θ.
4. Learn $\mathbf{w}_j$ and $\mathbf{v}_j$ for each of the target problems by minimizing the empirical risk:
$$\frac{1}{n} \sum_{i=1}^{n} L\left(\left(\mathbf{w}_j + \Theta^T \mathbf{v}_j\right)^T \mathbf{X}_i, Y_i \right) + \lambda \left|\left|\mathbf{w}_j\right|\right|^2.$$
5. The weight vector for the j-th target problem is:
$$\mathbf{u}_j = \mathbf{w}_j + \Theta^T \mathbf{v}_j.$$

5.2 ASO for Grammatical Error Correction

The key observation in our work is that the selection task on non-learner text is a highly informative *auxiliary problem* for the correction task on learner text. For example, a classifier that can predict the presence or absence of the preposition *on* can be helpful for correcting wrong uses of *on* in learner text, e.g., if the classifier's confidence for *on* is low but the writer used the preposition *on*, the writer might have made a mistake. As the auxiliary problems can be created automatically, we can leverage the power of very large corpora of non-learner text.

Let us assume a grammatical error correction task with m classes. For each class, we define a binary auxiliary problem. The feature space of the auxiliary problems is a restriction of the original feature space $\mathcal{X}$ to all features except the observed word: $\mathcal{X} \backslash \{X_{obs}\}$. The weight vectors of the auxiliary problems form the matrix U in Step 2 of the ASO algorithm from which we obtain Θ through SVD. Given Θ, we learn the vectors $\mathbf{w}_j$ and $\mathbf{v}_j$, $j = 1, \ldots, k$ from the annotated learner text using the complete feature space $\mathcal{X}$.

This can be seen as an instance of *transfer learning* (Pan and Yang, 2010), as the auxiliary problems are trained on data from a different domain (non-learner text) and have a slightly different feature space ($\mathcal{X} \backslash \{X_{obs}\}$). We note that our method is general and can be applied to any classification problem in GEC.

6 Experiments

6.1 Data Sets

The main corpus in our experiments is the *NUS Corpus of Learner English (NUCLE)*. The corpus consists of about 1,400 essays written by EFL/ESL university students on a wide range of topics, like environmental pollution or healthcare. It contains over one million words which are completely annotated with error tags and corrections. All annotations have been performed by professional English instructors. We use about 80% of the essays for training, 10% for development, and 10% for testing. We ensure that no sentences from the same essay appear in both the training and the test or development data. NUCLE is available to the community for research purposes.

On average, only 1.8% of the articles and 1.3% of the prepositions in NUCLE contain an error. This figure is considerably lower compared to other learner corpora (Leacock et al., 2010, Ch. 3) and shows that our writers have a relatively high proficiency of English. We argue that this makes the task considerably more difficult. Furthermore, to keep the task as realistic as possible, we do not filter the

test data in any way.

In addition to NUCLE, we use a subset of the New York Times section of the Gigaword corpus[1] and the Wall Street Journal section of the Penn Treebank (Marcus et al., 1993) for some experiments. We pre-process all corpora using the following tools: We use NLTK[2] for sentence splitting, OpenNLP[3] for POS tagging, YamCha (Kudo and Matsumoto, 2003) for chunking, the C&C tools (Clark and Curran, 2007) for CCG parsing and named entity recognition, and the Stanford parser (Klein and Manning, 2003a; Klein and Manning, 2003b) for constituency and dependency parsing.

6.2 Evaluation Metrics

For experiments on non-learner text, we report accuracy, which is defined as the number of correct predictions divided by the total number of test instances. For experiments on learner text, we report F_1-measure

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

where precision is the number of suggested corrections that agree with the human annotator divided by the total number of proposed corrections by the system, and recall is the number of suggested corrections that agree with the human annotator divided by the total number of errors annotated by the human annotator.

6.3 Selection Task Experiments on WSJ Test Data

The first set of experiments investigates predicting articles and prepositions in non-learner text. This primarily serves as a reference point for the correction task described in the next section. We train classifiers as described in Section 4 on the Gigaword corpus. We train with up to 10 million training instances, which corresponds to about 37 million words of text for articles and 112 million words of text for prepositions. The test instances are extracted from section 23 of the WSJ and no text from the WSJ is included in the training data. The observed article or preposition choice of the writer is the class we want to predict. Therefore, the article or preposition cannot be part of the input features. Our proposed ASO method is not included in these experiments, as it uses the observed article or preposition as a feature which is only applicable when testing on learner text.

6.4 Correction Task Experiments on NUCLE Test Data

The second set of experiments investigates the primary goal of this work: to automatically correct grammatical errors in learner text. The test instances are extracted from NUCLE. In contrast to the previous selection task, the observed word choice of the writer can be different from the correct class and the observed word is available during testing. We investigate two different baselines and our ASO method.

The first baseline is a classifier trained on the Gigaword corpus in the same way as described in the selection task experiment. We use a simple thresholding strategy to make use of the observed word during testing. The system only flags an error if the difference between the classifier's confidence for its first choice and the confidence for the observed word is higher than a threshold t. The threshold parameter t is tuned on the NUCLE development data for each feature set. In our experiments, the value for t is between 0.7 and 1.2.

The second baseline is a classifier trained on NUCLE. The classifier is trained in the same way as the Gigaword model, except that the observed word choice of the writer is included as a feature. The correct class during training is the correction provided by the human annotator. As the observed word is part of the features, this model does not need an extra thresholding step. Indeed, we found that thresholding is harmful in this case. During training, the instances that do not contain an error greatly outnumber the instances that do contain an error. To reduce this imbalance, we keep all instances that contain an error and retain a random sample of q percent of the instances that do not contain an error. The undersample parameter q is tuned on the NUCLE development data for each data set. In our experiments, the value for q is between 20% and 40%.

Our ASO method is trained in the following way. We create binary auxiliary problems for articles or prepositions, i.e., there are 3 auxiliary problems for

[1] LDC2009T13

[2] www.nltk.org

[3] opennlp.sourceforge.net

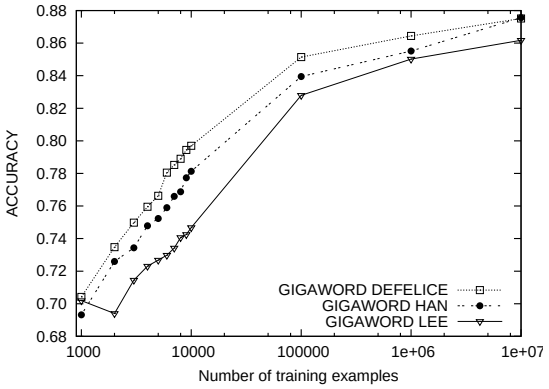
articles and 36 auxiliary problems for prepositions. We train the classifiers for the auxiliary problems on the complete 10 million instances from Gigaword in the same ways as in the selection task experiment. The weight vectors of the auxiliary problems form the matrix U. We perform SVD to get $U = V_1 D V_2^T$. We keep all columns of V_1 to form Θ. The target problems are again binary classification problems for each article or preposition, but this time trained on NUCLE. The observed word choice of the writer is included as a feature for the target problems. We again undersample the instances that do not contain an error and tune the parameter q on the NUCLE development data. The value for q is between 20% and 40%. No thresholding is applied.

We also experimented with a classifier that is trained on the concatenated data from NUCLE and Gigaword. This model always performed worse than the better of the individual baselines. The reason is that the two data sets have different feature spaces which prevents simple concatenation of the training data. We therefore omit these results from the paper.

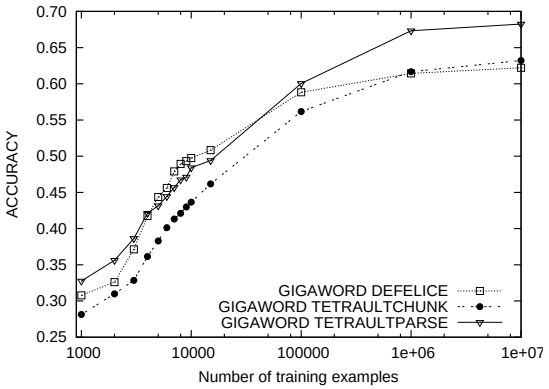
7 Results

The learning curves of the selection task experiments on WSJ test data are shown in Figure 1. The three curves in each plot correspond to different feature sets. Accuracy improves quickly in the beginning but improvements get smaller as the size of the training data increases. The best results are 87.56% for articles (Han) and 68.25% for prepositions (TetreaultParse). The best accuracy for articles is comparable to the best reported results of 87.70% (Lee, 2004) on this data set.

The learning curves of the correction task experiments on NUCLE test data are shown in Figure 2 and 3. Each sub-plot shows the curves of three models as described in the last section: ASO trained on NUCLE and Gigaword, the baseline classifier trained on NUCLE, and the baseline classifier trained on Gigaword. For ASO, the x-axis shows the number of target problem training instances. The first observation is that high accuracy for the selection task on non-learner text does not automatically entail high F_1-measure on learner text. We also note that feature sets with similar performance on non-learner text can show very different performance on



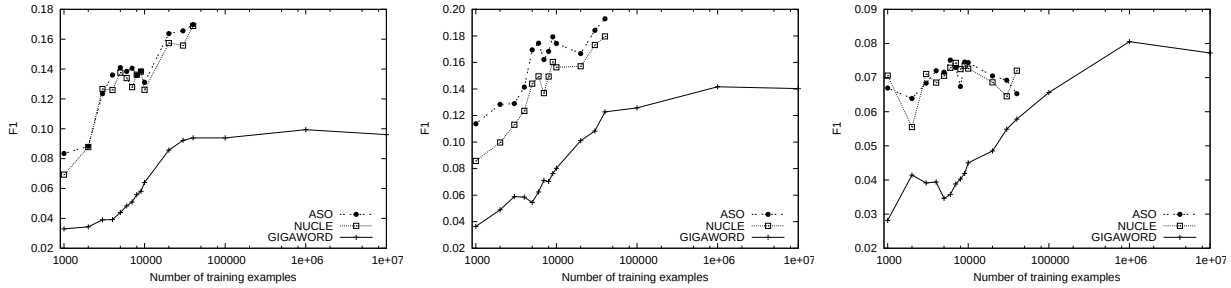
(a) Articles



(b) Prepositions

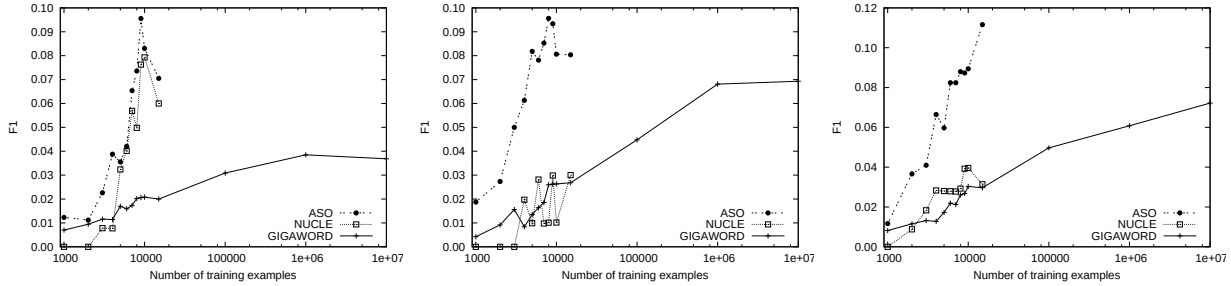
Figure 1: Accuracy for the selection task on WSJ test data.

learner text. The second observation is that training on annotated learner text can significantly improve performance. In three experiments (articles DeFelice, Han, prepositions DeFelice), the NUCLE model outperforms the Gigaword model trained on 10 million instances. Finally, the ASO models show the best results. In the experiments where the NUCLE models already perform better than the Gigaword baseline, ASO gives comparable or slightly better results (articles DeFelice, Han, Lee, prepositions DeFelice). In those experiments where neither baseline shows good performance (TetreaultChunk, TetreaultParse), ASO results in a large improvement over either baseline. The best results are 19.29% F_1-measure for articles (Han) and 11.15% F_1-measure for prepositions (TetreaultParse) achieved by the ASO model.



(a) DeFelice (b) Han (c) Lee

Figure 2: F_1-measure for the article correction task on NUCLE test data. Each plot shows ASO and two baselines for a particular feature set.



(a) DeFelice (b) TetreaultChunk (c) TetreaultParse

Figure 3: F_1-measure for the preposition correction task on NUCLE test data. Each plot shows ASO and two baselines for a particular feature set.

8 Analysis

In this section, we analyze the results in more detail and show examples from our test set for illustration.

Table 1 shows precision, recall, and F_1-measure for the best models in our experiments. ASO achieves a higher F_1-measure than either baseline. We use the sign-test with bootstrap re-sampling for statistical significance testing. The sign-test is a non-parametric test that makes fewer assumptions than parametric tests like the t-test. The improvements in F_1-measure of ASO over either baseline are statistically significant ($p < 0.001$) for both articles and prepositions.

The difficulty in GEC is that in many cases, more than one word choice can be correct. Even with a threshold, the Gigaword baseline model suggests too many corrections, because the model cannot make use of the observed word as a feature. This results in low precision. For example, the model replaces **as** with **by** in the sentence *"This group should be categorized **as the vulnerable group**"*, which is wrong.

Articles			
Model	**Prec**	**Rec**	**F_1**
Gigaword (Han)	10.33	**21.81**	14.02
NUCLE (Han)	**29.48**	12.91	17.96
ASO (Han)	26.44	15.18	**19.29**
Prepositions			
Model	**Prec**	**Rec**	**F_1**
Gigaword (TetreaultParse)	4.77	**14.81**	7.21
NUCLE (DeFelice)	13.84	5.55	7.92
ASO (TetreaultParse)	**18.30**	8.02	**11.15**

Table 1: Best results for the correction task on NUCLE test data. Improvements for ASO over either baseline are statistically significant ($p < 0.001$) for both tasks.

In contrast, the NUCLE model learns a bias towards the observed word and therefore achieves higher precision. However, the training data is

smaller and therefore recall is low as the model has not seen enough examples during training. This is especially true for prepositions which can occur in a large variety of contexts. For example, the preposition **in** should be **on** in the sentence *"... psychology had an impact **in** the way we process and manage technology"*. The phrase *"impact on the way"* does not appear in the NUCLE training data and the NUCLE baseline fails to detect the error.

The ASO model is able to take advantage of both the annotated learner text and the large non-learner text, thus achieving overall high F_1-measure. The phrase *"impact on the way"*, for example, appears many times in the Gigaword training data. With the common structure learned from the auxiliary problems, the ASO model successfully finds and corrects this mistake.

8.1 Manual Evaluation

We carried out a manual evaluation of the best ASO models and compared their output with two commercial grammar checking software packages which we call *System A* and *System B*. We randomly sampled 1000 test instances for articles and 2000 test instances for prepositions and manually categorized each test instance into one of the following categories: *(1) Correct* means that both human and system flag an error and suggest the same correction. If the system's correction differs from the human but is equally acceptable, it is considered *(2) Both Ok*. If the system identifies an error but fails to correct it, we consider it *(3) Both Wrong*, as both the writer and the system are wrong. *(4) Other Error* means that the system's correction does not result in a grammatical sentence because of another grammatical error that is outside the scope of article or preposition errors, e.g., a noun number error as in *"all **the** dog"*. If the system corrupts a previously correct sentence it is a *(5) False Flag*. If the human flags an error but the system does not, it is a *(6) Miss*. *(7) No Flag* means that neither the human annotator nor the system flags an error. We calculate precision by dividing the count of category (1) by the sum of counts of categories (1), (3), and (5), and recall by dividing the count of category (1) by the sum of counts of categories (1), (3), and (6). The results are shown in Table 2. Our ASO method outperforms both commercial software packages. Our evaluation shows that even commercial software packages achieve low F_1-measure for article and preposition errors, which confirms the difficulty of these tasks.

Articles			
	ASO	**System A**	**System B**
(1) Correct	4	1	1
(2) Both Ok	16	12	18
(3) Both Wrong	0	1	0
(4) Other Error	1	0	0
(5) False Flag	1	0	4
(6) Miss	3	5	6
(7) No Flag	975	981	971
Precision	**80.00**	50.00	20.00
Recall	**57.14**	14.28	14.28
F_1	**66.67**	22.21	16.67
Prepositions			
	ASO	**System A**	**System B**
(1) Correct	3	3	0
(2) Both Ok	35	39	24
(3) Both Wrong	0	2	0
(4) Other Error	0	0	0
(5) False Flag	5	11	1
(6) Miss	12	11	15
(7) No Flag	1945	1934	1960
Precision	**37.50**	18.75	0.00
Recall	**20.00**	18.75	0.00
F_1	**26.09**	18.75	0.00

Table 2: Manual evaluation and comparison with commercial grammar checking software.

9 Conclusion

We have presented a novel approach to grammatical error correction based on Alternating Structure Optimization. We have introduced the NUS Corpus of Learner English (NUCLE), a fully annotated corpus of learner text. Our experiments for article and preposition errors show the advantage of our ASO approach over two baseline methods. Our ASO approach also outperforms two commercial grammar checking software packages in a manual evaluation.

Acknowledgments

This research was done for CSIDM Project No. CSIDM-200804 partially funded by a grant from the National Research Foundation (NRF) administered by the Media Development Authority (MDA) of Singapore.

References

R.K. Ando and T. Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6.

S. Bergsma, D. Lin, and R. Goebel. 2009. Web-scale n-gram models for lexical disambiguation. In *Proceedings of IJCAI*.

M. Chodorow, J. Tetreault, and N.R. Han. 2007. Detection of grammatical errors involving prepositions. In *Proceedings of the 4th ACL-SIGSEM Workshop on Prepositions*.

S. Clark and J.R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4).

R. De Felice. 2008. *Automatic Error Detection in Non-native English*. Ph.D. thesis, University of Oxford.

C. Fellbaum, editor. 1998. *WordNet: An electronic lexical database*. MIT Press, Cambridge,MA.

M. Gamon, J. Gao, C. Brockett, A. Klementiev, W.B. Dolan, D. Belenko, and L. Vanderwende. 2008. Using contextual speller techniques and language modeling for ESL error correction. In *Proceedings of IJCNLP*.

M. Gamon. 2010. Using mostly native data to correct errors in learners' writing: A meta-classifier approach. In *Proceedings of HLT-NAACL*.

N.R. Han, M. Chodorow, and C. Leacock. 2006. Detecting errors in English article usage by non-native speakers. *Natural Language Engineering*, 12(02).

N.R. Han, J. Tetreault, S.H. Lee, and J.Y. Ha. 2010. Using an error-annotated learner corpus to develop an ESL/EFL error correction system. In *Proceedings of LREC*.

E. Izumi, K. Uchimoto, T. Saiga, T. Supnithi, and H. Isahara. 2003. Automatic error detection in the Japanese learners' English spoken data. In *Companion Volume to the Proceedings of ACL*.

D. Klein and C.D. Manning. 2003a. Accurate unlexicalized parsing. In *Proceedings of ACL*.

D. Klein and C.D. Manning. 2003b. Fast exact inference with a factored model for natural language processing. *Advances in Neural Information Processing Systems (NIPS 2002)*, 15.

K. Knight and I. Chander. 1994. Automated postediting of documents. In *Proceedings of AAAI*.

T Kudo and Y. Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proceedings of ACL*.

M. Lapata and F. Keller. 2005. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*, 2(1).

C. Leacock, M. Chodorow, M. Gamon, and J. Tetreault. 2010. *Automated Grammatical Error Detection for Language Learners*. Morgan & Claypool Publishers, San Rafael,CA.

J. Lee and O. Knutsson. 2008. The role of PP attachment in preposition generation. In *Proceedings of CICLing*.

J. Lee. 2004. Automatic article restoration. In *Proceedings of HLT-NAACL*.

M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19.

G. Minnen, F. Bond, and A. Copestake. 2000. Memory-based learning for article generation. In *Proceedings of CoNLL*.

R. Nagata, A. Kawai, K. Morihiro, and N. Isu. 2006. A feedback-augmented method for detecting errors in the writing of learners of English. In *Proceedings of COLING-ACL*.

S.J. Pan and Q. Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10).

A. Rozovskaya and D. Roth. 2010a. Generating confusion sets for context-sensitive error correction. In *Proceedings of EMNLP*.

A. Rozovskaya and D. Roth. 2010b. Training paradigms for correcting errors in grammar and usage. In *Proceedings of HLT-NAACL*.

J. Tetreault and M. Chodorow. 2008. The ups and downs of preposition error detection in ESL writing. In *Proceedings of COLING*.

J. Tetreault, J. Foster, and M. Chodorow. 2010. Using parse features for preposition selection and error detection. In *Proceedings of ACL*.

X. Yi, J. Gao, and W.B. Dolan. 2008. A web-based English proofing system for English as a second language users. In *Proceedings of IJCNLP*.

Algorithm Selection and Model Adaptation for ESL Correction Tasks

Alla Rozovskaya and Dan Roth
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{rozovska,danr}@illinois.edu

Abstract

We consider the problem of correcting errors made by English as a Second Language (ESL) writers and address two issues that are essential to making progress in ESL error correction - algorithm selection and model adaptation to the first language of the ESL learner.

A variety of learning algorithms have been applied to correct ESL mistakes, but often comparisons were made between incomparable data sets. We conduct an extensive, fair comparison of four popular learning methods for the task, reversing conclusions from earlier evaluations. Our results hold for different training sets, genres, and feature sets.

A second key issue in ESL error correction is the adaptation of a model to the first language of the writer. Errors made by non-native speakers exhibit certain regularities and, as we show, models perform much better when they use knowledge about error patterns of the non-native writers. We propose a novel way to adapt a learned algorithm to the first language of the writer that is both cheaper to implement and performs better than other adaptation methods.

1 Introduction

There has been a lot of recent work on correcting writing mistakes made by English as a Second Language (ESL) learners (Izumi et al., 2003; Eeg-Olofsson and Knuttson, 2003; Han et al., 2006; Felice and Pulman, 2008; Gamon et al., 2008; Tetreault and Chodorow, 2008; Elghaari et al., 2010; Tetreault et al., 2010; Gamon, 2010; Rozovskaya and Roth, 2010c). Most of this work has focused on correcting mistakes in article and preposition usage, which are some of the most common error types among non-native writers of English (Dalgish, 1985; Bitchener et al., 2005; Leacock et al., 2010). Examples below illustrate some of these errors:

1. "They listen to *None*/the* lecture carefully."
2. "He is an engineer with a passion *to*/for* what he does."

In (1) the definite article is incorrectly omitted. In (2), the writer uses an incorrect preposition.

Approaches to correcting preposition and article mistakes have adopted the methods of the *context-sensitive spelling correction task*, which addresses the problem of correcting spelling mistakes that result in legitimate words, such as confusing *their* and *there* (Carlson et al., 2001; Golding and Roth, 1999). A *candidate set* or a *confusion set* is defined that specifies a list of confusable words, e.g., {*their, there*}. Each occurrence of a confusable word in text is represented as a vector of features derived from a context window around the target, e.g., words and part-of-speech tags. A classifier is trained on text assumed to be error-free. At decision time, for each word in text, e.g. *there*, the classifier predicts the most likely candidate from the corresponding confusion set {*their, there*}.

Models for correcting article and preposition errors are similarly trained on error-free native English text, where the confusion set includes all articles or prepositions (Izumi et al., 2003; Eeg-Olofsson and Knuttson, 2003; Han et al., 2006; Felice and Pulman, 2008; Gamon et al., 2008; Tetreault and Chodorow, 2008; Tetreault et al., 2010).

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 924–933,
Portland, Oregon, June 19-24, 2011.

Although the choice of a particular learning algorithm differs, with the exception of decision trees (Gamon et al., 2008), all algorithms used are linear learning algorithms, some discriminative (Han et al., 2006; Felice and Pulman, 2008; Tetreault and Chodorow, 2008; Rozovskaya and Roth, 2010c; Rozovskaya and Roth, 2010b), some probabilistic (Gamon et al., 2008; Gamon, 2010), or "counting" (Bergsma et al., 2009; Elghaari et al., 2010).

While model comparison has not been the goal of the earlier studies, it is quite common to compare systems, even when they are trained on different data sets and use different features. Furthermore, since there is no shared ESL data set, systems are also evaluated on data from different ESL sources or even on native data. Several conclusions have been made when comparing systems developed for ESL correction tasks. A *language model* was found to outperform a *maximum entropy classifier* (Gamon, 2010). However, the language model was trained on the Gigaword corpus, $17 \cdot 10^9$ words (Linguistic Data Consortium, 2003), a corpus several orders of magnitude larger than the corpus used to train the classifier. Similarly, web-based models built on Google Web1T 5-gram Corpus (Bergsma et al., 2009) achieve better results when compared to a maximum entropy model that uses a corpus $10,000$ times smaller (Chodorow et al., 2007)[1].

In this work, we compare four popular learning methods applied to the problem of correcting preposition and article errors and evaluate on a common ESL data set. We compare two probabilistic approaches – *Naïve Bayes* and *language modeling*; a discriminative algorithm *Averaged Perceptron*; and a count-based method *SumLM* (Bergsma et al., 2009), which, as we show, is very similar to Naïve Bayes, but with a different free coefficient. We train our models on data from several sources, varying training sizes and feature sets, and show that there are significant differences in the performance of these algorithms. Contrary to previous results (Bergsma et al., 2009; Gamon, 2010), we find that when trained on the same data with the same features, Averaged Perceptron achieves the best performance, followed by Naïve Bayes, then the language model, and finally the count-based approach. Our results hold for training sets of different sizes, genres, and feature sets. We also explain the performance differences from the perspective of each algorithm.

The second important question that we address is that of adapting the decision to the source language of the writer. Errors made by non-native speakers exhibit certain regularities. Adapting a model so that it takes into consideration the specific error patterns of the non-native writers was shown to be extremely helpful in the context of discriminative classifiers (Rozovskaya and Roth, 2010c; Rozovskaya and Roth, 2010b). However, this method requires generating new training data and training a separate classifier for each source language. Our key contribution here is a novel, simple, and elegant adaptation method within the framework of the Naïve Bayes algorithm, which yields even greater performance gains. Specifically, we show how the error patterns of the non-native writers can be viewed as *a different distribution on candidate priors* in the confusion set. Following this observation, we train Naïve Bayes in a traditional way, regardless of the source language of the writer, and then, only at decision time, change the prior probabilities of the model from the ones observed in the native training data to the ones corresponding to error patterns in the non-native writer's source language (Section 4). A related idea has been applied in Word Sense Disambiguation to adjust the model priors to a new domain with different sense distributions (Chan and Ng, 2005).

The paper has two main contributions. First, we conduct a fair comparison of four learning algorithms and show that the discriminative approach Averaged Perceptron is the best performing model (Sec. 3). Our results do not support earlier conclusions with respect to the performance of count-based models (Bergsma et al., 2009) and language models (Gamon, 2010). In fact, we show that SumLM is comparable to Averaged Perceptron trained with a *10 times* smaller corpus, and language model is comparable to Averaged Perceptron trained with a *2 times* smaller corpus.

The second, and most significant, of our contributions is a novel way to adapt a model to the source language of the writer, *without re-training the model* (Sec. 4). As we show, adapting to the source language of the writer provides significant performance improvement, and our new method also performs

[1] These two models also use different features.

better than previous, more complicated methods.

Section 2 presents the theoretical component of the linear learning framework. In Section 3, we describe the experiments, which compare the four learning models. Section 4 presents the key result of this work, a novel method of adapting the model to the source language of the learner.

2 The Models

The standard approach to preposition correction is to cast the problem as a multi-class classification task and train a classifier on features defined on the surrounding context[2]. The model selects the most likely candidate from the confusion set, where the set of candidates includes the top n most frequent English prepositions. Our confusion set includes the top ten prepositions[3]: $ConfSet = \{$*on, from, for, of, about, to, at, in, with, by*$\}$. We use p to refer to a candidate preposition from $ConfSet$.

Let *preposition context* denote the preposition and the window around it. For instance, "a passion *to* what he" is a context for window size 2. We use three feature sets, varying window size from 2 to 4 words on each side (see Table 1). All feature sets consist of word n-grams of various lengths spanning p and all the features are of the form $s^{-k}ps^{+m}$, where s^{-k} and s^{+m} denote k words before and m words after p; we show two 3-gram features for illustration:

1. a passion *p*
2. passion *p* what

We implement four linear learning models: the discriminative method *Averaged Perceptron* (AP); two probabilistic methods – *a language model* (LM) and *Naïve Bayes* (NB); and a "counting" method *SumLM* (Bergsma et al., 2009).

Each model produces a score for a candidate in the confusion set. Since all of the models are linear, the hypotheses generated by the algorithms differ only in the weights they assign to the features

[2]We also report one experiment on the article correction task. We take the preposition correction task as an example; the article case is treated in the same way.

[3]This set of prepositions is also considered in other works, e.g. (Rozovskaya and Roth, 2010b). The usage of the ten most frequent prepositions accounts for 82% of all preposition errors (Leacock et al., 2010).

Feature set	Preposition context	N-gram lengths
Win2	a passion [to] what he	2,3,4
Win3	with a passion [to] what he does	2,3,4
Win4	engineer with a passion [to] what he does .	2,3,4,5

Table 1: **Description of the three feature sets** used in the experiments. All feature sets consist of word n-grams of various lengths spanning the preposition and vary by n-gram length and window size.

Method	Free Coefficient	Feature weights
AP	bias parameter	mistake-driven
LM	$\lambda \cdot prior(p)$	$\sum_{v_l \circ v_r} \lambda_{v_r} \cdot log(P(u\|v_r))$
NB	$log(prior(p))$	$log(P(f\|p))$
SumLM	$\|F(S,p)\| \cdot log(C(p))$	$log(P(f\|p))$

Table 2: **Summary of the learning methods**. $C(p)$ denotes the number of times preposition p occurred in training. λ is a smoothing parameter, u is the rightmost word in f, $v_l \circ v_r$ denotes all concatenations of substrings v_l and v_r of feature f without u.

(Roth, 1998; Roth, 1999). Thus a score computed by each of the models for a preposition p in the context S can be expressed as follows:

$$g(S,p) = C(p) + \sum_{f \in F(S,p)} w_a(f), \qquad (1)$$

where $F(S,p)$ is the set of features active in context S relative to preposition p, $w_a(f)$ is the weight algorithm a assigns to feature $f \in F$, and $C(p)$ is a free coefficient. Predictions are made using the winner-take-all approach: $argmax_p g(S,p)$. The algorithms make use of the same feature set F and differ only by how the weights $w_a(f)$ and $C(p)$ are computed. Below we explain how the weights are determined in each method. Table 2 summarizes the four approaches.

2.1 Averaged Perceptron

Discriminative classifiers represent the most common learning paradigm in error correction. AP (Freund and Schapire, 1999) is a discriminative mistake-driven online learning algorithm. It maintains a vector of feature weights w and processes one training example at a time, updating w if the current weight assignment makes a mistake on the training example. In the case of AP, the $C(p)$ coefficient refers to the bias parameter (see Table 2).

We use the regularized version of AP in Learning Based Java[4] (LBJ, (Rizzolo and Roth, 2007)). While classical Perceptron comes with a generalization bound related to the margin of the data, Averaged Perceptron also comes with a PAC-like generalization bound (Freund and Schapire, 1999). This linear learning algorithm is known, both theoretically and experimentally, to be among the best linear learning approaches and is competitive with SVM and Logistic Regression, while being more efficient in training. It also has been shown to produce state-of-the-art results on many natural language applications (Punyakanok et al., 2008).

2.2 Language Modeling

Given a feature $f = s^{-k}ps^{+m}$, let u denote the rightmost word in f and $v_l \circ v_r$ denote all concatenations of substrings v_l and v_r of feature f without u. The language model computes several probabilities of the form $P(u|v_r)$. If f ="with a passion *p* what", then u ="what", and $v_r \in$ {"with a passion *p*", "a passion *p*", "passion *p*", "*p*" }. In practice, these probabilities are smoothed and replaced with their corresponding log values, and the total weight contribution of f to the scoring function of p is $\sum_{v_l \circ v_r} \lambda_{v_r} \cdot log(P(u|v_r))$. In addition, this scoring function has a coefficient that only depends on p: $C(p) = \lambda \cdot prior(p)$ (see Table 2). The *prior* probability of a candidate p is:

$$prior(p) = \frac{C(p)}{\sum_{q \in ConfSet} C(q)}, \qquad (2)$$

where $C(p)$ and $C(q)$ denote the number of times preposition p and q, respectively, occurred in the training data. We implement a count-based LM with Jelinek-Mercer linear interpolation as a smoothing method[5] (Chen and Goodman, 1996), where each n-gram length, from 1 to n, is associated with an interpolation smoothing weight λ. Weights are optimized on a held-out set of ESL sentences.

Win2 and Win3 features correspond to 4-gram LMs and Win4 to 5-gram LMs. Language models are trained with SRILM (Stolcke, 2002).

[4]LBJ can be downloaded from `http://cogcomp.cs.illinois.edu`.

[5]Unlike other LM methods, this approach allows us to train LMs on very large data sets. Although we found that backoff LMs may perform slightly better, they still maintain the same hierarchy in the order of algorithm performance.

2.3 Naïve Bayes

NB is another linear model, which is often hard to beat using more sophisticated approaches. NB architecture is also particularly well-suited for adapting the model to the first language of the writer (Section 4). Weights in NB are determined, similarly to LM, by the feature counts and the *prior* probability of each candidate p (Eq. (2)). For each candidate p, NB computes the joint probability of p and the feature space F, assuming that the features are conditionally independent given p:

$$\begin{aligned} g(S,p) &= log\{prior(p) \cdot \prod_{f \in F(S,p)} P(f|p)\} \\ &= log(prior(p)) + \\ &+ \sum_{f \in F(S,p)} log(P(f|p)) \qquad (3) \end{aligned}$$

NB weights and its free coefficient are also summarized in Table 2.

2.4 SumLM

For candidate p, SumLM (Bergsma et al., 2009)[6] produces a score by summing over the logs of all feature counts:

$$\begin{aligned} g(S,p) &= \sum_{f \in F(S,p)} log(C(f)) \\ &= \sum_{f \in F(S,p)} log(P(f|p)C(p)) \\ &= |F(S,p)|C(p) + \sum_{f \in F(S,p)} log(P(f|p)) \end{aligned}$$

where $C(f)$ denotes the number of times n-gram feature f was observed with p in training. It should be clear from equation 3 that SumLM is very similar to NB, with a different free coefficient (Table 2).

3 Comparison of Algorithms

3.1 Evaluation Data

We evaluate the models using a corpus of ESL essays, annotated[7] by native English speakers (Rozovskaya and Roth, 2010a). For each preposition

[6]SumLM is one of several related methods proposed in this work; its accuracy on the preposition selection task on native English data nearly matches the best model, SuperLM (73.7% vs. 75.4%), while being much simpler to implement.

[7]The annotation of the ESL corpus can be downloaded from `http://cogcomp.cs.illinois.edu`.

Source language	Prepositions		Articles	
	Total	Incorrect	Total	Incorrect
Chinese	953	144	1864	150
Czech	627	28	575	55
Italian	687	43	-	-
Russian	1210	85	2292	213
Spanish	708	52	-	-
All	4185	352	4731	418

Table 3: **Statistics on prepositions and articles in the ESL data.** Column *Incorrect* denotes the number of cases judged to be incorrect by the annotator.

(article) used incorrectly, the annotator indicated the correct choice. The data include sentences by speakers of five first languages. Table 3 shows statistics by the source language of the writer.

3.2 Training Corpora

We use two training corpora. The first corpus, *WikiNYT*, is a selection of texts from English Wikipedia and the New York Times section of the Gigaword corpus and contains 10^7 preposition contexts. We build models of 3 sizes[8]: 10^6, $5 \cdot 10^6$, and 10^7.

To experiment with larger data sets, we use the Google Web1T 5-gram Corpus, which is a collection of n-gram counts of length one to five over a corpus of 10^{12} words. The corpus contains $2.6 \cdot 10^{10}$ prepositions. We refer to this corpus as *GoogleWeb*.

We stress that *GoogleWeb* does not contain complete sentences, but only n-gram counts. Thus, we cannot generate training data for AP for feature sets Win3 and Win4: Since the algorithm does not assume feature independence, we need to have 7 and 9-word sequences, respectively, with a preposition in the middle (as shown in Table 1) and their corpus frequencies. The other three models can be evaluated with the n-gram counts available. For example, we compute NB scores by obtaining the count of each feature independently, e.g. the count for left context 5-gram "engineer with a passion p" and right context 5-gram "p what he does .", due to the conditional independence assumption that NB makes. On *GoogleWeb*, we train NB, SumLM, and LM with three feature sets: Win2, Win3, and Win4.

From *GoogleWeb*, we also generate a smaller training set of size 10^8: We use 5-grams with a preposition in the middle and generate a new count, proportional to the size of the smaller corpus[9]. For instance, a preposition 5-gram with a count of 2600 in *GoogleWeb*, will have a count of 10 in $GoogleWeb$-10^8.

3.3 Results

Our key results of the fair comparison of the four algorithms are shown in Fig. 1 and summarized in Table 4. The table shows that AP trained on $5 \cdot 10^6$ preposition contexts performs as well as NB trained on 10^7 (i.e., with twice as much data; the performance of LM trained on 10^7 contexts is better than that of AP trained with 10 times less data (10^6), but not as good as that of AP trained with half as much data ($5 \cdot 10^6$); AP outperforms SumLM, when the latter uses 10 times more data. Fig. 1 demonstrates the performance results reported in Table 4; it shows the behavior of different systems with respect to *precision* and *recall* on the error correction task. We generate the curves by varying the decision threshold on the confidence of the classifier (Carlson et al., 2001) and propose a correction only when the confidence of the classifier is above the threshold. A higher precision and a lower recall are obtained when the decision threshold is high, and vice versa.

Key results
$AP > NB > LM > SumLM$
$AP \sim 2 \cdot NB$
$5 \cdot AP > 10 \cdot LM > AP$
$AP > 10 \cdot SumLM$

Table 4: **Key results on the comparison of algorithms**. $2 \cdot NB$ refers to NB trained with twice as much data as AP; $10 \cdot LM$ refers to LM trained with 10 times more data as AP; $10 \cdot SumLM$ refers to $SumLM$ trained with 10 times more data as AP. These results are also shown in Fig. 1.

We now show a fair comparison of the four algorithms for different window sizes, training data and training sizes. Figure 2 compares the models trained on $WikiNYT$-10^7 corpus for Win4. AP is the superior model, followed by NB, then LM, and finally SumLM.

Results for other training sizes and feature[10] set

[8] *Training size* refers to the number of preposition contexts.

[9] Scaling down $GoogleWeb$ introduces some bias but we believe that it should not have an effect on our experiments.

[10] We have also experimented with additional POS-based features that are commonly used in these tasks and observed similar behavior.

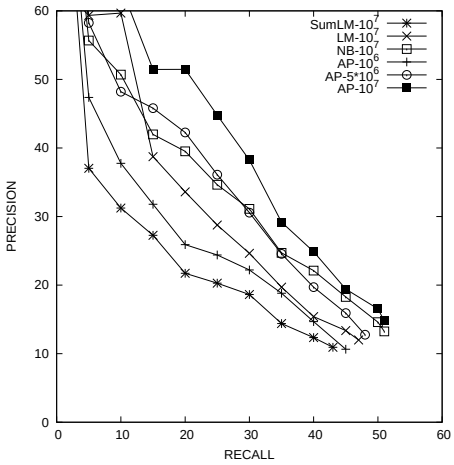


Figure 1: **Algorithm comparison across different training sizes.** (*WikiNYT, Win3*). AP (10^6 preposition contexts) performs as well as SumLM with 10 times more data, and LM requires at least twice as much data to achieve the performance of AP.

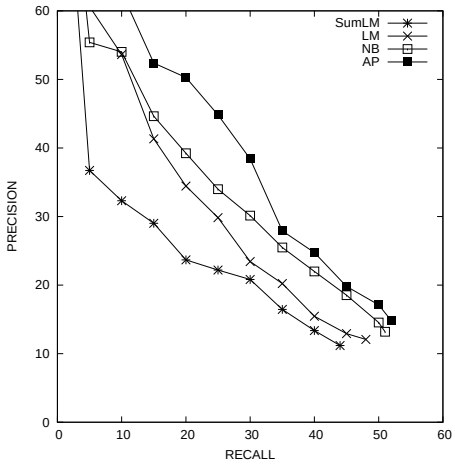


Figure 2: **Model Comparison for training data of the same size**: Performance of models for feature set *Win4* trained on $WikiNYT$-10^7.

configurations show similar behavior and are reported in Table 5, which provides model comparison in terms of *Average Area Under Curve* (AAUC, (Hanley and McNeil, 1983)). AAUC is a measure commonly used to generate a summary statistic and is computed here as an average precision value over 12 recall points (from 5 to 60):

$$AAUC = \frac{1}{12} \cdot \sum_{i=1}^{12} Precision(i \cdot 5)$$

The Table also shows results on the *article correction task*[11].

Training data	**Feature set**	**Performance** ($AAUC$)			
		AP	NB	LM	SumLM
$WikiNYT$-$5 \cdot 10^6$	$Win3$	**26**	22	20	13
$WikiNYT$-10^7	$Win4$	**33**	28	24	16
$GoogleWeb$-10^8	$Win2$	**30**	29	28	15
$GoogleWeb$	$Win4$	-	**44**	41	32
Article $WikiNYT$-$5 \cdot 10^6$	$Win3$	**40**	39	-	30

Table 5: **Performance Comparison** of the four algorithms for different training data, training sizes, and window sizes. Each row shows results for training data of the same size. The last row shows performance on the *article correction* task. All other results are for prepositions.

[11]We do not evaluate the LM approach on the article correction task, since with LM it is difficult to handle missing article errors, one of the most common error types for articles, but the expectation is that it will behave as it does for prepositions.

3.3.1 Effects of Window Size

We found that expanding window size from 2 to 3 is helpful for all of the models, but expanding window to 4 is only helpful for the models trained on *GoogleWeb* (Table 6). Compared to *Win3*, *Win4* has five additional 5-gram features. We look at the proportion of features in the ESL data that occurred in two corpora: $WikiNYT$-10^7 and *GoogleWeb* (Table 7). We observe that only 4% of test 5-grams occur in $WikiNYT$-10^7. This number goes up 7 times to 28% for *GoogleWeb*, which explains why increasing the window size is helpful for this model. By comparison, a set of native English sentences (different from the training data) has 50% more 4-grams and about 3 times more 5-grams, because ESL sentences often contain expressions not common for native speakers.

Training data	**Performance** ($AAUC$)		
	Win2	Win3	Win4
$GoogleWeb$	35	39	44

Table 6: **Effect of Window Size** in terms of $AAUC$. Performance improves, as the window increases.

4 Adapting to Writer's Source Language

In this section, we discuss adapting error correction systems to the first language of the writer. Non-native speakers make mistakes in a systematic manner, and errors often depend on the first language of the writer (Lee and Seneff, 2008; Rozovskaya and

Test	Train	N-gram length			
		2	3	4	5
ESL	$WikiNYT\text{-}10^7$	98%	66%	22%	4%
Native	$WikiNYT\text{-}10^7$	98%	67%	32%	13%
ESL	*GoogleWeb*	99%	92%	64%	28%
Native-*B09*	*GoogleWeb*	-	99%	93%	70%

Table 7: **Feature coverage for ESL and native data**. Percentage of test n-gram features that occurred in training. *Native* refers to data from Wikipedia and NYT. *B09* refers to statistics from Bergsma et al. (2009).

Roth, 2010a). For instance, a Chinese learner of English might say "congratulations *to* this achievement" instead of "congratulations *on* this achievement", while a Russian speaker might say "congratulations *with* this achievement".

A system performs much better when it makes use of knowledge about typical errors. When trained on annotated ESL data instead of native data, systems improve both precision and recall (Han et al., 2010; Gamon, 2010). Annotated data include both the writer's preposition and the intended (correct) one, and thus the knowledge about typical errors is made available to the system.

Another way to adapt a model to the first language is to generate in native training data artificial errors mimicking the typical errors of the non-native writers (Rozovskaya and Roth, 2010c; Rozovskaya and Roth, 2010b). Henceforth, we refer to this method, proposed within the discriminative framework AP, as *AP-adapted*. To determine typical mistakes, error statistics are collected on a small set of annotated ESL sentences. However, for the model to use these language-specific error statistics, a separate classifier for each source language needs to be trained.

We propose a novel adaptation method, which shows performance improvement over *AP-adapted*. Moreover, this method is much simpler to implement, since there is no need to train per source language; only one classifier is trained. The method relies on the observation that error regularities can be viewed as a *distribution on priors over the correction candidates*. Given a preposition *s* in text, the *prior* for candidate p is the probability that p is the correct preposition for *s*. If a model is trained on native data without adaptation to the source language, candidate priors correspond to the relative frequencies of the candidates in the native training data. More importantly, these priors remain the same regardless of the source language of the writer or of the preposition used in text. From the model's perspective, it means that a correction candidate, for example *to*, is equally likely given that the author's preposition is *for* or *from*, which is clearly incorrect and disagrees with the notion that errors are regular and language-dependent.

We use the annotated ESL data and define *adapted* candidate priors that are dependent on the author's preposition and the author's source language. Let *s* be a preposition appearing in text by a writer of source language L_1, and *p* a correction candidate. Then the *adapted* prior of *p* given *s* is:

$$prior(p, s, L_1) = \frac{C_{L_1}(s,p)}{C_{L_1}(s)},$$

where $C_{L_1}(s)$ denotes the number of times *s* appeared in the ESL data by L_1 writers, and $C_{L_1}(s,p)$ denotes the number of times *p* was the correct preposition when *s* was used by an L_1 writer.

Table 8 shows adapted candidate priors for two author's choices – when an ESL writer used *on* and *at* – based on the data from Chinese learners. One key distinction of the adapted priors is the high probability assigned to the author's preposition: the new prior for *on* given that it is also the preposition found in text is 0.70, vs. the 0.07 prior based on the native data. The adapted prior of preposition p, when p is used, is always high, because the majority of prepositions are used correctly. Higher probabilities are also assigned to those candidates that are most often observed as corrections for the author's preposition. For example, the adapted prior for *at* when the writer chose *on* is 0.10, since *on* is frequently incorrectly chosen instead of *at*.

To determine a mechanism to inject the adapted priors into a model, we note that while all of our models use priors in some way, NB architecture directly specifies the prior probability as one of its parameters (Sec. 2.3). We thus train NB in a traditional way, on native data, and then replace the prior component in Eq. (3) with the adapted prior, language and preposition dependent, to get the score for *p* of the *NB-adapted* model:

$$g(S,p) = log\{prior(p, s, L_1) \cdot \prod_{f \in F(S,p)} P(f|p)\}$$

Candidate	Global prior	Adapted prior			
		author's choice	prior	author's choice	prior
of	0.25	on	0.03	at	0.02
to	0.22	on	0.06	at	0.00
in	0.15	on	0.04	at	0.16
for	0.10	on	0.00	at	0.03
on	0.07	on	**0.70**	at	0.09
by	0.06	on	0.00	at	0.02
with	0.06	on	0.04	at	0.00
at	0.04	on	0.10	at	**0.75**
from	0.04	on	0.00	at	0.02
about	0.01	on	0.03	at	0.00

Table 8: **Examples of *adapted* candidate priors for two author's choices – *on* and *at* – based on the errors made by Chinese learners**. *Global prior* denotes the probability of the candidate in the standard model and is based on the relative frequency of the candidate in native training data. *Adapted priors* are dependent on the author's preposition and the author's first language. Adapted priors for the author's choice are very high. Other candidates are given higher priors if they often appear as corrections for the author's choice.

We stress that in the new method there is no need to train per source language, as with previous adaption methods. Only one model is trained, and only at decision time, we change the prior probabilities of the model. Also, while we need a lot of data to train the model, only one parameter depends on annotated data. Therefore, with rather small amounts of data, it is possible to get reasonably good estimates of these prior parameters.

In the experiments below, we compare four models: *AP*, *NB AP-adapted* and *NB-adapted*. *AP-adapted* is the adaptation through artificial errors and *NB-adapted* is the method proposed here. Both of the adapted models use the same error statistics in k-fold cross-validation (CV): We randomly partition the ESL data into k parts, with each part tested on the model that uses error statistics estimated on the remaining $k-1$ parts. We also remove all preposition errors that occurred only once (23% of all errors) to allow for a better evaluation of the adapted models. Although we observe similar behavior on all the data, the models especially benefit from the adapted priors when a particular error occurred more than once. Since the majority of errors are not due to chance, we focus on those errors that the writers will make repeatedly.

Fig. 3 shows the four models trained on $WikiNYT$-10^7. First, we note that the adapted

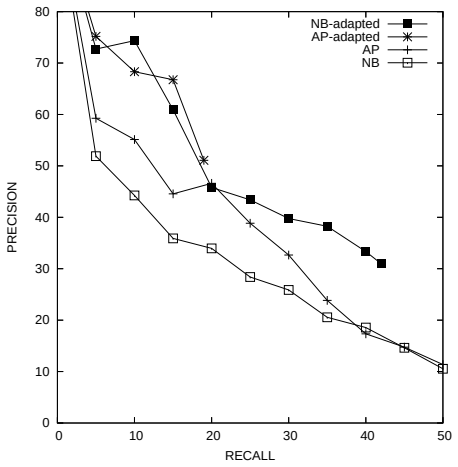


Figure 3: **Adapting to Writer's Source Language**. *NB-adapted* is the method proposed here. *AP-adapted* and *NB-adapted* results are obtained using 2-fold CV, with 50% of the ESL data used for estimating the new priors. All models are trained on $WikiNYT$-10^7.

models outperform their non-adapted counterparts with respect to precision. Second, for the recall points less than 20%, the adapted models obtain very similar precision values. This is interesting, especially because *NB* does not perform as well as *AP*, as we also showed in Sec. 3.3. Thus, *NB-adapted* not only improves over *NB*, but its gap compared to the latter is much wider than the gap between the AP-based systems. Finally, an important performance distinction between the two adapted models is the loss in recall exhibited by *AP-adapted* – its curve is shorter because *AP-adapted* is very conservative and does not propose many corrections. In contrast, *NB-adapted* succeeds in improving its precision over *NB* with almost no recall loss.

To evaluate the effect of the size of the data used to estimate the new priors, we compare the performance of *NB-adapted* models in three settings: *2-fold CV*, *10-fold CV*, and *Leave-One-Out* (Figure 4). In *2-fold CV*, priors are estimated on 50% of the ESL data, in *10-fold* on 90%, and in *Leave-One-Out* on all data but the testing example. Figure 4 shows the averaged results over 5 runs of CV for each setting. The model converges very quickly: there is almost no difference between *10-fold CV* and *Leave-One-Out*, which suggests that we can get a good estimate of the priors using just a little annotated data.

Table 9 compares *NB* and *NB-adapted* for two corpora: $WikiNYT$-10^7 and $GoogleWeb$. Since

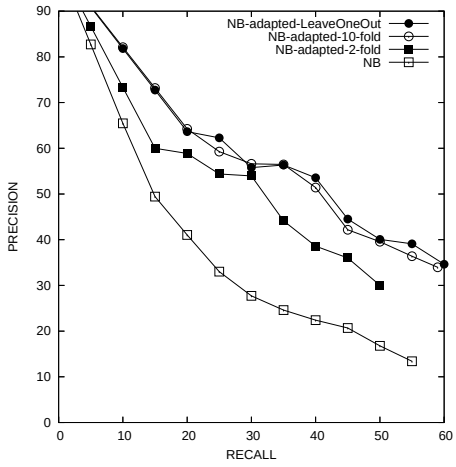


Figure 4: **How much data are needed to estimate adapted priors**. Comparison of *NB-adapted* models trained on *GoogleWeb* that use different amounts of data to estimate the new priors. In *2-fold CV*, priors are estimated on 50% of the data; in *10-fold* on 90% of the data; in *Leave-One-Out*, the new priors are based on all the data but the testing example.

$GoogleWeb$ is several orders of magnitude larger, the adapted model behaves better for this corpus.

So far, we have discussed performance in terms of precision and recall, but we can also discuss it in terms of accuracy, to see how well the algorithm is performing compared to the *baseline* on the task. Following Rozovskaya and Roth (2010c), we consider as the baseline the accuracy of the ESL data before applying the model[12], or *the percentage of prepositions used correctly* in the test data. From Table 3, the baseline is 93.44%[13]. Compared to this high baseline, NB trained on $WikiNYT$-10^7 achieves an accuracy of 93.54, and *NB-adapted* achieves an accuracy of 93.93[14].

Training data	Algorithms	
	NB	NB-adapted
$WikiNYT$-10^7	29	**53**
$GoogleWeb$	38	**62**

Table 9: **Adapting to writer's source language.** Results are reported in terms of $AAUC$. *NB-adapted* is the model with adapted priors. Results for *NB-adapted* are based on 10-fold CV.

[12]Note that this baseline is different from the *majority* baseline used in the preposition *selection* task, since here we have the author's preposition in text.

[13]This is the baseline after removing the singleton errors.

[14]We select the best accuracy among different values that can be achieved by varying the decision threshold.

5 Conclusion

We have addressed two important issues in ESL error correction, which are essential to making progress in this task. First, we presented an extensive, fair comparison of four popular linear learning models for the task and demonstrated that there are significant performance differences between the approaches. Since all of the algorithms presented here are linear, the only difference is in how they learn the weights. Our experiments demonstrated that the discriminative approach (AP) is able to generalize better than any of the other models. These results correct earlier conclusions, made with incomparable data sets. The model comparison was performed using two popular tasks – correcting errors in article and preposition usage – and we expect that our results will generalize to other ESL correction tasks.

The second, and most important, contribution of the paper is a novel method that allows one to adapt the learned model to the source language of the writer. We showed that error patterns can be viewed as a distribution on priors over the correction candidates and proposed a method of injecting the adapted priors into the learned model. In addition to performing much better than the previous approaches, this method is also very cheap to implement, since it does not require training a separate model for each source language, but adapts the system to the writer's language at decision time.

Acknowledgments

The authors thank Nick Rizzolo for many helpful discussions. The authors also thank Josh Gioja, Nick Rizzolo, Mark Sammons, Joel Tetreault, Yuancheng Tu, and the anonymous reviewers for their insightful comments. This research is partly supported by a grant from the U.S. Department of Education.

References

S. Bergsma, D. Lin, and R. Goebel. 2009. Web-scale n-gram models for lexical disambiguation. In *21st International Joint Conference on Artificial Intelligence*, pages 1507–1512.

J. Bitchener, S. Young, and D. Cameron. 2005. The effect of different types of corrective feedback on ESL student writing. *Journal of Second Language Writing*.

A. Carlson, J. Rosen, and D. Roth. 2001. Scaling up context sensitive text correction. In *Proceedings of the*

National Conference on Innovative Applications of Artificial Intelligence (IAAI), pages 45–50.

Y. S. Chan and H. T. Ng. 2005. Word sense disambiguation with distribution estimation. In *Proceedings of IJCAI 2005*.

S. Chen and J. Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of ACL 1996*.

M. Chodorow, J. Tetreault, and N.-R. Han. 2007. Detection of grammatical errors involving prepositions. In *Proceedings of the Fourth ACL-SIGSEM Workshop on Prepositions*, pages 25–30, Prague, Czech Republic, June. Association for Computational Linguistics.

G. Dalgish. 1985. Computer-assisted ESL research. *CALICO Journal*, 2(2).

J. Eeg-Olofsson and O. Knuttson. 2003. Automatic grammar checking for second language learners - the use of prepositions. *Nodalida*.

A. Elghaari, D. Meurers, and H. Wunsch. 2010. Exploring the data-driven prediction of prepositions in english. In *Proceedings of COLING 2010*, Beijing, China.

R. De Felice and S. Pulman. 2008. A classifier-based approach to preposition and determiner error correction in L2 English. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 169–176, Manchester, UK, August.

Y. Freund and R. E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

M. Gamon, J. Gao, C. Brockett, A. Klementiev, W. Dolan, D. Belenko, and L. Vanderwende. 2008. Using contextual speller techniques and language modeling for ESL error correction. In *Proceedings of IJCNLP*.

M. Gamon. 2010. Using mostly native data to correct errors in learners' writing. In *NAACL*, pages 163–171, Los Angeles, California, June.

A. R. Golding and D. Roth. 1999. A Winnow based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130.

N. Han, M. Chodorow, and C. Leacock. 2006. Detecting errors in English article usage by non-native speakers. *Journal of Natural Language Engineering*, 12(2):115–129.

N. Han, J. Tetreault, S. Lee, and J. Ha. 2010. Using an error-annotated learner corpus to develop and ESL/EFL error correction system. In *LREC*, Malta, May.

J. Hanley and B. McNeil. 1983. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148(3):839–843.

E. Izumi, K. Uchimoto, T. Saiga, T. Supnithi, and H. Isahara. 2003. Automatic error detection in the Japanese learners' English spoken data. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, pages 145–148, Sapporo, Japan, July.

C. Leacock, M. Chodorow, M. Gamon, and J. Tetreault. 2010. Morgan and Claypool Publishers.

J. Lee and S. Seneff. 2008. An analysis of grammatical errors in non-native speech in English. In *Proceedings of the 2008 Spoken Language Technology Workshop*.

V. Punyakanok, D. Roth, and W. Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2).

N. Rizzolo and D. Roth. 2007. Modeling Discriminative Global Inference. In *Proceedings of the First International Conference on Semantic Computing (ICSC)*, pages 597–604, Irvine, California, September. IEEE.

D. Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 806–813.

D. Roth. 1999. Learning in natural language. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 898–904.

A. Rozovskaya and D. Roth. 2010a. Annotating ESL errors: Challenges and rewards. In *Proceedings of the NAACL Workshop on Innovative Use of NLP for Building Educational Applications*.

A. Rozovskaya and D. Roth. 2010b. Generating confusion sets for context-sensitive error correction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

A. Rozovskaya and D. Roth. 2010c. Training paradigms for correcting errors in grammar and usage. In *Proceedings of the NAACL-HLT*.

A. Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing*, pages 257–286, November.

J. Tetreault and M. Chodorow. 2008. The ups and downs of preposition error detection in ESL writing. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 865–872, Manchester, UK, August.

J. Tetreault, J. Foster, and M. Chodorow. 2010. Using parse features for preposition selection and error detection. In *ACL*.

Automated Whole Sentence Grammar Correction Using a Noisy Channel Model

Y. Albert Park
Department of Computer Science and Engineering
9500 Gilman Drive
La Jolla, CA 92037-404, USA
yapark@ucsd.edu

Roger Levy
Department of Linguistics
9500 Gilman Drive
La Jolla, CA 92037-108, USA
rlevy@ucsd.edu

Abstract

Automated grammar correction techniques have seen improvement over the years, but there is still much room for increased performance. Current correction techniques mainly focus on identifying and correcting a specific type of error, such as verb form misuse or preposition misuse, which restricts the corrections to a limited scope. We introduce a novel technique, based on a noisy channel model, which can utilize the whole sentence context to determine proper corrections. We show how to use the EM algorithm to learn the parameters of the noise model, using only a data set of erroneous sentences, given the proper language model. This frees us from the burden of acquiring a large corpora of corrected sentences. We also present a cheap and efficient way to provide automated evaluation results for grammar corrections by using BLEU and METEOR, in contrast to the commonly used manual evaluations.

1 Introduction

The process of editing written text is performed by humans on a daily basis. Humans work by first identifying the writer's intent, and then transforming the text so that it is coherent and error free. They can read text with several spelling errors and grammatical errors and still easily identify what the author originally meant to write. Unfortunately, current computer systems are still far from such capabilities when it comes to the task of recognizing incorrect text input. Various approaches have been taken, but to date it seems that even many spell checkers such as Aspell do not take context into consideration, which prevents them from finding misspellings which have the same form as valid words. Also, current grammar correction systems are mostly rule-based, searching the text for defined types of rule violations in the English grammar. While this approach has had some success in finding various grammatical errors, it is confined to specifically defined errors.

In this paper, we approach this problem by modeling various types of human errors using a noisy channel model (Shannon, 1948). Correct sentences are produced by a predefined generative probabilistic model, and lesioned by the noise model. We learn the noise model parameters using an expectation-maximization (EM) approach (Dempster et al., 1977; Wu, 1983). Our model allows us to deduce the original intended sentence by looking for the the highest probability parses over the entire sentence, which leads to automated whole sentence spelling and grammar correction based on contextual information.

In Section 2, we discuss previous work, followed by an explanation of our model and its implementation in Sections 3 and 4. In Section 5 we present a novel technique for evaluating the task of automated grammar and spelling correction, along with the data set we collected for our experiments. Our experiment results and discussion are in Section 6. Section 7 concludes this paper.

2 Background

Much of the previous work in the domain of automated grammar correction has focused on identi-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 934–944,
Portland, Oregon, June 19-24, 2011.

fying grammatical errors. Chodorow and Leacock (2000) used an unsupervised approach to identifying grammatical errors by looking for contextual cues in a ± 2 word window around a target word. To identify errors, they searched for cues which did not appear in the correct usage of words. Eeg-olofsson and Knutsson (2003) used rule-based methods to approach the problem of discovering preposition and determiner errors of L2 writers, and various classifier-based methods using Maximum Entropy models have also been proposed (Izumi et al., 2003; Tetreault and Chodorow, 2008; De Felice and Pulman, 2008). Some classifier-based methods can be used not only to identify errors, but also to determine suggestions for corrections by using the scores or probabilities from the classifiers for other possible words. While this is a plausible approach for grammar correction, there is one fundamental difference between this approach and the way humans edit. The output scores of classifiers do not take into account the observed erroneous word, changing the task of editing into a fill-in-the-blank selection task. In contrast, editing makes use of the writer's erroneous word which often encompasses information neccessary to correctly deduce the writer's intent.

Generation-based approaches to grammar correction have also been taken, such as Lee and Seneff (2006), where sentences are paraphrased into an over-generated word lattice, and then parsed to select the best rephrasing. As with the previously mentioned approaches, these approaches often have the disadvantage of ignoring the writer's selected word when used for error correction instead of just error detection.

Other work which relates to automated grammar correction has been done in the field of machine translation. Machine translation systems often generate output which is grammatically incorrect, and automated post-editing systems have been created to address this problem. For instance, when translating Japanese to English, the output sentence needs to be edited to include the correct articles, since the Japanese language does not contain articles. Knight and Chander (1994) address the problem of selecting the correct article for MT systems. These types of systems could also be used to facilitate grammar correction.

While grammar correction can be used on the output of MT systems, note that the task of grammar correction itself can also be thought of as a machine translation task, where we are trying to 'translate' a sentence from an 'incorrect grammar' language to a 'correct grammar' language. Under this idea, the use of statistical machine translation techniques to correct grammatical errors has also been explored. Brockett et al. (2006) uses phrasal SMT techniques to identify and correct mass noun errors of ESL students. Désilets and Hermet (2009) use a round-trip translation from L2 to L1 and back to L2 to correct errors using an SMT system, focusing on errors which link back to the writer's native language.

Despite the underlying commonality between the tasks of machine translation and grammar correction, there is a practical difference in that the field of grammar correction suffers from a lack of good quality parallel corpora. While machine translation has taken advantage of the plethora of translated documents and books, from which various corpora have been built, the field of grammar correction does not have this luxury. Annotated corpora of grammatical errors do exist, such as the NICT Japanese Learner of English corpus and the Chinese Learner English Corpus (Shichun and Huizhong, 2003), but the lack of definitive corpora often makes obtaining data for use in training models a task within itself, and often limits the approaches which can be taken.

Using classification or rule-based systems for grammatical error detection has proven to be successful to some extent, but many approaches are not sufficient for real-world automated grammar correction for various of reasons. First, as we have already mentioned, classification systems and generation-based systems do not make full use of the given data when trying to make a selection. This limits the system's ability to make well-informed edits which match the writer's original intent. Second, many of the systems start with the assumption that there is only one type of error. However, ESL students often make several combined mistakes in one sentence. These combined mistakes can throw off error detection/correction schemes which assume that the rest of the sentence is correct. For example, if a student erroneously writes 'much poeple' instead of 'many people', a system trying to correct 'many/much' errors may skip correction of much to many because it does not have any reference to the misspelled word

'poeple'. Thus there are advantages in looking at the sentence as a whole, and creating models which allow several types of errors to occur within the same sentence. We now present our model, which supports the addition of various types of errors into one combined model, and derives its response by using the whole of the observed sentence.

3 Base Model

Our noisy channel model consists of two main components, a base language model and a noise model. The base language model is a probabilistic language model which generates an 'error-free' sentence[1] with a given probability. The probabilistic noise model then takes this sentence and decides whether or not to make it erroneous by inserting various types of errors, such as spelling mistakes, article choice errors, wordform choice errors, etc., based on its parameters (see Figure 1 for example). Using this model, we can find the posterior probability $p(S_{orig}|S_{obs})$ using Bayes rule where S_{orig} is the original sentence created by our base language model, and S_{obs} is the observed erroneous sentence.

$$p(S_{orig}|S_{obs}) = \frac{p(S_{obs}|S_{orig})p(S_{orig})}{p(S_{obs})}$$

For the language model, we can use various known probabilistic models which already have defined methods for learning the parameters, such as n-gram models or PCFGs. For the noise model, we need some way to learn the parameters for the mistakes that a group of specified writers (such as Korean ESL students) make. We address this issue in Section 4.

Using this model, we can find the highest likelihood error-free sentence for an observed output sentence by tracing all possible paths from the language model through the noise model and ending in the observed sentence as output.

4 Implementation

To actually implement our model, we use a bigram model for the base language model, and various noise models which introduce spelling errors, article choice errors, preposition choice errors, etc.

[1] In reality, the language model will most likely produce sentences with errors as seen by humans, but from the modeling perspective, we assume that the language model is a perfect representation of the language for our task.

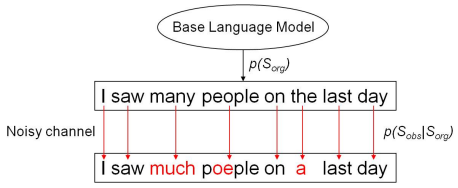


Figure 1: Example of noisy channel model

All models are implemented using weighted finite-state tranducers (wFST). For operations on the wFSTs, we use OpenFST (Allauzen et al., 2007), along with expectation semiring code supplied by Markus Dryer for Dreyer et al. (2008).

4.1 Base language model

The base language model is a bigram model implemented by using a weighted finite-state transducer (wFST). The model parameters are learned from the British National Corpus modified to use American English spellings with Kneser-Ney smoothing. To lower our memory usage, only bigrams whose words are found in the observed sentences, or are determined to be possible candidates for the correct words of the original sentence (due to the noise models) are used. While we use a bigram model here for simplicity, any probabilistic language model having a tractable intersection with wFSTs could be used. For the bigram model, each state in the wFST represents a bigram context, except the end state. The arcs of the wFST are set so that the weight is the bigram probability of the output word given the context specified by the from state, and the output word is a word of the vocabulary. Thus, given a set of n words in the vocabulary, the language model wFST had one start state, from which n arcs extended to each of their own context states. From each of these nodes, $n + 1$ arcs extend to each of the n context states and the end state. Thus the number of states in the language model is $n + 2$ and the number of arcs is $O(n^2)$.

4.2 Noise models

For our noise model, we created a weighted finite-state transducer (wFST) which accepts error-free input, and outputs erroneous sentences with a specified probability. To model various types of human errors, we created several different noise models and

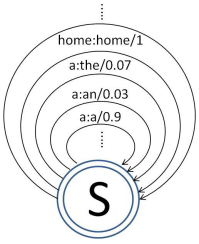


Figure 2: Example of noise model

composed them together, creating a layered noise model. The noise models we implement are spelling errors, article choice errors, preposition choice errors, and insertion errors, which we will explain in more detail later in this section.

The basic design of each noise wFST starts with an initial state, which is also the final state of the wFST. For each word found in the language model, an arc going from the initial state to itself is created, with the input and output values set as the word. These arcs model the case of no error being made. In addition to these arcs, arcs representing prediction errors are also inserted. For example, in the article choice error model, an arc is added for each possible (*input, output*) article pair, such as *a*:*an* for making the mistake of writing *an* instead of *a*. The weights of the arcs are the probabilities of introducing errors, given the input word from the language model. For example, the noise model shown in Figure 2 shows a noise model in which *a* will be written correctly with a probability of 0.9, and will be changed to *an* or *the* with probabilities 0.03 and 0.07, respectively. For this model to work correctly, the setting of the probabilities for each error is required. How this is done is explained in Section 4.3.

4.2.1 Spelling errors

The spelling error noise model accounts for spelling errors made by writers. For spelling errors, we allowed all spelling errors which were a Damerau-Levenshtein distance of 1 (Damerau, 1964; Levenshtein, 1966). While allowing a DL distance of 2 or higher may likely have better performance, the model was constrained to a distance of 1 due to memory constraints. We specified one parameter λ_n for each possible word length n. This parameter is the total probability of making a spelling error for a given word length. For each word length we distributed the probability of each possible spelling error equally. Thus for word length n, we have n deletion errors, $25n$ substitution errors, $n-1$ transposition errors, and $26(n+1)$ insertion errors, and the probability for each possible error is $\frac{\lambda_n}{n+25n+n-1+26(n+1)}$. We set the maximum word length for spelling errors to 22, giving us 22 parameters.

4.2.2 Article choice errors

The article choice error noise model simulates incorrect selection of articles. In this model we learn $n(n-1)$ parameters, one for each article pair. Since there are only 3 articles (*a, an, the*), we only have 6 parameters for this model.

4.2.3 Preposition choice errors

The preposition choice error noise model simulates incorrect selection of prepositions. We take the 12 most commonly misused prepositions by ESL writers (Gamon et al., 2009) and specify one parameter for each preposition pair, as we do in the article choice error noise model, giving us a total of 132 parameters.

4.2.4 Wordform choice errors

The wordform choice error noise model simulates choosing the incorrect wordform of a word. For example, choosing the incorrect tense of a verb (e.g. *went→go*), or the incorrect number marking on a noun or verb (e.g. *are→is*) would be a part of this model. This error model has one parameter for every number of possible inflections, up to a maximum of 12 inflections, giving us 12 parameters. The parameter is the total probability of choosing the wrong inflection of a word, and the probability is spread evenly between each possible inflection. We used CELEX (Baayen et al., 1995) to find all the possible wordforms of each observed word.

4.2.5 Word insertion errors

The word insertion error model simulates the addition of extraneous words to the original sentence. We create a list of words by combining the prepositions and articles found in the article choice and preposition choice errors. We assume that the words on the list have a probability of being inserted erroneously. There is a parameter for each word, which

is the probability of that word being inserted. Thus we have 15 parameters for this noise model.

4.3 Learning noise model parameters

To achieve maximum performance, we wish to learn the parameters of the noise models. If we had a large set of erroneous sentences, along with a hand-annotated list of the specific errors and their corrections, it would be possible to do some form of supervised learning to find the parameters. We looked at the NICT Japanese Learner of English (JLE) corpus, which is a corpus of transcripts of 1,300 Japanese learners' English oral proficiency interview. This corpus has been annotated using an error tagset (Izumi et al., 2004). However, because the JLE corpus is a set of transcribed sentences, it is in a different domain from our task. The Chinese Learner English Corpus (CLEC) contains erroneous sentences which have been annotated, but the CLEC corpus had too many manual errors, such as typos, as well as many incorrect annotations, making it very difficult to automate the processing. Many of the corrections themselves were also incorrect. We were not able to find of a set of annotated errors which fit our task, nor are we aware that such a set exists. Instead, we collected a large data set of possibly erroneous sentences from Korean ESL students (Section 5.1). Since these sentences are not annotated, we need to use an unsupervised learning method to learn our parameters.

To learn the parameters of the noise models, we assume that the collected sentences are random output of our model, and train our model using the EM algorithm. This was done by making use of the V-expectation semiring (Eisner, 2002). The V-expectation semiring is a semiring in which the weight is defined as $\mathbb{R}_{\geq 0} \times V$, where $\mathbb{R}$ can be used to keep track of the probability, and V is a vector which can be used to denote arc traversal counts or feature counts. The weight for each of the arcs in the noise models was set so that the real value was the probability and the vector V denoted which choice (having a specified error or not) was made by selecting the arc. We create a generative language-noise model by composing the language model wFST with the noise model wFSTs, as shown in Figure 3. By using the expectation semiring, we can keep track of the probability of each path going over an erroneous arc or non-erroneous arc.

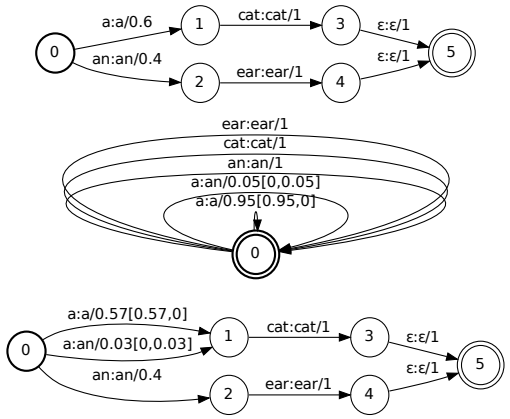


Figure 3: Example of language model (top) and noise model (middle) wFST composition. The vector of the V-expectation semiring weight is in brackets. The first value of the vector denotes no error being made on writing 'a' and the second value denotes the error of writing 'an' instead of 'a'

Once our model is set up for the E step using the initial parameters, we must compute the expected number of noise model arc traversals for use in calculating our new parameters. To do this, we need to find all possible paths resulting in the observed sentence as output, for each observed sentence. Then, for each possible path, we need to calculate the probability of the path given the output sentence, and get the expected counts of going over each erroneous and error-free arc to learn the parameters of the noise model. To find a wFST with just the possible paths for each observed sentence, we can compose the language-noise wFST with the observed sentence wFST. The observed sentence wFST is created in the following manner. Given an observed sentence, an initial state is created. For each word in the sentence, in the order appearing in the sentence, a new state is added, and an arc is created going from the previously added state to the newly added state. The new arc takes the observed word as input and also uses it as output. The weight/probability for each arc is set to 1. Composing the sentence wFST with the language-noise wFST has the effect of restricting the new wFST to only have sentences which output the observed sentence from the language-noise wFST. We now have a new wFST where all valid paths are the paths which can produce the observed

sentence. To find the total weight of all paths, we first change all input and output symbols into the empty string. Since all arcs in this wFST are epsilon arcs, we can use the epsilon-removal operation (Mohri, 2002), which will reduce the wFST to one state with no arcs. This operation combines the total weight of all paths into the final weight of the sole state, giving us the total expectation value for that sentence. By doing this for each sentence, and adding the expectation values for each sentence, we can easily compute the expectation step, from which we can find the maximizing parameters and update our parameters accordingly.

4.4 Finding the maximum likelihood correction

Once the parameters are learned, we can use our model to find the maximum likelihood error-free sentence. This is done by again creating the language model and noise model with the learned parameters, but this time we set the weights of the noise model to just the probabilities, using the log semiring, since we do not need to keep track of expected values. We also set the language model input for each arc to be the same word as the output, instead of using an empty string. Once again, we compose the language model with the noise models. We create a sentence wFST using the observed sentence we wish to correct, the same way the observed sentence wFST for training was created. This is now composed with the language-noise wFST. Now all we need to do is find the shortest path (when using minus-log probabilities) of the new wFST, and the input to that path will be our corrected sentence.

5 Experiment

We now present the data set and evaluation technique used for our experiments.

5.1 Data Set

To train our noise models, we collected around 25,000 essays comprised of 478,350 sentences written by Korean ESL students preparing for the TOEFL writing exam. These were collected from open web postings by Korean ESL students asking for advice on their writing samples. In order to automate the process, a program was written to download the posts, and discard the posts that were deemed too short to be TOEFL writing samples. Also discarded were the posts that had a "[re" or "re.." in the title. Next, all sentences containing Korean were removed, after which some characters were changed so that they were in ASCII form. The remaining text was separated into sentences solely by punctuation marks ., *!*, and *?*. This resulted in the 478,350 sentences stated above. Due to the process, some of the sentences collected are actually sentence fragments, where punctuation had been misused. For training and evaluation purposes, the data set was split into a test set with 504 randomly selected sentences, an evaluation set of 1017 randomly selected sentences, and a training set composed of the remaining sentences.

5.2 Evaluation technique

In the current literature, grammar correction tasks are often manually evaluated for each output correction, or evaluated by taking a set of proper sentences, artificially introducing some error, and seeing how well the algorithm fixes the error. Manual evaluation of automatic corrections may be the best method for getting a more detailed evaluation, but to do manual evaluation for every test output requires a large amount of human resources, in terms of both time and effort. In the case where artificial lesioning is introduced, the lesions may not always reflect the actual errors found in human data, and it is difficult to replicate the actual tendency of humans to make a variety of different mistakes in a single sentence. Thus, this method of evaluation, which may be suitable for evaluating the correction performance of specific grammatical errors, would not be fit for evaluating our model's overall performance. For evaluation of the given task, we have incorporated evaluation techniques based on current evaluation techniques used in machine translation, BLEU (Papineni et al., 2002) and METEOR (Lavie and Agarwal, 2007).

Machine translation addresses the problem of changing a sentence in one language to a sentence of another. The task of correcting erroneous sentences can also be thought of as translating a sentence from a given language A, to another language B, where A is a broken language, and B is the correct language. Under this context, we can apply machine translation evaluation techniques to evaluate the performance of our system. Our model's sentence correc-

tions can be thought of as the output translation to be evaluated. In order to use BLEU and METEOR, we need to have reference translations on which to score our output. As we have already explained in section 5.1, we have a collection of erroneous sentences, but no corrections. To obtain manually corrected sentences for evaluation, the test and evaluation set sentences and were put on Amazon Mechanical Turk as a correction task. Workers residing in the US were asked to manually correct the sentences in the two sets. Workers had a choice of selecting 'Impossible to understand', 'Correct sentence', or 'Incorrect sentence', and were asked to correct the sentences so no spelling errors, grammatical errors, or punctuation errors were present. Each sentence was given to 8 workers, giving us a set of 8 or fewer corrected sentences for each erroneous sentence. We asked workers not to completely rewrite the sentences, but to maintain the original structure as much as possible. Each hit was comprised of 6 sentences, and the reward for each hit was 10 cents. To ensure the quality of our manually corrected sentences, a native English speaker research assistant went over each of the 'corrected' sentences and marked them as correct or incorrect. We then removed all the incorrect 'corrections'.

Using our manually corrected reference sentences, we evaluate our model's correction performance using METEOR and BLEU. Since METEOR and BLEU are fully automated after we have our reference translations (manual corrections), we can run evaluation on our tests without any need for further manual input. While these two evaluation methods were created for machine translation, they also have the potential of being used in the field of grammar correction evaluation. One difference between machine translation and our task is that finding the right lemma is in itself something to be rewarded in MT, but is not sufficient for our task. In this respect, evaluation of grammar correction should be more strict. Thus, for METEOR, we used the 'exact' module for evaluation.

To validate our evaluation method, we ran a simple test by calculating the METEOR and BLEU scores for the observed sentences, and compared them with the scores for the manually corrected sentences, to test for an expected increase. The scores for each correction were evaluated using the set of corrected sentences minus the correction sentence being evaluated. For example, let us say we have the observed sentence o, and correction sentences c_1, c_2, c_3 and c_4 from Mechanical Turk. We run METEOR and BLEU on both o and c_1 using c_2, c_3 and c_4 as the reference set. We repeat the process for o and c_2, using c_1, c_3 and c_4 as the reference, and so on, until we have run METEOR and BLEU on all 4 correction sentences. With a set of 100 manually labeled sentences, the average METEOR score for the ESL sentences was 0.8327, whereas the corrected sentences had an average score of 0.9179. For BLEU, the average scores were 0.7540 and 0.8786, respectively, as shown in Table 1. Thus, we have confirmed that the corrected sentences score higher than the ESL sentence. It is also notable that finding corrections for the sentences is a much easier task than finding various correct translations, since the task of editing is much easier and can be done by a much larger set of qualified people.

	METEOR	BLEU
Original ESL sentences	0.8327	0.7540
Manual corrections	0.9179	0.8786

Table 1: BLEU and METEOR scores for ESL sentences vs manual corrections on 100 randomly chosen sentences

	METEOR	BLEU
Aspell	0.824144	0.719713
Spelling noise model	0.825001	0.722383

Table 2: Aspell vs Spelling noise model

6 Results

For our experiments, we used 2000 randomly selected sentences for training, and a set of 1017 annotated sentences for evaluation. We also set aside a set of 504 annotated sentences as a development set. With the 2000 sentence training, the performance generally converged after around 10 iterations of EM.

6.1 Comparison with Aspell

To check how well our spelling error noise model is doing, we compared the results of using the spelling error noise model with the output results of using the GNU Aspell 0.60.6 spelling checker. Since we

	METEOR	↑	↓	BLEU	↑	↓
ESL Baseline	0.821000			0.715634		
Spelling only	0.825001	49	5	0.722383	53	8
Spelling, Article	0.825437	55	6	0.723022	59	9
Spelling, Preposition	0.824157	52	17	0.720702	55	19
Spelling, Wordform	0.825654	81	25	0.723599	85	27
Spelling, Insertion	0.825041	52	5	0.722564	56	8

Table 3: Average evaluation scores for various noise models run on 1017 sentences, along with counts of sentences with increased (↑) and decreased (↓) scores. All improvements are significant by the binomial test at $p < 0.001$

are using METEOR and BLEU for our evaluation metric, we needed to get a set of corrected sentences for using Aspell. Aspell lists the suggested spelling corrections of misspelled words in a ranked order, so we replaced each misspelled word found by Aspell with the word with the highest rank (lowest score) for the Aspell corrections. One difference between Aspell and our model is that Aspell only corrects words which do not appear in the dictionary, while our method looks at all words, even those found in the dictionary. Thus our model can correct words which look correct by themselves, but seem to be incorrect due to the bigram context. Another difference is that Aspell has the capability to split words, whereas our model does not allow the insertion of spaces. A comparison of the scores is shown in Table 2. We can see that our model has better performance, due to better word selection, despite the advantage that Aspell has by using phonological information to find the correct word, and the disadvantage that our model is restricted to spellings which are within a Damerau-Levenstein distance of 1. This is due to the fact that our model is context-sensitive, and can use other information in addition to the misspelled word. For example, the sentence 'In contast, high prices of products would be the main reason for dislike.' was edited in Aspell by changing 'contast' to 'contest', while our model correctly selected 'contrast'. The sentence 'So i can reach the theater in ten minuets by foot' was not edited by Aspell, but our model changed 'minuets' to 'minutes'. Another difference that can be seen by looking through the results is that Aspell changes every word not found in the dictionary, while our algorithm allows words it has not seen by treating them as unknown tokens. Since we are using smoothing, these tokens are left in place if there is no other high probability bigram to take its place. This helps leave intact the proper nouns and words not in the vocabulary.

6.2 Noise model performance and output

Our next experiment was to test the performance of our model on various types of errors. Table 3 shows the BLEU and METEOR scores of our various error models, along with the number of sentences achieving improved and reduced scores. As we have already seen in section 6.1, the spelling error model increases the evaluation scores from the ESL baseline. Adding in the article choice error model and the word insertion error models in addition to the spelling error noise model increases the BLEU score performance of finding corrections. Upon observing the outputs of the corrections on the development set, we found that the corrections changing *a* to *an* were all correct. Changes between *a* and *the* were sometimes correct, and sometimes incorrect. For example, 'which makes me know a existence about' was changed to 'which makes me know *the* existence about', 'when I am in a trouble.' was changed to 'when I am in *the* trouble.', and 'many people could read a nonfiction books' was changed to 'many people could read *the* nonfiction books'. For the last correction, the manual corrections all changed the sentence to contain 'many people could read a nonfiction book', bringing down the evaluation score. Overall, the article corrections which were being made seemed to change the sentence for the better, or left it at the same quality.

The preposition choice error model decreased the performance of the system overall. Looking through the development set corrections, we found that many correct prepositions were being changed to incorrect prepositions. For example, in the sentence 'Distrust about desire between two have been growing in their

relationship.', *about* was changed to *of*, and in 'As time goes by, ...', *by* was changed to *on*. Since these changes were not found in the manual corrections, the scores were decreased.

For wordform errors, the BLEU and METEOR scores both increased. While the wordform choice noise model had the most sentences with increased scores, it also had the most sentences with decreased scores. Overall, it seems that to correct wordform errors, more context than just the preceding and following word are needed. For example, in the sentence 'There are a lot of a hundred dollar phones in the market.', *phones* was changed to *phone*. To infer which is correct, you would have to have access to the previous context 'a lot of'. Another example is '..., I prefer being indoors to going outside ...', where *going* was changed to *go*. These types of cases illustrate the restrictions of using a bigram model as the base language model.

The word insertion error model was restricted to articles and 12 prepositions, and thus did not make many changes, but was correct when it did. One thing to note is that since we are using a bigram model for the language model, the model itself is biased towards shorter sentences. Since we only included words which were needed when they were used, we did not run into problems with this bias. When we tried including a large set of commonly used words, we found that many of the words were being erased because of the bigrams models probabilistic preference for shorter sentences.

6.3 Limitations of the bigram language model

Browsing through the development set data, we found that many of our model's incorrect 'corrections' were the result of using a bigram model as our language model. For example, '.., I prefer being indoors to going outside in that...' was changed to '.., I prefer being indoors to *go* outside in that...'. From the bigram model, the probabilities $p(go \mid to)$ and $p(outside|go)$ are both higher than $p(going|to)$ and $p(outside \mid going)$, respectively. To infer that going is actually correct, we would need to know the previous context, that we are comparing 'being indoors' to 'going outside'. Unfortunately, since we are using a bigram model, this is not possible. These kind of errors are found throughout the corrections. It seems likely that making use of a language model which can keep track of this kind of information would increase the performance of the correction model by preventing these kinds of errors.

7 Conclusion and future work

We have introduced a novel way of finding grammar and spelling corrections, which uses the EM algorithm to train the parameters of our noisy channel approach. One of the benefits of this approach is that it does not require a parallel set of erroneous sentences and their corrections. Also, our model is not confined to a specific error, and various error models may be added on. For training our noise model, all that is required is finding erroneous data sets. Depending on which domain you are training on, this can also be quite feasible as we have shown by our collection of Korean ESL students' erroneous writing samples. Our data set could have been for ESL students of any native language, or could also be a data set of other groups such as young native English speakers, or the whole set of English speakers for grammar correction. Using only these data sets, we can train our noisy channel model, as we have shown using a bigram language model, and a wFST for our noise model. We have also shown how to use weighted finite-state transducers and the expectation semiring, as well as wFST algorithms implemented in OpenFST to train the model using EM. For evaluation, we have introduced a novel way of evaluating grammar corrections, using MT evaluation methods, which we have not seen in other grammar correction literature. The produced corrections show the restrictions of using a bigram language model. For future work, we plan to use a more accurate language model, and add more types of complex error models, such as word deletion and word ordering error models to improve performance and address other types of errors.

Acknowledgments

We are grateful to Randy West for his input and assistance, and to Markus Dreyer who provided us with his expectation semiring code. We would also like to thank the San Diego Supercomputer Center for use of their DASH high-performance computing system.

References

Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., and Mohri, M. (2007). OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23. Springer. `http://www.openfst.org`.

Baayen, H. R., Piepenbrock, R., and Gulikers, L. (1995). *The CELEX Lexical Database. Release 2 (CD-ROM)*. Linguistic Data Consortium, University of Pennsylvania, Philadelphia, Pennsylvania.

Brockett, C., Dolan, W. B., and Gamon, M. (2006). Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 249–256, Morristown, NJ, USA. Association for Computational Linguistics.

Chodorow, M. and Leacock, C. (2000). An unsupervised method for detecting grammatical errors. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 140–147, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7:171–176.

De Felice, R. and Pulman, S. G. (2008). A classifier-based approach to preposition and determiner error correction in L2 English. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 169–176, Morristown, NJ, USA. Association for Computational Linguistics.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):pp. 1–38.

Désilets, A. and Hermet, M. (2009). Using automatic roundtrip translation to repair general errors in second language writing. In *Proceedings of the twelfth Machine Translation Summit*, MT Summit XII, pages 198–206.

Dreyer, M., Smith, J., and Eisner, J. (2008). Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1080–1089, Honolulu, Hawaii. Association for Computational Linguistics.

Eeg-olofsson, J. and Knutsson, O. (2003). Automatic grammar checking for second language learners - the use of prepositions. In *In Nodalida*.

Eisner, J. (2002). Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–8, Philadelphia.

Gamon, M., Leacock, C., Brockett, C., Dolan, W. B., Gao, J., Belenko, D., and Klementiev, A. (2009). Using statistical techniques and web search to correct ESL errors. In *Calico Journal*, Vol 26, No. 3, pages 491–511, Menlo Park, CA, USA. CALICO Journal.

Izumi, E., Uchimoto, K., and Isahara, H. (2004). The NICT JLE corpus exploiting the language learnersspeech database for research and education. In *International Journal of the Computer, the Internet and Management*, volume 12(2), pages 119–125.

Izumi, E., Uchimoto, K., Saiga, T., Supnithi, T., and Isahara, H. (2003). Automatic error detection in the Japanese learners' English spoken data. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 2*, ACL '03, pages 145–148, Morristown, NJ, USA. Association for Computational Linguistics.

Knight, K. and Chander, I. (1994). Automated postediting of documents. In *Proceedings of the twelfth national conference on Artificial intelligence (vol. 1)*, AAAI '94, pages 779–784, Menlo Park, CA, USA. American Association for Artificial Intelligence.

Lavie, A. and Agarwal, A. (2007). Meteor: an automatic metric for MT evaluation with high levels of correlation with human judgments. In *StatMT '07: Proceedings of the Second Workshop on*

Statistical Machine Translation, pages 228–231, Morristown, NJ, USA. Association for Computational Linguistics.

Lee, J. and Seneff, S. (2006). Automatic grammar correction for second-language learners. In *Proceedings of Interspeech*.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710.

Mohri, M. (2002). Generic epsilon-removal and input epsilon-normalization algorithms for weighted transducers. In *International Journal of Foundations of Computer Science 13*, pages 129–143.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Shannon, C. (1948). A mathematical theory of communications. *Bell Systems Technical Journal*, 27(4):623–656.

Shichun, G. and Huizhong, Y. (2003). Chinese Learner English Corpus. Shanghai Foreign Language Education Press.

Tetreault, J. R. and Chodorow, M. (2008). The ups and downs of preposition error detection in ESL writing. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 865–872, Morristown, NJ, USA. Association for Computational Linguistics.

Wu, C.-F. J. (1983). On the convergence properties of the EM algorithm. *Ann. Statist.*, 11(1):95–103.

A Generative Entity-Mention Model for Linking Entities with Knowledge Base

Xianpei Han **Le Sun**
Institute of Software, Chinese Academy of Sciences
HaiDian District, Beijing, China.
{xianpei, sunle}@nfs.iscas.ac.cn

Abstract

Linking entities with knowledge base (entity linking) is a key issue in bridging the textual data with the structural knowledge base. Due to the name variation problem and the name ambiguity problem, the entity linking decisions are critically depending on the heterogenous knowledge of entities. In this paper, we propose a generative probabilistic model, called *entity-mention model*, which can leverage heterogenous entity knowledge (including *popularity knowledge*, *name knowledge* and *context knowledge*) for the entity linking task. In our model, each name mention to be linked is modeled as a sample generated through a three-step generative story, and the entity knowledge is encoded in the distribution of entities in document *P(e)*, the distribution of possible names of a specific entity *P(s|e)*, and the distribution of possible contexts of a specific entity *P(c|e)*. To find the referent entity of a name mention, our method combines the evidences from all the three distributions *P(e)*, *P(s|e)* and *P(c|e)*. Experimental results show that our method can significantly outperform the traditional methods.

1 Introduction

In recent years, due to the proliferation of knowledge-sharing communities like *Wikipedia* [1] and the many research efforts for the automated knowledge base population from Web like the *Read the Web*[2] project, more and more large-scale knowledge bases are available. These knowledge bases contain rich knowledge about the world's entities, their semantic properties, and the semantic relations between each other. One of the most notorious examples is *Wikipedia*: its 2010 English version contains more than 3 million entities and 20 million semantic relations. Bridging these knowledge bases with the textual data can facilitate many different tasks such as entity search, information extraction and text classification. For example, as shown in Figure 1, knowing the word *Jordan* in the document refers to a basketball player and the word *Bulls* refers to a NBA team would be helpful in classifying this document into the *Sport/Basketball* class.

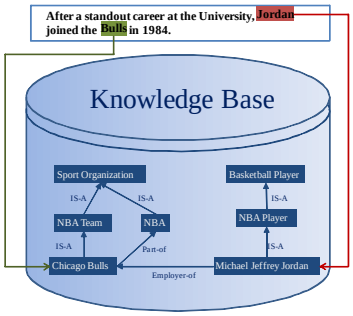


Figure 1. A Demo of Entity Linking

A key issue in bridging the knowledge base with the textual data is linking the entities in a document with their referents in a knowledge base, which is usually referred to as the *Entity Linking* task. Given a set of name mentions $M = \{m_1, m_2, \ldots, m_k\}$ contained in documents and a knowledge base *KB* containing a set of entities $E = \{e_1, e_2, \ldots, e_n\}$, an entity linking system is a function $\sigma : M \rightarrow E$ which links these name mentions to their referent entities in *KB*. For example, in Figure 1 an entity linking system should link the name mention *Jordan* to the entity *Michael Jeffrey Jordan* and the name mention *Bulls* to the entity *Chicago Bulls*.

The entity linking task, however, is not trivial due to the name variation problem and the name ambiguity problem. Name variation means that an entity can be mentioned in different ways such as *full name*, *aliases*, *acronyms* and *misspellings*. For

[1] http://www.wikipedia.org/
[2] http://rtw.ml.cmu.edu/

example, the entity *Michael Jeffrey Jordan* can be mentioned using more than 10 names, such as *Michael Jordan*, *MJ* and *Jordan*. The name ambiguity problem is related to the fact that a name may refer to different entities in different contexts. For example, the name *Bulls* can refer to more than 20 entities in Wikipedia, such as the NBA team *Chicago Bulls*, the football team *Belfast Bulls* and the cricket team *Queensland Bulls*.

Complicated by the name variation problem and the name ambiguity problem, the entity linking decisions are critically depending on the knowledge of entities (Li et al., 2004; Bunescu & Pasca, 2006; Cucerzan, 2007; Milne & Witten, 2008 and Fader et al., 2009). Based on the previous work, we found that the following three types of entity knowledge can provide critical evidence for the entity linking decisions:

- **Popularity Knowledge**. The popularity knowledge of entities tells us the likelihood of an entity appearing in a document. In entity linking, the entity popularity knowledge can provide a *priori* information to the possible referent entities of a name mention. For example, without any other information, the popularity knowledge can tell that in a Web page the name "*Michael Jordan*" will more likely refer to the notorious basketball player *Michael Jeffrey Jordan*, rather than the less popular Berkeley professor *Michael I. Jordan*.
- **Name Knowledge**. The name knowledge tells us the possible names of an entity and the likelihood of a name referring to a specific entity. For example, we would expect the name knowledge tells that both the "*MJ*" and "*Michael Jordan*" are possible names of the basketball player *Michael Jeffrey Jordan*, but the "*Michael Jordan*" has a larger likelihood. The name knowledge plays the central role in resolving the name variation problem, and is also helpful in resolving the name ambiguity problem.
- **Context Knowledge**. The context knowledge tells us the likelihood of an entity appearing in a specific context. For example, given the context "*__wins NBA MVP*", the name "*Michael Jordan*" should more likely refer to the basketball player *Michael Jeffrey Jordan* than the Berkeley professor *Michael I. Jordan*. Context knowledge is crucial in solving the name ambiguities.

Unfortunately, in entity linking system, the modeling and exploitation of these types of entity knowledge is not straightforward. As shown above, these types of knowledge are heterogenous, making it difficult to be incorporated in the same model. Furthermore, in most cases the knowledge of entities is not explicitly given, making it challenging to extract the entity knowledge from data.

To resolve the above problems, this paper proposes a generative probabilistic model, called *entity-mention model*, which can leverage the heterogeneous entity knowledge (including popularity knowledge, name knowledge and context knowledge) for the entity linking task. In our model, each name mention is modeled as a sample generated through a three-step generative story, where the entity knowledge is encoded in three distributions: the entity popularity knowledge is encoded in the distribution of entities in document $P(e)$, the entity name knowledge is encoded in the distribution of possible names of a specific entity $P(s|e)$, and the entity context knowledge is encoded in the distribution of possible contexts of a specific entity $P(c|e)$. The $P(e)$, $P(s|e)$ and $P(c|e)$ are respectively called the *entity popularity model*, the *entity name model* and the *entity context model*. To find the referent entity of a name mention, our method combines the evidences from all the three distributions $P(e)$, $P(s|e)$ and $P(c|e)$. We evaluate our method on both Wikipedia articles and general newswire documents. Experimental results show that our method can significantly improve the entity linking accuracy.

Our Contributions. Specifically, the main contributions of this paper are as follows:

1) We propose a new generative model, the *entity-mention model*, which can leverage heterogenous entity knowledge (including popularity knowledge, name knowledge and context knowledge) for the entity linking task;

2) By modeling the entity knowledge as probabilistic distributions, our model has a statistical foundation, making it different from most previous *ad hoc* approaches.

This paper is organized as follows. The entity-mention model is described in Section 2. The model estimation is described in Section 3. The experimental results are presented and discussed in Section 4. The related work is reviewed in Section 5. Finally we conclude this paper in Section 6.

2 The Generative Entity-Mention Model for Entity Linking

In this section we describe the generative entity-mention model. We first describe the generative story of our model, then formulate the model and show how to apply it to the entity linking task.

2.1 The Generative Story

In the entity mention model, each name mention is modeled as a generated sample. For demonstration, Figure 2 shows two examples of name mention generation. As shown in Figure 2, the generative story of a name mention is composed of three steps, which are detailed as follows:

(i) Firstly, the model chooses the referent entity *e* of the name mention from the given knowledge base, according to the distribution of entities in document *P(e)*. In Figure 2, the model chooses the entity "*Michael Jeffrey Jordan*" for the first name mention, and the entity "*Michael I. Jordan*" for the second name mention;

(ii) Secondly, the model outputs the name *s* of the name mention according to the distribution of possible names of the referent entity *P(s|e)*. In Figure 2, the model outputs "*Jordan*" as the name of the entity "*Michael Jeffrey Jordan*", and the "*Michael Jordan*" as the name of the entity "*Michael I. Jordan*";

(iii) Finally, the model outputs the context *c* of the name mention according to the distribution of possible contexts of the referent entity *P(c|e)*. In Figure 2, the model outputs the context "*joins Bulls in 1984*" for the first name mention, and the context "*is a professor in UC Berkeley*" for the second name mention.

2.2 Model

Based on the above generative story, the probability of a name mention *m* (its context is *c* and its name is *s*) referring to a specific entity *e* can be expressed as the following formula (here we assume that *s* and *c* are independent given *e*):

$$P(m,e) = P(s,c,e) = P(e)P(s\,|\,e)P(c\,|\,e)$$

This model incorporates the three types of entity knowledge we explained earlier: *P(e)* corresponds to the popularity knowledge, *P(s|e)* corresponds to the name knowledge and *P(c|e)* corresponds to the context knowledge.

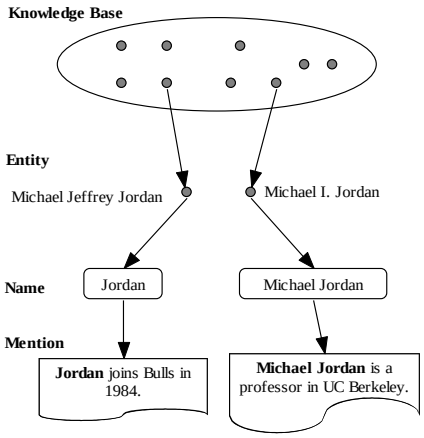


Figure 2. Two examples of name mention generation

Given a name mention *m*, to perform entity linking, we need to find the entity *e* which maximizes the probability *P(e|m)*. Then we can resolve the entity linking task as follows:

$$e = \arg\max_{e} \frac{P(m,e)}{P(m)} = \arg\max_{e} P(e)P(s\,|\,e)P(c\,|\,e)$$

Therefore, the main problem of entity linking is to estimate the three distributions *P(e)*, *P(s|e)* and *P(c|e)*, i.e., to extract the entity knowledge from data. In Section 3, we will show how to estimate these three distributions.

Candidate Selection. Because a knowledge base usually contains millions of entities, it is time-consuming to compute all *P(m,e)* scores between a name mention and all the entities contained in a knowledge base. To reduce the time required, the entity linking system employs a candidate selection process to filter out the impossible referent candidates of a name mention. In this paper, we adopt the candidate selection method of *NLPR_KBP* system (Han and Zhao, 2009), the main idea of which is first building a name-to-entity dictionary using the *redirect links*, *disambiguation pages*, *anchor texts* of Wikipedia, then the candidate entities of a name mention are selected by finding its name's corresponding entry in the dictionary.

3 Model Estimation

Section 2 shows that the entity mention model can decompose the entity linking task into the estimation of three distributions $P(e)$, $P(s|e)$ and $P(c|e)$. In this section, we describe the details of the estimation of these three distributions. We first

introduce the training data, then describe the estimation methods.

3.1 Training Data

In this paper, the training data of our model is a set of annotated name mentions $M = \{m_1, m_2, \ldots, m_n\}$. Each annotated name mention is a triple $m=\{s, e, c\}$, where s is the name, e is the referent entity and c is the context. For example, two annotated name mentions are as follows:

- *Jordan* | **Michael Jeffrey Jordan** | *… wins his first NBA MVP in 1991.*
- *NBA* | **National Basketball Association** | *… is the pre-eminent men's professional basketball league.*

In this paper, we focus on the task of linking entities with Wikipedia, even though the proposed method can be applied to other resources. We will only show how to get the training data from Wikipedia. In Wikipedia, a hyperlink between two articles is an annotated name mention (Milne & Witten, 2008): its anchor text is the name and its target article is the referent entity. For example, in following hyperlink (in Wiki syntax), the *NBA* is the name and the *National Basketball Association* is the referent entity.

"*He won his first* **[[National Basketball Association | NBA]]** *championship with the Bulls*"

Therefore, we can get the training data by collecting all annotated name mentions from the hyperlink data of Wikipedia. In total, we collected more than 23,000,000 annotated name mentions.

3.2 Entity Popularity Model

The distribution $P(e)$ encodes the popularity knowledge as a distribution of entities, i.e., the $P(e_1)$ should be larger than $P(e_2)$ if e_1 is more popular than e_2. For example, on the Web the *P(Michael Jeffrey Jordan)* should be higher than the *P(Michael I. Jordan)*. In this section, we estimate the distribution $P(e)$ using a model called *entity popularity model*.

Given a knowledge base *KB* which contains N entities, in its simplest form, we can assume that all entities have equal popularity, and the distribution $P(e)$ can be estimated as:

$$P(e) = 1/N$$

However, this does not reflect well the real situation because some entities are obviously more popular than others. To get a more precise estimation, we observed that a more popular entity usually appears more times than a less popular entity in a large text corpus, i.e., more name mentions refer to this entity. For example, in Wikipedia the NBA player *Michael Jeffrey Jordan* appears more than 10 times than the Berkeley professor *Michael I. Jordan*. Based on the above observation, our entity popularity model uses the entity frequencies in the name mention data set M to estimate the distribution $P(e)$ as follows:

$$P(e) = \frac{Count(e)+1}{|M|+N}$$

where $Count(e)$ is the count of the name mentions whose referent entity is e, and the $|M|$ is the total name mention size. The estimation is further smoothed using the simple *add-one smoothing* method for the zero probability problem. For illustration, Table 1 shows three selected entities' popularity.

Entity	Popularity
National Basketball Association	$1.73*10^{-5}$
Michael Jeffrey Jordan(NBA player)	$8.21*10^{-6}$
Michael I. Jordan(Berkeley Professor)	$7.50*10^{-8}$

Table 1. Three examples of entity popularity

3.3 Entity Name Model

The distribution $P(s|e)$ encodes the name knowledge of entities, i.e., for a specific entity e, its more frequently used name should be assigned a higher $P(s|e)$ value than the less frequently used name, and a zero $P(s|e)$ value should be assigned to those never used names. For instance, we would expect the *P*(*Michael Jordan*|*Michael Jeffrey Jordan*) to be high, *P*(*MJ*|*Michael Jeffrey Jordan*) to be relative high and *P*(*Michael I. Jordan*|*Michael Jeffrey Jordan*) to be zero.

Intuitively, the name model can be estimated by first collecting all (entity, name) pairs from the name mention data set, then using the maximum likelihood estimation:

$$P(s|e) = \frac{Count(e,s)}{\sum_s Count(e,s)}$$

where the $Count(e,s)$ is the count of the name mentions whose referent entity is e and name is s. However, this method does not work well because it cannot correctly deal with an unseen entity or an unseen name. For example, because the name "*MJ*" doesn't refer to the *Michael Jeffrey Jordan* in Wikipedia, the name model will not be able to identify "*MJ*" as a name of him, even "*MJ*" is a popular name of *Michael Jeffrey Jordan* on Web.

To better estimate the distribution *P(s|e)*, this paper proposes a much more generic model, called *entity name model*, which can capture the variations (including *full name*, *aliases*, *acronyms* and *misspellings*) of an entity's name using a statistical translation model. Given an entity's name *s*, our model assumes that it is a translation of this entity's full name *f* using the IBM model 1 (Brown, et al., 1993). Let Σ be the vocabulary containing all words may be used in the name of entities, the entity name model assumes that a word in Σ can be translated through the following four ways:

1) It is retained (translated into itself);
2) It is translated into its acronym;
3) It is omitted(translated into the word *NULL*);
4) It is translated into another word (misspelling or alias).

In this way, all name variations of an entity are captured as the possible translations of its full name. To illustrate, Figure 3 shows how the full name "*Michael Jeffrey Jordan*" can be transalted into its misspelling name "*Micheal Jordan*".

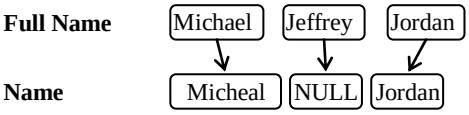


Figure 3. The translation from *Michael Jefferey Jordan* to *Micheal Jordan*

Based on the translation model, *P(s|e)* can be written as:

$$P(s \mid e) = \frac{\varepsilon}{(l_f + 1)^{l_s}} \prod_{j=1}^{l_s} \sum_{i=0}^{l_f} t(s_i \mid f_j)$$

where ε is a normalization factor, *f* is the full name of entity *e*, l_f is the length of *f*, l_s is the length of the name *s*, s_i the i^{th} word of *s*, f_j is the j^{th} word of *f* and $t(s_i|f_j)$ is the lexical translation probability which indicates the probability of a word f_j in the full name will be written as s_i in the output name.

Now the main problem is to estimate the lexical translation probability $t(s_i|f_j)$. In this paper, we first collect the (*name*, *entity full name*) pairs from all annotated name mentions, then get the lexical translation probability by feeding this data set into an IBM model 1 training system (we use the GIZA++ Toolkit[3]).

Table 2 shows several resulting lexical translation probabilities through the above process. We can see that the entity name model can capture the different name variations, such as the acronym (*Michael*→*M*), the misspelling (*Michael*→*Micheal*) and the omission (*St.* → *NULL*).

Full name word	Name word	Probability
Michael	*Michael*	*0.77*
Michael	*M*	*0.008*
Michael	*Micheal*	$2.64*10^{-4}$
Jordan	*Jordan*	*0.96*
Jordan	*J*	$6.13*10^{-4}$
St.	*NULL*	*0.14*
Sir	*NULL*	*0.02*

Table 2. Several lexical translation probabilities

3.4 Entity Context Model

The distribution *P(c|e)* encodes the context knowledge of entities, i.e., it will assign a high *P(c|e)* value if the entity *e* frequently appears in the context *c*, and will assign a low *P(c|e)* value if the entity *e* rarely appears in the context *c*. For example, given the following two contexts:

C1: *__wins NBA MVP.*

C2: *__is a researcher in machine learning.*

Then *P(C1|Michael Jeffrey Jordan)* should be high because the NBA player *Michael Jeffrey Jordan* often appears in C1 and the *P(C2|Michael Jeffrey Jordan)* should be extremely low because he rarely appears in C2.

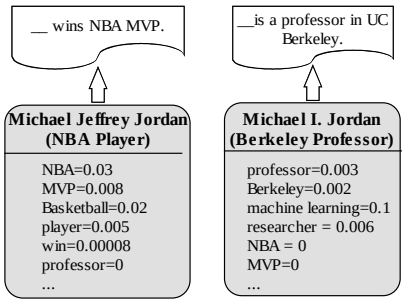


Figure 4. Two entity context models

To estimate the distribution *P(c|e)*, we propose a method based on language modeling, called *entity context model*. In our model, the context of each name mention *m* is the word window surrounding *m*, and the window size is set to 50 according to the experiments in (Pedersen et al., 2005). Specifically, the context knowledge of an entity *e* is encoded in an unigram language model:

$$M_e = \{P_e(t)\}$$

where $P_e(t)$ is the probability of the term *t* appearing in the context of *e*. In our model, the term may indicate a word, a named entity (extracted using the *Stanford Named Entity*

[3] http://fjoch.com/GIZA++.html

Recognizer[4]) or a Wikipedia concept (extracted using the method described in (Han and Zhao, 2010)). Figure 4 shows two entity context models and the contexts generated using them.

Now, given a context c containing n terms $t_1t_2...t_n$, the entity context model estimates the probability $P(c|e)$ as:

$$P(c \mid e) = P(t_1 t_2 ... t_n \mid M_e) = P_e(t_1)P_e(t_2)....P_e(t_n)$$

So the main problem is to estimate $P_e(t)$, the probability of a term t appearing in the context of the entity e.

Using the annotated name mention data set M, we can get the maximum likelihood estimation of $P_e(t)$ as follows:

$$P_{e\_ML}(t) = \frac{Count_e(t)}{\sum_t Count_e(t)}$$

where $Count_e(t)$ is the frequency of occurrences of a term t in the contexts of the name mentions whose referent entity is e.

Because an entity e's name mentions are usually not enough to support a robust estimation of $P_e(t)$ due to the sparse data problem (Chen and Goodman, 1999), we further smooth $P_e(t)$ using the Jelinek-Mercer smoothing method (Jelinek and Mercer, 1980):

$$P_e(t) = \lambda P_{e\_ML}(t) + (1-\lambda)P_g(t)$$

where $P_g(t)$ is a general language model which is estimated using the whole Wikipedia data, and the optimal value of λ is set to 0.2 through a learning process shown in Section 4.

3.5 The NIL Entity Problem

By estimating $P(e)$, $P(s|e)$ and $P(c|e)$, our method can effectively link a name mention to its referent entity contained in a knowledge base. Unfortunately, there is still the *NIL entity problem* (McNamee and Dang, 2009), i.e., the referent entity may not be contained in the given knowledge base. In this situation, the name mention should be linked to the NIL entity. Traditional methods usually resolve this problem with an additional classification step (Zheng et al. 2010): a classifier is trained to identify whether a name mention should be linked to the NIL entity.

Rather than employing an additional step, our entity mention model seamlessly takes into account the NIL entity problem. The start assumption of our solution is that "If a name mention refers to a specific entity, then the probability of this name mention is generated by the specific entity's model should be significantly higher than the probability it is generated by a general language model". Based on the above assumption, we first add a pseudo entity, the NIL entity, into the knowledge base and assume that the NIL entity generates a name mention according to the general language model P_g, without using any entity knowledge; then we treat the NIL entity in the same way as other entities: if the probability of a name mention is generated by the NIL entity is higher than all other entities in Knowledge base, we link the name mention to the NIL entity. Based on the above discussion, we compute the three probabilities of the NIL entity: $P(e)$, $P(s|e)$ and $P(c|e)$ as follows:

$$P(NIL) = \frac{1}{|M| + N}$$

$$P(s \mid NIL) = \prod_{t \in s} P_g(t)$$

$$P(c \mid NIL) = \prod_{t \in c} P_g(t)$$

4 Experiments

In this section, we assess the performance of our method and compare it with the traditional methods. In following, we first explain the experimental settings in Section 4.1, 4.2 and 4.3, then evaluate and discuss the results in Section 4.4.

4.1 Knowledge Base

In our experiments, we use the Jan. 30, 2010 English version of Wikipedia as the knowledge base, which contains over 3 million distinct entities.

4.2 Data Sets

To evaluate the entity linking performance, we adopted two data sets: the first is *WikiAmbi*, which is used to evaluate the performance on Wikipedia articles; the second is *TAC_KBP*, which is used to evaluate the performance on general newswire documents. In following, we describe these two data sets in detail.

WikiAmbi: The *WikiAmbi* data set contains 1000 annotated name mentions which are randomly selected from Wikipedia hyperlinks data set (as shown in Section 3.1, the hyperlinks between Wikipedia articles are manually annotated name mentions). In *WikiAmbi*, there were 207 distinct

[4] http://nlp.stanford.edu/software/CRF-NER.shtml

names and each name contains at least two possible referent entities (on average 6.7 candidate referent entities for each name) [5]. In our experiments, the name mentions contained in the *WikiAmbi* are removed from the training data.

TAC_KBP: The *TAC_KBP* is the standard data set used in the Entity Linking task of the TAC 2009 (McNamee and Dang, 2009). The *TAC_KBP* contains 3904 name mentions which are selected from English newswire articles. For each name mention, its referent entity in Wikipedia is manually annotated. Overall, 57% (2229 of 3904) name mentions's referent entities are missing in Wikipedia, so *TAC_KBP* is also suitable to evaluate the NIL entity detection performance.

The above two data sets can provide a standard testbed for the entity linking task. However, there were still some limitations of these data sets: First, these data sets only annotate the salient name mentions in a document, meanwhile many NLP applications need all name mentions are linked. Second, these data sets only contain well-formed documents, but in many real-world applications the entity linking often be applied to noisy documents such as product reviews and microblog messages. In future, we want to develop a data set which can reflect these real-world settings.

4.3 Evaluation Criteria

We adopted the standard performance metrics used in the Entity Linking task of the TAC 2009 (McNamee and Dang, 2009). These metrics are:

- Micro-Averaged Accuracy (**Micro-Accuracy**): measures entity linking accuracy averaged over all the name mentions;
- Macro-Averaged Accuracy (**Macro-Accuracy)**: measures entity linking accuracy averaged over all the target entities.

As in TAC 2009, we used **Micro-Accuracy** as the primary performance metric.

4.4 Experimental Results

We compared our method with three baselines: (1) The first is the traditional *Bag of Words* based method (Cucerzan, 2007): a name mention's referent entity is the entity which has the highest cosine similarity with its context – we denoted it as *BoW*; (2) The second is the method described in (Medelyan et al., 2008), where a name mention's referent entity is the entity which has the largest average semantic relatedness with the name mention's unambiguous context entities – we denoted it as *TopicIndex*. (3) The third one is the same as the method described in (Milne & Witten, 2008), which uses learning techniques to balance the semantic relatedness, commoness and context quality – we denoted it as *Learning2Link.*

4.4.1 Overall Performance

We conduct experiments on both ***WikiAmbi*** and ***TAC_KBP*** datasets with several methods: the baseline *BoW*; the baseline *TopicIndex;* the baseline *Learning2Link*; the proposed method using only popularity knowledge (***Popu***), i.e., the $P(m,e)=P(e)$; the proposed method with one component of the model is ablated(this is used to evaluate the independent contributions of the three components), correspondingly ***Popu+Name***(i.e., the $P(m,e)=P(e)P(s|e)$), ***Name+Context***(i.e., the $P(m,e)=P(c|e)P(s|e)$) and ***Popu+Context*** (i.e., the $P(m,e)=P(e)P(c|e)$); and the full entity mention model (***Full Model***). For all methods, the parameters were configured through 10-fold cross validation. The overall performance results are shown in Table 3 and 4.

	Micro-Accuracy	**Macro-Accuracy**
BoW	*0.60*	*0.61*
TopicIndex	*0.66*	*0.49*
Learning2Link	*0.70*	*0.54*
Popu	*0.39*	*0.24*
Popu + Name	*0.50*	*0.31*
Name+Context	*0.70*	*0.68*
Popu+Context	*0.72*	*0.73*
Full Model	***0.80***	***0.77***

Table 3. The overall results on *WikiAmbi* dataset

	Micro-Accuracy	**Macro-Accuracy**
BoW	*0.72*	*0.75*
TopicIndex	*0.80*	*0.76*
Learning2Link	*0.83*	*0.79*
Popu	*0.60*	*0.53*
Popu + Name	*0.63*	*0.59*
Name+Context	*0.81*	*0.78*
Popu+Context	*0.84*	*0.83*
Full Model	***0.86***	***0.88***

Table 4. The overall results on *TAC-KBP* dataset

From the results in Table 3 and 4, we can make the following observations:

1) Compared with the traditional methods, our entity mention model can achieve a significant

[5] This is because we want to create a highly ambiguous test data set

performance improvement: In *WikiAmbi* and *TAC_KBP* datasets, compared with the *BoW* baseline, our method respectively gets 20% and 14% micro-accuracy improvement; compared with the *TopicIndex* baseline, our method respectively gets 14% and 6% micro-accuracy improvement; compared with the *Learning2Link* baseline, our method respectively gets 10% and 3% micro-accuracy improvement.

2) By incorporating more entity knowledge, our method can significantly improve the entity linking performance: When only using the popularity knowledge, our method can only achieve 49.5% micro-accuracy. By adding the name knowledge, our method can achieve 56.5% micro-accuracy, a 7% improvement over the *Popu*. By further adding the context knowledge, our method can achieve 83% micro-accuracy, a 33.5% improvement over *Popu* and a 26.5% improvement over *Popu+Name*.

3) All three types of entity knowledge contribute to the final performance improvement, and the context knowledge contributes the most: By respectively ablating the popularity knowledge, the name knowledge and the context knowledge, the performance of our model correspondingly reduces 7.5%, 5% and 26.5%.

NIL Entity Detection Performance. To compare the performances of resolving the NIL entity problem, Table 5 shows the micro-accuracies of different systems on the *TAC_KBP* data set (where **All** is the whole data set, **NIL** only contains the name mentions whose referent entity is NIL, **InKB** only contains the name mentions whose referent entity is contained in the knowledge base). From Table 5 we can see that our method can effectively detect the NIL entity meanwhile retaining the high InKB accuracy.

	All	**NIL**	**InKB**
BoW	*0.72*	*0.77*	*0.65*
TopicIndex	*0.80*	***0.91***	*0.65*
Learning2Link	*0.83*	***0.90***	*0.73*
Full Model	***0.86***	***0.90***	***0.79***

Table 5. The NIL entity detection performance on the *TAC_KBP* data set

4.4.2 Optimizing Parameters

Our model needs to tune one parameter: the Jelinek-Mercer smoothing parameter λ used in the entity context model. Intuitively, a smaller λ means that the general language model plays a more important role. Figure 5 plots the tradeoff. In both *WikiAmbi* and *TAC_KBP* data sets, Figure 5 shows that a λ value 0.2 will result in the best performance.

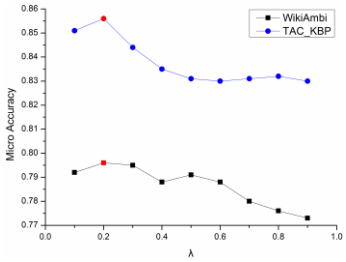


Figure 5. The micro-accuracy vs. λ

4.4.3 Detailed Analysis

To better understand the reasons why and how the proposed method works well, in this Section we analyze our method in detail.

The Effect of Incorporating Heterogenous Entity Knowledge. The first advantage of our method is the entity mention model can incorporate heterogeneous entity knowledge. The Table 3 and 4 have shown that, by incorporating heterogenous entity knowledge (including the name knowledge, the popularity knowledge and the context knowledge), the entity linking performance can obtain a significant improvement.

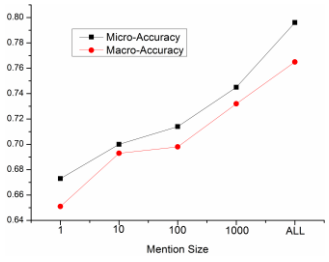


Figure 6. The performance vs. training mention size on *WikiAmbi* data set

The Effect of Better Entity Knowledge Extraction. The second advantage of our method is that, by representing the entity knowledge as probabilistic distributions, our model has a statistical foundation and can better extract the entity knowledge using more training data through the *entity popularity model*, the *entity name model* and the *entity context model*. For instance, we can train a better entity context model *P(c|e)* using more name mentions. To find whether a better

entity knowledge extraction will result in a better performance, Figure 6 plots the micro-accuray along with the size of the training data on name mentions for *P(c|e)* of each entity *e*. From Figure 6, we can see that when more training data is used, the performance increases.

4.4.4 Comparision with State-of-the-Art Performance

We also compared our method with the state-of-the-art entity linking systems in the TAC 2009 KBP track (McNamee and Dang, 2009). Figure 7 plots the comparison with the top five performances in TAC 2009 KBP track. From Figure 7, we can see that our method can outperform the state-of-the-art approaches: compared with the best ranking system, our method can achieve a 4% performance improvement.

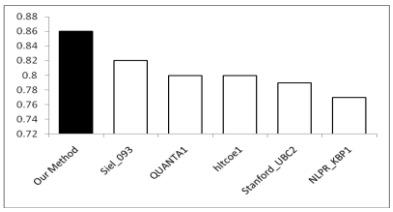


Figure 7. A comparison with top 5 TAC 2009 KBP systems

5 Related Work

In this section, we briefly review the related work. To the date, most entity linking systems employed the context similarity based methods. The essential idea was to extract the discriminative features of an entity from its description, then link a name mention to the entity which has the largest context similarity with it. Cucerzan (2007) proposed a *Bag of Words* based method, which represents each target entity as a vector of terms, then the similarity between a name mention and an entity was computed using the cosine similarity measure. Mihalcea & Csomai (2007), Bunescu & Pasca (2006), Fader et al. (2009) extended the *BoW* model by incorporating more entity knowledge such as popularity knowledge, entity category knowledge, etc. Zheng et al. (2010), Dredze et al. (2010), Zhang et al. (2010) and Zhou et al. (2010) employed the learning to rank techniques which can further take the relations between candidate entities into account. Because the context similarity based methods can only represent the entity knowledge as features, the main drawback of it was the difficulty to incorporate heterogenous entity knowledge.

Recently there were also some entity linking methods based on inter-dependency. These methods assumed that the entities in the same document are related to each other, thus the referent entity of a name mention is the entity which is most related to its contextual entities. Medelyan et al. (2008) found the referent entity of a name mention by computing the weighted average of semantic relatedness between the candidate entity and its unambiguous contextual entities. Milne and Witten (2008) extended Medelyan et al. (2008) by adopting learning-based techniques to balance the semantic relatedness, commoness and context quality. Kulkarni et al. (2009) proposed a method which collectively resolves the entity linking tasks in a document as an optimization problem. The drawback of the inter-dependency based methods is that they are usually specially designed to the leverage of semantic relations, doesn't take the other types of entity knowledge into consideration.

6 Conclusions and Future Work

This paper proposes a generative probabilistic model, the *entity-mention model*, for the entity linking task. The main advantage of our model is it can incorporate multiple types of heterogenous entity knowledge. Furthermore, our model has a statistical foundation, making the entity knowledge extraction approach different from most previous ad hoc approaches. Experimental results show that our method can achieve competitive performance.

In our method, we did not take into account the dependence between entities in the same document. This aspect could be complementary to those we considered in this paper. For our future work, we can integrate such dependencies in our model.

Acknowledgments

The work is supported by the National Natural Science Foundation of China under Grants no. 60773027, 60736044, 90920010, 61070106 and 61003117, and the National High Technology Development 863 Program of China under Grants no. 2008AA01Z145. Moreover, we sincerely thank the reviewers for their valuable comments.

References

Adafre, S. F. & de Rijke, M. 2005. *Discovering missing links in Wikipedia*. In: Proceedings of the 3rd international workshop on Link discovery.

Bunescu, R. & Pasca, M. 2006. *Using encyclopedic knowledge for named entity disambiguation*. In: Proceedings of EACL, vol. 6.

Brown, P., Pietra, S. D., Pietra, V. D., and Mercer, R. 1993. *The mathematics of statistical machine translation: parameter estimation*. Computational Linguistics, 19(2), 263-31.

Chen, S. F. & Goodman, J. 1999. *An empirical study of smoothing techniques for language modeling*. In Computer Speech and Language, London; Orlando: Academic Press, c1986-, pp. 359-394.

Cucerzan, S. 2007. *Large-scale named entity disambiguation based on Wikipedia data*. In: Proceedings of EMNLP-CoNLL, pp. 708-716.

Dredze, M., McNamee, P., Rao, D., Gerber, A. & Finin, T. 2010. *Entity Disambiguation for Knowledge Base Population*. In: Proceedings of the 23rd International Conference on Computational Linguistics.

Fader, A., Soderland, S., Etzioni, O. & Center, T. 2009. *Scaling Wikipedia-based named entity disambiguation to arbitrary web text*. In: Proceedings of Wiki-AI Workshop at IJCAI, vol. 9.

Han, X. & Zhao, J. 2009. *NLPR_KBP in TAC 2009 KBP Track: A Two-Stage Method to Entity Linking*. In: Proceeding of Text Analysis Conference.

Han, X. & Zhao, J. 2010. *Structural semantic relatedness: a knowledge-based method to named entity disambiguation*. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics.

Jelinek, Frederick and Robert L. Mercer. 1980. *Interpolated estimation of Markov source parameters from sparse data*. In: Proceedings of the Workshop on Pattern Recognition in Practice.

Kulkarni, S., Singh, A., Ramakrishnan, G. & Chakrabarti, S. 2009. *Collective annotation of Wikipedia entities in web text*. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 457-466.

Li, X., Morie, P. & Roth, D. 2004. *Identification and tracing of ambiguous names: Discriminative and generative approaches*. In: *Proceedings of the National Conference on Artificial Intelligence*, pp. 419-424.

McNamee, P. & Dang, H. T. 2009. *Overview of the TAC 2009 Knowledge Base Population Track*. In: Proceeding of Text Analysis Conference.

Milne, D. & Witten, I. H. 2008. *Learning to link with Wikipedia*. In: Proceedings of the 17th ACM conference on Conference on information and knowledge management.

Milne, D., et al. 2006. *Mining Domain-Specific Thesauri from Wikipedia: A case study*. In Proc. of IEEE/WIC/ACM WI.

Medelyan, O., Witten, I. H. & Milne, D. 2008. *Topic indexing with Wikipedia*. In: Proceedings of the AAAI WikiAI workshop.

Mihalcea, R. & Csomai, A. 2007. *Wikify!: linking documents to encyclopedic knowledge*. In: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, pp. 233-242.

Pedersen, T., Purandare, A. & Kulkarni, A. 2005. *Name discrimination by clustering similar contexts*. Computational Linguistics and Intelligent Text Processing, pp. 226-237.

Zhang, W., Su, J., Tan, Chew Lim & Wang, W. T. 2010. *Entity Linking Leveraging Automatically Generated Annotation*. In: Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010).

Zheng, Z., Li, F., Huang, M. & Zhu, X. 2010. *Learning to Link Entities with Knowledge Base*. In: The Proceedings of the Annual Conference of the North American Chapter of the ACL.

Zhou, Y., Nie, L., Rouhani-Kalleh, O., Vasile, F. & Gaffney, S. 2010. *Resolving Surface Forms to Wikipedia Topics*. In: Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pp. 1335-1343.

Simple Supervised Document Geolocation with Geodesic Grids

Benjamin P. Wing
Department of Linguistics
University of Texas at Austin
Austin, TX 78712 USA
ben@benwing.com

Jason Baldridge
Department of Linguistics
University of Texas at Austin
Austin, TX 78712 USA
jbaldrid@mail.utexas.edu

Abstract

We investigate automatic *geolocation* (i.e. identification of the location, expressed as latitude/longitude coordinates) of documents. Geolocation can be an effective means of summarizing large document collections and it is an important component of geographic information retrieval. We describe several simple supervised methods for document geolocation using only the document's raw text as evidence. All of our methods predict locations in the context of geodesic grids of varying degrees of resolution. We evaluate the methods on geotagged Wikipedia articles and Twitter feeds. For Wikipedia, our best method obtains a median prediction error of just 11.8 kilometers. Twitter geolocation is more challenging: we obtain a median error of 479 km, an improvement on previous results for the dataset.

1 Introduction

There are a variety of applications that arise from connecting linguistic content—be it a word, phrase, document, or entire corpus—to geography. Leidner (2008) provides a systematic overview of geography-based language applications over the previous decade, with a special focus on the problem of *toponym resolution*—identifying and disambiguating the references to locations in texts. Perhaps the most obvious and far-reaching application is geographic information retrieval (Ding et al., 2000; Martins, 2009; Andogah, 2010), with applications like MetaCarta's geographic text search (Rauch et al., 2003) and NewsStand (Teitler et al., 2008); these allow users to browse and search for content through a geo-centric interface. The Perseus project performs automatic toponym resolution on historical texts in order to display a map with each text showing the locations that are mentioned (Smith and Crane, 2001); Google Books also does this for some books, though the toponyms are identified and resolved quite crudely. Hao et al (2010) use a location-based topic model to summarize travelogues, enrich them with automatically chosen images, and provide travel recommendations. Eisenstein et al (2010) investigate questions of dialectal differences and variation in regional interests in Twitter users using a collection of geotagged tweets.

An intuitive and effective strategy for summarizing geographically-based data is identification of the location—a specific latitude and longitude—that forms the primary focus of each document. Determining a *single* location of a document is only a well-posed problem for certain documents, generally of fairly small size, but there are a number of natural situations in which such collections arise. For example, a great number of articles in Wikipedia have been manually geotagged; this allows those articles to appear in their geographic locations while geobrowsing in an application like Google Earth.

Overell (2009) investigates the use of Wikipedia as a source of data for article geolocation, in addition to article classification by category (location, person, etc.) and toponym resolution. Overell's main goal is toponym resolution, for which geolocation serves as an input feature. For document geolocation, Overell uses a simple model that makes use only of the metadata available (article title, incoming and outgoing links, etc.)—the actual article text

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 955–964,
Portland, Oregon, June 19-24, 2011.

is not used at all. However, for many document collections, such metadata is unavailable, especially in the case of recently digitized historical documents.

Eisenstein et al. (2010) evaluate their geographic topic model by geolocating USA-based Twitter users based on their tweet content. This is essentially a document geolocation task, where each document is a concatenation of all the tweets for a single user. Their geographic topic model receives supervision from many documents/users and predicts locations for unseen documents/users.

In this paper, we tackle document geolocation using several simple supervised methods on the textual content of documents and a geodesic grid as a discrete representation of the earth's surface. Our approach is similar to that of Serdyukov et al. (2009), who geolocate Flickr images using their associated textual tags.[1] Essentially, the task is cast similarly to language modeling approaches in information retrieval (Ponte and Croft, 1998). Discrete cells representing areas on the earth's surface correspond to documents (with each cell-document being a concatenation of all actual documents that are located in that cell); new documents are then geolocated to the most similar cell according to standard measures such as Kullback-Leibler divergence (Zhai and Lafferty, 2001). Performance is measured both on geotagged Wikipedia articles (Overell, 2009) and tweets (Eisenstein et al., 2010). We obtain high accuracy on Wikipedia using KL divergence, with a median error of just 11.8 kilometers. For the Twitter data set, we obtain a median error of 479 km, which improves on the 494 km error of Eisenstein et al. An advantage of our approach is that it is far simpler, is easy to implement, and scales straightforwardly to large datasets like Wikipedia.

2 Data

Wikipedia As of April 15, 2011, Wikipedia has some 18.4 million content-bearing articles in 281 language-specific encyclopedias. Among these, 39 have over 100,000 articles, including 3.61 million articles in the English-language edition alone. Wikipedia articles generally cover a single subject; in addition, most articles that refer to geographically fixed subjects are *geotagged* with their coordinates. Such articles are well-suited as a source of supervised content for document geolocation purposes. Furthermore, the existence of versions in multiple languages means that the techniques in this paper can easily be extended to cover documents written in many of the world's most common languages.

Wikipedia's geotagged articles encompass more than just cities, geographic formations and landmarks. For example, articles for events (like the shooting of JFK) and vehicles (such as the frigate USS *Constitution*) are geotagged. The latter type of article is actually quite challenging to geolocate based on the text content: though the ship is moored in Boston, most of the page discusses its role in various battles along the eastern seaboard of the USA. However, such articles make up only a small fraction of the geotagged articles.

For the experiments in this paper, we used a full dump of Wikipedia from September 4, 2010.[2] Included in this dump is a total of 10,355,226 articles, of which 1,019,490 have been geotagged. Excluding various types of special-purpose articles used primarily for maintaining the site (specifically, redirect articles and articles outside the main namespace), the dump includes 3,431,722 content-bearing articles, of which 488,269 are geotagged.

It is necessary to process the raw dump to obtain the plain text, as well as metadata such as geotagged coordinates. Extracting the coordinates, for example, is not a trivial task, as coordinates can be specified using multiple templates and in multiple formats. Automatically-processed versions of the English-language Wikipedia site are provided by Metaweb,[3] which at first glance promised to significantly simplify the preprocessing. Unfortunately, these versions still need significant processing and they incorrectly eliminate some of the important metadata. In the end, we wrote our own code to process the raw dump. It should be possible to extend this code to handle other languages with little difficulty. See Lieberman and Lin (2009) for more discussion of a related effort to extract and use the geotagged articles in Wikipedia.

The entire set of articles was split 80/10/10 in

[1] We became aware of Serdyukov et al. (2009) during the writing of the camera-ready version of this paper.

[2] `http://download.wikimedia.org/enwiki/20100904/pages-articles.xml.bz2`

[3] `http://download.freebase.com/wex/`

round-robin fashion into training, development, and testing sets after randomizing the order of the articles, which preserved the proportion of geotagged articles. Running on the full data set is time-consuming, so development was done on a subset of about 80,000 articles (19.9 million tokens) as a training set and 500 articles as a development set. Final evaluation was done on the full dataset, which includes 390,574 training articles (97.2 million tokens) and 48,589 test articles. A full run with all the six strategies described below (three baseline, three non-baseline) required about 4 months of computing time and about 10-16 GB of RAM when run on a 64-bit Intel Xeon E5540 CPU; we completed such jobs in under two days (wall clock) using the Longhorn cluster at the Texas Advanced Computing Center.

Geo-tagged Microblog Corpus As a second evaluation corpus on a different domain, we use the corpus of geotagged tweets collected and used by Eisenstein et al. (2010).[4] It contains 380,000 messages from 9,500 users tweeting within the 48 states of the continental USA.

We use the train/dev/test splits provided with the data; for these, the tweets of each user (a feed) have been concatenated to form a single document, and the location label associated with each document is the location of the first tweet by that user. This is generally a fair assumption as Twitter users typically tweet within a relatively small region. Given this setup, we will refer to Twitter users as documents in what follows; this keeps the terminology consistent with Wikipedia as well. The training split has 5,685 documents (1.58 million tokens).

Replication Our code (part of the TextGrounder system), our processed version of Wikipedia, and instructions for replicating our experiments are available on the TextGrounder website.[5]

3 Grid representation for connecting texts to locations

Geolocation involves identifying some spatial region with a unit of text—be it a word, phrase, or document. The earth's surface is continuous, so a natural approach is to predict locations using a continuous distribution. For example, Eisenstein et al. (2010) use Gaussian distributions to model the locations of Twitter users in the United States of America. This appears to work reasonably well for that restricted region, but is likely to run into problems when predicting locations for anywhere on earth—instead, spherical distributions like the von Mises-Fisher distribution would need to be employed.

We take here the simpler alternative of discretizing the earth's surface with a geodesic grid; this allows us to predict locations with a variety of standard approaches over discrete outcomes. There are many ways of constructing geodesic grids. Like Serdyukov et al. (2009), we use the simplest strategy: a grid of square cells of *equal degree*, such as 1° by 1°. This produces variable-size regions that shrink latitudinally, becoming progressively smaller and more elongated the closer they get towards the poles. Other strategies, such as the quaternary triangular mesh (Dutton, 1996), preserve *equal area*, but are considerably more complex to implement. Given that most of the populated regions of interest for us are closer to the equator than not and that we use cells of quite fine granularity (down to 0.05°), the simple grid system was preferable.

With such a discrete representation of the earth's surface, there are four distributions that form the core of all our geolocation methods. The first is a standard multinomial distribution over the vocabulary for every cell in the grid. Given a grid G with cells c_i and a vocabulary V with words w_j, we have $\theta_{c_i j} = P(w_j|c_i)$. The second distribution is the equivalent distribution for a single test document d_k, i.e. $\theta_{d_k j} = P(w_j|d_k)$. The third distribution is the reverse of the first: for a given word, its distribution over the earth's cells, $\kappa_{ji} = P(c_i|w_j)$. The final distribution is over the cells, $\gamma_i = P(c_i)$.

This grid representation ignores all higher level regions, such as states, countries, rivers, and mountain ranges, but it is consistent with the geocoding in both the Wikipedia and Twitter datasets. Nonetheless, note that the κ_{ji} for words referring to such regions is likely to be much flatter (spread out) but with most of the mass concentrated in a set of connected cells. Those for highly focused point-locations will jam up in a few disconnected cells—in the extreme case, toponyms like *Spring-*

[4] http://www.ark.cs.cmu.edu/GeoText/

[5] http://code.google.com/p/textgrounder/wiki/WingBaldridge2011

field which are connected to many specific point locations around the earth.

We use grids with cell sizes of varying granularity $d \times d$ for $d = 0.1^\circ, 0.5^\circ, 1^\circ, 5^\circ, 10^\circ$. For example, with d=0.5°, a cell at the equator is roughly 56x55 km and at 45° latitude it is 39x55 km. At this resolution, there are a total of 259,200 cells, of which 35,750 are non-empty when using our Wikipedia training set. For comparison, at the equator a cell at d=5° is about 557x553 km (2,592 cells; 1,747 non-empty) and at d=0.1° a cell is about 11.3x10.6 km (6,480,000 cells; 170,005 non-empty).

The geolocation methods predict a cell $\hat{c}$ for a document, and the latitude and longitude of the degree-midpoint of the cell is used as the predicted location. Prediction error is the great-circle distance from these predicted locations to the locations given by the gold standard. The use of cell midpoints provides a fair comparison for predictions with different cell sizes. This differs from the evaluation metrics used by Serdyukov et al. (2009), which are all computed relative to a given grid size. With their metrics, results for different granularities cannot be directly compared because using larger cells means less ambiguity when choosing $\hat{c}$. With our distance-based evaluation, large cells are penalized by the distance from the midpoint to the actual location even when that location is in the same cell. Smaller cells reduce this penalty and permit the word distributions $\theta_{c_i j}$ to be much more specific for each cell, but they are harder to predict exactly and suffer more from sparse word counts compared to courser granularity. For large datasets like Wikipedia, fine-grained grids work very well, but the trade-off between resolution and sufficient training material shows up more clearly for the smaller Twitter dataset.

4 Supervised models for document geolocation

Our methods use only the text in the documents; predictions are made based on the distributions θ, κ, and ρ introduced in the previous section. No use is made of metadata, such as links/followers and infoboxes.

4.1 Supervision

We acquire θ and κ straightforwardly from the training material. The unsmoothed estimate of word w_j's probability in a test document d_k is:[6]

$$\tilde{\theta}_{d_k j} = \frac{\#(w_j, d_k)}{\sum_{w_l \in V} \#(w_l, d_k)} \quad (1)$$

Similarly for a cell c_i, we compute the unsmoothed word distribution by aggregating all of the documents located within c_i:

$$\tilde{\theta}_{c_i j} = \frac{\sum_{d_k \in c_i} \#(w_j, d_k)}{\sum_{d_k \in c_i} \sum_{w_l \in V} \#(w_l, d_k)} \quad (2)$$

We compute the global distribution θ_{Dj} over the set of all documents D in the same fashion.

The word distribution of document d_k backs off to the global distribution θ_{Dj}. The probability mass α_{d_k} reserved for unseen words is determined by the empirical probability of having seen a word once in the document, motivated by Good-Turing smoothing. (The cell distributions are treated analogously.) That is:[7]

$$\alpha_{d_k} = \frac{|w_j \in V \, s.t. \, \#(w_j, d_k)=1|}{\sum_{w_j \in V} \#(w_j, d_k)} \quad (3)$$

$$\theta_{Dj}^{(-d_k)} = \frac{\theta_{Dj}}{1 - \sum_{w_l \in d_k} \theta_{Dl}} \quad (4)$$

$$\theta_{d_k j} = \begin{cases} \alpha_{d_k} \theta_{Dj}^{(-d_k)}, & \text{if } \tilde{\theta}_{d_k j} = 0 \\ (1-\alpha_{d_k}) \tilde{\theta}_{d_k j}, & \text{o.w.} \end{cases} \quad (5)$$

The distributions over cells for each word simply renormalizes the $\theta_{c_i j}$ values to achieve a proper distribution:

$$\kappa_{ji} = \frac{\theta_{c_i j}}{\sum_{c_i \in G} \theta_{c_i j}} \quad (6)$$

A useful aspect of the κ distributions is that they can be plotted in a geobrowser using thematic mapping

[6] We use $\#()$ to indicate the count of an event.

[7] $\theta_{Dj}^{(-d_k)}$ is an adjusted version of θ_{Dj} that is normalized over the subset of words not found in document d_k. This adjustment ensures that the entire distribution is properly normalized.

techniques (Sandvik, 2008) to inspect the spread of a word over the earth. We used this as a simple way to verify the basic hypothesis that words that do not name locations are still useful for geolocation. Indeed, the Wikipedia distribution for *mountain* shows high density over the Rocky Mountains, Smokey Mountains, the Alps, and other ranges, while *beach* has high density in coastal areas. Words without inherent locational properties also have intuitively correct distributions: e.g., *barbecue* has high density over the south-eastern United States, Texas, Jamaica, and Australia, while *wine* is concentrated in France, Spain, Italy, Chile, Argentina, California, South Africa, and Australia.[8]

Finally, the cell distributions are simply the relative frequency of the number of documents in each cell: $\gamma_i = \frac{|c_i|}{|D|}$.

A standard set of stop words are ignored. Also, all words are lowercased except in the case of the most-common-toponym baselines, where uppercase words serve as a fallback in case a toponym cannot be located in the article.

4.2 Kullback-Leibler divergence

Given the distributions for each cell, θ_{c_i}, in the grid, we use an information retrieval approach to choose a location for a test document d_k: compute the similarity between its word distribution θ_{d_k} and that of each cell, and then choose the closest one. Kullback-Leibler (KL) divergence is a natural choice for this (Zhai and Lafferty, 2001). For distribution P and Q, KL divergence is defined as:

$$KL(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{7}$$

This quantity measures how good Q is as an encoding for P – the smaller it is the better. The best cell $\hat{c}_{KL}$ is the one which provides the best encoding for the test document:

$$\hat{c}_{KL} = \arg\min_{c_i \in G} KL(\theta_{d_k}||\theta_{c_i}) \tag{8}$$

The fact that KL is not symmetric is desired here: the other direction, $KL(\theta_{c_i}||\theta_{d_k})$, asks which cell the test document is a good encoding for. With $KL(\theta_{d_k}||\theta_{c_i})$, the log ratio of probabilities for each word is weighted by the probability of the word in the test document, $\boldsymbol{\theta_{d_k j}} \log \frac{\theta_{d_k j}}{\theta_{c_i j}}$, which means that the divergence is more sensitive to the document rather than the overall cell.

As an example for why non-symmetric KL in this order is appropriate, consider geolocating a page in a densely geotagged cell, such as the page for the Washington Monument. The distribution of the cell containing the monument will represent the words from many other pages having to do with museums, US government, corporate buildings, and other nearby memorials and will have relatively small values for many of the words that are highly indicative of the monument's location. Many of those words appear only once in the monument's page, but this will still be a higher value than for the cell and will weight the contribution accordingly.

Rather than computing $KL(\theta_{d_k}||\theta_{c_i})$ over the entire vocabulary, we restrict it to only the words in the document to compute KL more efficiently:

$$KL(\theta_{d_k}||\theta_{c_i}) = \sum_{w_j \in V_{d_k}} \theta_{d_k j} \log \frac{\theta_{d_k j}}{\theta_{c_i j}} \tag{9}$$

Early experiments showed that it makes no difference in the outcome to include the rest of the vocabulary. Note that because θ_{c_i} is smoothed, there are no zeros, so this value is always defined.

4.3 Naive Bayes

Naive Bayes is a natural generative model for the task of choosing a cell, given the distributions θ_{c_i} and γ: to generate a document, choose a cell c_i according to γ and then choose the words in the document according to θ_{c_i}:

$$\begin{aligned}
\hat{c}_{NB} &= \arg\max_{c_i \in G} P_{NB}(c_i|d_k) \\
&= \arg\max_{c_i \in G} \frac{P(c_i)P(d_k|c_i)}{P(d_k)} \\
&= \arg\max_{c_i \in G} \gamma_i \prod_{w_j \in V_{d_k}} \theta_{c_i j}^{\#(w_j, d_k)}
\end{aligned} \tag{10}$$

[8] This also acts as an exploratory tool. For example, due to a big spike on Cebu Province in the Philippines we learned that Cebuanos take barbecue very, very seriously.

This method maximizes the combination of the *likelihood* of the document $P(d_k|c_i)$ and the cell prior probability γ_i.

4.4 Average cell probability

For each word, κ_{ji} gives the probability of each cell in the grid. A simple way to compute a distribution for a document d_k is to take a weighted average of the distributions for all words to compute the average cell probability (ACP):

$$\begin{aligned}\hat{c}_{ACP} &= \arg\max_{c_i \in G} P_{ACP}(c_i|d_k) \\ &= \arg\max_{c_i \in G} \frac{\sum_{w_j \in V_{d_k}} \#(w_j, d_k)\kappa_{ji}}{\sum_{c_l \in G} \sum_{w_j \in V_{d_k}} \#(w_j, d_k)\kappa_{jl}} \\ &= \arg\max_{c_i \in G} \sum_{w_j \in V_{d_k}} \#(w_j, d_k)\kappa_{ji} \qquad (11)\end{aligned}$$

This method, despite its conceptual simplicity, works well in practice. It could also be easily modified to use different weights for words, such as TF/IDF or relative frequency ratios between geolocated documents and non-geolocated documents, which we intend to try in future work.

4.5 Baselines

There are several natural baselines to use for comparison against the methods described above.

Random Choose $\hat{c}_{rand}$ randomly from a uniform distribution over the entire grid G.

Cell prior maximum Choose the cell with the highest prior probability according to γ: $\hat{c}_{cpm} = \arg\max_{c_i \in G} \gamma_i$.

Most frequent toponym Identify the most frequent toponym in the article and the geotagged Wikipedia articles that match it. Then identify which of those articles has the most incoming links (a measure of its prominence), and then choose $\hat{c}_{mft}$ to be the cell that contains the geotagged location for that article. This is a strong baseline method, but can only be used with Wikipedia.

Note that a toponym matches an article (or equivalently, the article is a candidate for the toponym) either if the toponym is the same as the article's title, or the same as the title after a parenthetical tag or comma-separated higher-level division is removed. For example, the toponym *Tucson* would match articles named *Tucson*, *Tucson (city)* or *Tucson, Arizona*. In this fashion, the set of toponyms, and the list of candidates for each toponym, is generated from the set of all geotagged Wikipedia articles.

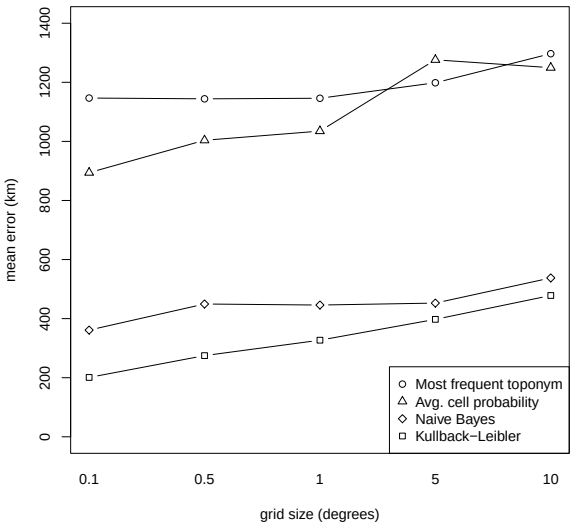


Figure 1: Plot of grid resolution in degrees versus mean error for each method on the Wikipedia dev set.

5 Experiments

The approaches described in the previous section are evaluated on both the geotagged Wikipedia and Twitter datasets. Given a predicted cell $\hat{c}$ for a document, the prediction error is the great-circle distance between the true location and the center of $\hat{c}$, as described in section 3.

Grid resolution and thresholding The major parameter of all our methods is the grid resolution. For both Wikipedia and Twitter, preliminary experiments on the development set were run to plot the prediction error for each method for each level of resolution, and the optimal resolution for each method was chosen for obtaining test results. For the Twitter dataset, an additional parameter is a threshold on the number of feeds each word occurs in: in the preprocessed splits of Eisenstein et al. (2010), all vocabulary items that appear in fewer than 40 feeds are ignored. This thresholding takes away a lot of very useful material; e.g. in the first feed, it removes

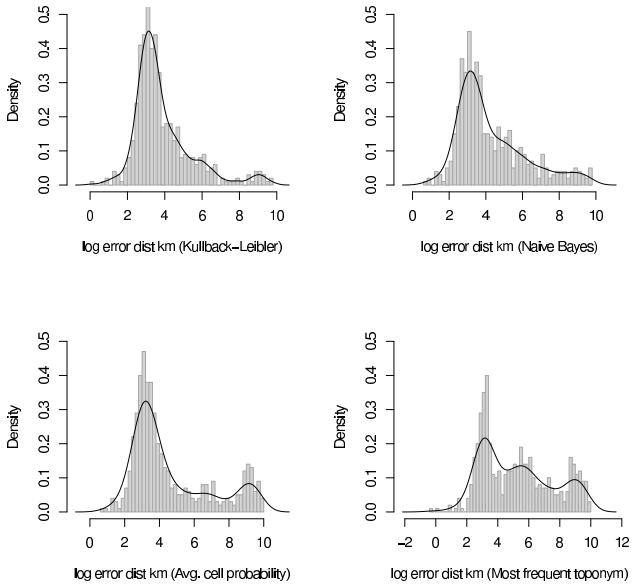


Figure 2: Histograms of distribution of error distances (in km) for grid size 0.5° for each method on the Wikipedia dev set.

both "kirkland" and "redmond" (towns in the Eastside of Lake Washington near Seattle), very useful information for geolocating that user. This suggests that a lower threshold would be better, and this is borne out by our experiments.

Figure 1 graphs the mean error of each method for different resolutions on the Wikipedia dev set, and Figure 2 graphs the distribution of error distances for grid size 0.5° for each method on the Wikipedia dev set. These results indicate that a grid size even smaller than 0.1° might be beneficial. To test this, we ran experiments using a grid size of 0.05° and 0.01° using KL divergence. The mean errors on the dev set increased slightly, from 323 km to 348 and 329 km, respectively, indicating that 0.1° is indeed the minimum.

For the Twitter dataset, we considered both grid size and vocabulary threshold. We recomputed the distributions using several values for both parameters and evaluated on the development set. Table 1 shows mean prediction error using KL divergence, for various combinations of threshold and grid size. Similar tables were constructed for the other strategies. Clearly, the larger grid size of 5° is more optimal than the 0.1° best for Wikipedia. This is unsurprising, given the small size of the corpus. Overall, there is a less clear trend for the other methods

	Grid size (degrees)				
Thr.	0.1	0.5	1	5	10
0	1113.1	996.8	1005.1	969.3	1052.5
2	1018.5	959.5	944.6	911.2	1021.6
3	1027.6	940.8	954.0	913.6	1026.2
5	1011.7	951.0	954.2	**892.0**	1013.0
10	1011.3	968.8	938.5	929.8	1048.0
20	1032.5	987.3	966.0	940.0	1070.1
40	1080.8	1031.5	998.6	981.8	1127.8

Table 1: Mean prediction error (km) on the Twitter dev set for various combinations of vocabulary threshold (in feeds) and grid size, using the KL divergence strategy.

in terms of optimal resolution. Our interpretation of this is that there is greater sparsity for the Twitter dataset, and thus it is more sensitive to arbitrary aspects of how different user feeds are captured in different cells at different granularities.

For the non-baseline strategies, a threshold between about 2 and 5 was best, although no one value in this range was clearly better than another.

Results Based on the optimal resolutions for each method, Table 2 provides the median and mean errors of the methods for both datasets, when run on the test sets. The results clearly show that KL divergence does the best of all the methods considered, with Naive Bayes a close second. Prediction on Wikipedia is very good, with a median value of 11.8 km. Error on Twitter is much higher at 479 km. Nonetheless, this beats Eisenstein et al.'s (2010) median results, though our mean is worse at 967. Using the same threshold of 40 as Eisenstein et al., our results using KL divergence are slightly worse than theirs: median error of 516 km and mean of 986 km.

The difference between Wikipedia and Twitter is unsurprising for several reasons. Wikipedia articles tend to use a lot of toponyms and words that correlate strongly with particular places while many, perhaps most, tweets discuss quotidian details such as what the user ate for lunch. Second, Wikipedia articles are generally longer and thus provide more text to base predictions on. Finally, there are orders of magnitude more training examples for Wikipedia, which allows for greater grid resolution and thus more precise location predictions.

	Wikipedia			Twitter			
Strategy	Degree	Median	Mean	Threshold	Degree	Median	Mean
Kullback-Leibler	0.1	11.8	221	5	5	479	967
Naive Bayes	0.1	15.5	314	5	5	528	989
Avg. cell probability	0.1	24.1	1421	2	10	659	1184
Most frequent toponym	0.5	136	1927	-	-	-	-
Cell prior maximum	5	2333	4309	N/A	0.1	726	1141
Random	0.1	7259	7192	20	0.1	1217	1588
Eisenstein et al.	-	-	-	40	N/A	494	900

Table 2: Prediction error (km) on the Wikipedia and Twitter test sets for each of the strategies using the optimal grid resolution and (for Twitter) the optimal threshold, as determined by performance on the corresponding development sets. Eisenstein et al. (2010) used a fixed Twitter threshold of 40. Threshold makes no difference for cell prior maximum.

Ships One of the most difficult types of Wikipedia pages to disambiguate are those of ships that either are stored or had sunk at a particular location. These articles tend to discuss the exploits of these ships, not their final resting places. Location error on these is usually quite large. However, prediction is quite good for ships that were sunk in particular battles which are described in detail on the page; examples are the USS *Gambier Bay*, USS *Hammann* (DD-412), and the HMS *Majestic* (1895). Another situation that gives good results is when a ship is retired in a location where it is a prominent feature and is thus mentioned in the training set at that location. An example is the USS *Turner Joy*, which is in Bremerton, Washington and figures prominently in the page for Bremerton (which is in the training set).

Another interesting aspect of geolocating ship articles is that ships tend to end up sunk in remote battle locations, such that their article is the only one located in the cell covering the location in the training set. Ship terminology thus dominates such cells, with the effect that our models often (incorrectly) geolocate test articles about other ships to such locations (and often about ships with similar properties). This also leads to generally more accurate geolocation of HMS ships over USS ships; the former seem to have been sunk in more concentrated regions that are themselves less spread out globally.

6 Related work

Lieberman and Lin (2009) also work with geotagged Wikipedia articles, but they do in order so to analyze the likely locations of users who edit such articles. Other researchers have investigated the use of Wikipedia as a source of data for other supervised NLP tasks. Mihalcea and colleagues have investigated the use of Wikipedia in conjunction with word sense disambiguation (Mihalcea, 2007), keyword extraction and linking (Mihalcea and Csomai, 2007) and topic identification (Coursey et al., 2009; Coursey and Mihalcea, 2009). Cucerzan (2007) used Wikipedia to do named entity disambiguation, i.e. identification and coreferencing of named entities by linking them to the Wikipedia article describing the entity.

Some approaches to document geolocation rely largely or entirely on non-textual metadata, which is often unavailable for many corpora of interest, Nonetheless, our methods could be combined with such methods when such metadata is available. For example, given that both Wikipedia and Twitter have a linked structure between documents, it would be possible to use the link-based method given in Backstrom et al. (2010) for predicting the location of Facebook users based on their friends' locations. It is possible that combining their approach with our text-based approach would provide improvements for Facebook, Twitter and Wikipedia datasets. For example, their method performs poorly for users with few geolocated friends, but results improved by combining link-based predictions with IP address predictions. The text written users' updates could be an additional aid for locating such users.

7 Conclusion

We have shown that automatic identification of the location of a document based only on its text can be performed with high accuracy using simple supervised methods and a discrete grid representation of the earth's surface. All of our methods are simple to implement, and both training and testing can be easily parallelized. Our most effective geolocation strategy finds the grid cell whose word distribution has the smallest KL divergence from that of the test document, and easily beats several effective baselines. We predict the location of Wikipedia pages to a median error of 11.8 km and mean error of 221 km. For Twitter, we obtain a median error of 479 km and mean error of 967 km. Using naive Bayes and a simple averaging of word-level cell distributions also both worked well; however, KL was more effective, we believe, because it weights the words in the document most heavily, and thus puts less importance on the less specific word distributions of each cell.

Though we only use text, link-based predictions using the follower graph, as Backstrom et al. (2010) do for Facebook, could improve results on the Twitter task considered here. It could also help with Wikipedia, especially for buildings: for example, the page for Independence Hall in Philadelphia links to geotagged "friend" pages for Philadelphia, the Liberty Bell, and many other nearby locations and buildings. However, we note that we are still primarily interested in geolocation with only text because there are a great many situations in which such linked structure is unavailable. This is especially true for historical corpora like those made available by the Perseus project.[9]

The task of identifying a single location for an entire document provides a convenient way of evaluating approaches for connecting texts with locations, but it is not fully coherent in the context of documents that cover multiple locations. Nonetheless, both the average cell probability and naive Bayes models output a distribution over all cells, which could be used to assign multiple locations. Furthermore, these cell distributions could additionally be used to define a document level prior for resolution of individual toponyms.

[9] www.perseus.tufts.edu/

Though we treated the grid resolution as a parameter, the grids themselves form a hierarchy of cells containing finer-grained cells. Given this, there are a number of obvious ways to combine predictions from different resolutions. For example, given a cell of the finest grain, the average cell probability and naive Bayes models could successively back off to the values produced by their coarser-grained containing cells, and KL divergence could be summed from finest-to-coarsest grain. Another strategy for making models less sensitive to grid resolution is to smooth the per-cell word distributions over neighboring cells; this strategy improved results on Flickr photo geolocation for Serdyukov et al. (2009).

An additional area to explore is to remove the bag-of-words assumption and take into account the ordering between words. This should have a number of obvious benefits, among which are sensitivity to multi-word toponyms such as *New York*, collocations such as *London, Ontario* or *London in Ontario*, and highly indicative terms such as *egg cream* that are made up of generic constituents.

Acknowledgments

This research was supported by a grant from the Morris Memorial Trust Fund of the New York Community Trust and from the Longhorn Innovation Fund for Technology. This paper benefited from reviewer comments and from discussion in the Natural Language Learning reading group at UT Austin, with particular thanks to Matt Lease.

References

Geoffrey Andogah. 2010. *Geographically Constrained Information Retrieval*. Ph.D. thesis, University of Groningen, Groningen, Netherlands, May.

Lars Backstrom, Eric Sun, and Cameron Marlow. 2010. Find me if you can: improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 61–70, New York, NY, USA. ACM.

Kino Coursey and Rada Mihalcea. 2009. Topic identification using wikipedia graph centrality. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, NAACL '09, pages 117–

120, Morristown, NJ, USA. Association for Computational Linguistics.

Kino Coursey, Rada Mihalcea, and William Moen. 2009. Using encyclopedic knowledge for automatic topic identification. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 210–218, Morristown, NJ, USA. Association for Computational Linguistics.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic, June. Association for Computational Linguistics.

Junyan Ding, Luis Gravano, and Narayanan Shivakumar. 2000. Computing geographical scopes of web resources. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 545–556, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

G. Dutton. 1996. Encoding and handling geospatial data with hierarchical triangular meshes. In M.J. Kraak and M. Molenaar, editors, *Advances in GIS Research II*, pages 505–518, London. Taylor and Francis.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1277–1287, Cambridge, MA, October. Association for Computational Linguistics.

Qiang Hao, Rui Cai, Changhu Wang, Rong Xiao, Jiang-Ming Yang, Yanwei Pang, and Lei Zhang. 2010. Equip tourists with knowledge mined from travelogues. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 401–410, New York, NY, USA. ACM.

Jochen L. Leidner. 2008. *Toponym Resolution in Text: Annotation, Evaluation and Applications of Spatial Grounding of Place Names*. Dissertation.Com, January.

M. D. Lieberman and J. Lin. 2009. You are where you edit: Locating Wikipedia users through edit histories. In *ICWSM'09: Proceedings of the 3rd International AAAI Conference on Weblogs and Social Media*, pages 106–113, San Jose, CA, May.

Bruno Martins. 2009. *Geographically Aware Web Text Mining*. Ph.D. thesis, University of Lisbon.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 233–242, New York, NY, USA. ACM.

Rada Mihalcea. 2007. Using Wikipedia for Automatic Word Sense Disambiguation. In *North American Chapter of the Association for Computational Linguistics (NAACL 2007)*.

Simon Overell. 2009. *Geographic Information Retrieval: Classification, Disambiguation and Modelling*. Ph.D. thesis, Imperial College London.

Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 275–281, New York, NY, USA. ACM.

Erik Rauch, Michael Bukatin, and Kenneth Baker. 2003. A confidence-based framework for disambiguating geographic terms. In *Proceedings of the HLT-NAACL 2003 workshop on Analysis of geographic references - Volume 1*, HLT-NAACL-GEOREF '03, pages 50–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bjorn Sandvik. 2008. Using KML for thematic mapping. Master's thesis, The University of Edinburgh.

Pavel Serdyukov, Vanessa Murdock, and Roelof van Zwol. 2009. Placing flickr photos on a map. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 484–491, New York, NY, USA. ACM.

David A. Smith and Gregory Crane. 2001. Disambiguating geographic names in a historical digital library. In *Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries*, ECDL '01, pages 127–136, London, UK. Springer-Verlag.

B. E. Teitler, M. D. Lieberman, D. Panozzo, J. Sankaranarayanan, H. Samet, and J. Sperling. 2008. NewsStand: A new view on news. In *GIS'08: Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 144–153, Irvine, CA, November.

Chengxiang Zhai and John Lafferty. 2001. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, pages 403–410, New York, NY, USA. ACM.

Piggyback: Using Search Engines for Robust Cross-Domain Named Entity Recognition

Stefan Rüd
Institute for NLP
University of Stuttgart
Germany

Massimiliano Ciaramita
Google Research
Zürich
Switzerland

Jens Müller and **Hinrich Schütze**
Institute for NLP
University of Stuttgart
Germany

Abstract

We use search engine results to address a particularly difficult cross-domain language processing task, the adaptation of named entity recognition (NER) from news text to web queries. The key novelty of the method is that we submit a token with context to a search engine and use similar contexts in the search results as additional information for correctly classifying the token. We achieve strong gains in NER performance on news, in-domain and out-of-domain, and on web queries.

1 Introduction

As statistical Natural Language Processing (NLP) matures, NLP components are increasingly used in real-world applications. In many cases, this means that some form of cross-domain adaptation is necessary because there are distributional differences between the labeled training set that is available and the real-world data in the application. To address this problem, we propose a new type of features for NLP data, features extracted from *search engine results*. Our motivation is that search engine results can be viewed as a *substitute for the world knowledge* that is required in NLP tasks, but that can only be extracted from a standard training set or precompiled resources to a limited extent. For example, a named entity (NE) recognizer trained on news text may tag the NE *London* in an out-of-domain web query like *London Klondike gold rush* as a location. But if we train the recognizer on features derived from search results for the sentence to be tagged, correct classification as person is possible. This is because the search results for *London Klondike gold rush* contain snippets in which the first name *Jack* precedes *London*; this is a sure indicator of a last name and hence an NE of type person.

We call our approach *piggyback* and search result-derived features *piggyback features* because we piggyback on a search engine like Google for solving a difficult NLP task.

In this paper, we use piggyback features to address a particularly hard cross-domain problem, the application of an NER system trained on news to web queries. This problem is hard for two reasons. First, the most reliable cue for NEs in English, as in many languages, is *capitalization.* But queries are generally lowercase and even if uppercase characters are used, they are not consistent enough to be reliable features. Thus, applying NER systems trained on news to web queries requires a robust cross-domain approach.

News to queries adaptation is also hard because queries provide *limited context* for NEs. In news text, the first mention of a word like *Ford* is often a fully qualified, unambiguous name like *Ford Motor Corporation* or *Gerald Ford*. In a short query like *buy ford* or *ford pardon*, there is much less context than in news. The lack of context and capitalization, and the noisiness of real-world web queries (tokenization irregularities and misspellings) all make NER hard. The low annotator agreement we found for queries (Section 5) also confirms this point.

The correct identification of NEs in web queries can be crucial for providing relevant pages and ads to users. Other domains have characteristics similar to web queries, e.g., automatically transcribed speech, social communities like Twitter, and SMS. Thus, NER for short, noisy text fragments, in the absence of capitalization, is of general importance.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 965–975,
Portland, Oregon, June 19-24, 2011.

NER performance is to a large extent determined by the quality of the feature representation. Lexical, part-of-speech (PoS), shape and gazetteer features are standard. While the impact of different types of features is well understood for standard NER, fundamentally different types of features can be used when leveraging search engine results. Returning to the NE *London* in the query *London Klondike gold rush*, the feature "proportion of search engine results in which a first name precedes the token of interest" is likely to be useful in NER. Since using search engine results for cross-domain robustness is a new approach in NLP, the design of appropriate features is crucial to its success. A significant part of this paper is devoted to feature design and evaluation.

This paper is organized as follows. Section 2 discusses related work. We describe standard NER features in Section 3. One main contribution of this paper is the large array of piggyback features that we propose in Section 4. We describe the data sets we use and our experimental setup in Sections 5–6. The results in Section 7 show that piggyback features significantly increase NER performance. This is the second main contribution of the paper. We discuss challenges of using piggyback features – due to the cost of querying search engines – and present our conclusions and future work in Section 8.

2 Related work

Barr et al. (2008) found that capitalization of NEs in web queries is inconsistent and not a reliable cue for NER. Guo et al. (2009) exploit query logs for NER in queries. This is also promising, but the context in search results is richer and potentially more informative than that of other queries in logs.

The insight that search results provide useful additional context for natural language expressions is not new. Perhaps the oldest and best known application is pseudo-relevance feedback which uses words and phrases from search results for query expansion (Rocchio, 1971; Xu and Croft, 1996). Search counts or search results have also been used for sentiment analysis (Turney, 2002), for transliteration (Grefenstette et al., 2004), candidate selection in machine translation (Lapata and Keller, 2005), text similarity measurements (Sahami and Heilman, 2006), incorrect parse tree filtering (Yates et al., 2006), and paraphrase evaluation (Fujita and Sato, 2008). The specific NER application we address is most similar to the work of Farkas et al. (2007), but they mainly used frequency statistics as opposed to what we view as the main strength of search results: the ability to get additional contextually similar uses of the token that is to be classified.

Lawson et al. (2010), Finin et al. (2010), and Yetisgen-Yildiz et al. (2010) investigate how to best use Amazon Mechanical Turk (AMT) for NER. We use AMT as a tool, but it is not our focus.

NLP settings where training and test sets are from different domains have received considerable attention in recent years. These settings are difficult because many machine learning approaches assume that source and target are drawn from the same distribution; this is not the case if they are from different domains. Systems applied out of domain typically incur severe losses in accuracy; e.g., Poibeau and Kosseim (2000) showed that newswire-trained NER systems perform poorly when applied to email data (a drop of F_1 from .9 to .5). Recent work in machine learning has made substantial progress in understanding how cross-domain features can be used in effective ways (Ben-David et al., 2010). The development of such features however is to a large extent an empirical problem. From this perspective, one of the most successful approaches to adaptation for NER is based on generating shared feature representations between source and target domains, via unsupervised methods (Ando, 2004; Turian et al., 2010). Turian et al. (2010) show that adapting from CoNLL to MUC-7 (Chinchor, 1998) data (thus between different newswire sources), the best unsupervised feature (Brown clusters) improves F_1 from .68 to .79. Our approach fits within this line of work in that it empirically investigates features with good cross-domain generalization properties. The main contribution of this paper is the design and evaluation of a novel family of features extracted from the largest and most up-to-date repository of world knowledge, the web.

Another source of world knowledge for NER is Wikipedia: Kazama and Torisawa (2007) show that pseudocategories extracted from Wikipedia help for in-domain NER. Cucerzan (2007) uses Wikipedia and web search frequencies to improve NE disambiguation, including simple web search frequencies

	BASE: lexical and input-text part-of-speech features	
1	WORD(k,i)	binary: $w_k = w^i$
2	POS(k,t)	binary: w_k has part-of-speech t
3	SHAPE(k,i)	binary: w_k has (regular expression) shape regexp_i
4	PREFIX(j)	binary: w_0 has prefix j (analogously for suffixes)
	GAZ: gazetteer features	
5	GAZ-B$_l$(k,i)	binary: w_k is the initial word of a phrase, consisting of l words, whose gaz. category is i
6	GAZ-I$_l$(k,i)	binary: w_k is a non-initial word in a phrase, consisting of l words, whose gaz. category is i
	URL: URL features	
7	URL-SUBPART	N(w_0 is substring of a URL)/N(URL)
8	URL-MI(PER)	$1/N$(URL-parts)$\sum$[[$p \in$ URL-parts]] 3MI$_u(p, \mathrm{PER})$–MI$_u(p, \mathrm{O})$–MI$_u(p, \mathrm{ORG})$–MI$_u(p, \mathrm{LOC})$
	LEX: local lexical features	
9	NEIGHBOR(k)	$1/N$(k-neighbors) $\sum$[[$v \in k$-neighbors]] log[NE-BNC(v,k)/OTHER-BNC(v,k)]
10	LEX-MI(PER,d)	$1/N$(d-words) $\sum$[[$v \in$ d-words]] 3MI$_d(v, \mathrm{PER})$–MI$_d(v, \mathrm{O})$–MI$_d(v, \mathrm{ORG})$–MI$_d(v, \mathrm{LOC})$
	BOW: bag-of-word features	
11	BOW-MI(PER)	$1/N$(bow-words)$\sum$[[$v \in$ bow-words]] 3MI$_b(v, \mathrm{PER})$–MI$_b(v, \mathrm{O})$–MI$_b(v, \mathrm{ORG})$–MI$_b(v, \mathrm{LOC})$
	MISC: shape, search part-of-speech, and title features	
12	UPPERCASE	N(s_0 is uppercase)/N(s_0)
13	ALLCAPS	N(s_0 is all-caps)/N(s_0)
14	SPECIAL	binary: w_0 contains special character
15	SPECIAL-TITLE	N(s_{-1} or s_1 in title contains special character)/(N(s_{-1})+N(s_1))
16	TITLE-WORD	N(s_0 occurs in title)/N(title)
17	NOMINAL-POS	N(s_0 is tagged with nominal PoS)/N(s_0)
18	CONTEXT(k)	N(s_k is typical neighbor at position k of named entity)/N(s_0)
19	PHRASE-HIT(k)	N($w_k = s_k$, i.e., word at position k occurs in snippet)/N(s_0)
20	ACRONYM	N($w_{-1}w_0$ or w_0w_1 or $w_{-1}w_0w_1$ occur as acronym)/N(s_0)
21	EMPTY	binary: search result is empty

Table 1: NER features used in this paper. BASE and GAZ are standard features. URL, LEX, BOW and MISC are piggyback (search engine-based) features. See text for explanation of notation. The definitions of URL-MI, LEX-MI, and BOW-MI for LOC, ORG and O are analogous to those for PER. For better readability, we write $\sum$[[x]] for $\sum_x$.

for compound entities.

3 Standard NER features

As is standard in supervised NER, we train an NE tagger on a dataset where each token is represented as a feature vector. In this and the following section we present the features used in our study divided in groups. We will refer to the *target token* – the token we define the feature vector for – as w_0. Its left neighbor is w_{-1} and its right neighbor w_1. Table 1 provides a summary of all features.

Feature group BASE. The first class of features, BASE, is standard in NER. The binary feature WORD(k,i) (line 1) is 1 iff w^i, the i^{th} word in the dictionary, occurs at position k with respect to w_0. The dictionary consists of all words in the training set. The analogous feature for part of speech, POS(k,t) (line 2), is 1 iff w_k has been tagged with PoS t, as determined by TnT tagger (Brants, 2000). We also encode surface properties of the word with simple regular expressions, e.g., *x-ray* is encoded as *x-x* and *9/11* as *d/dd* (SHAPE, line 3). For these features, $k \in \{-1, 0, 1\}$. Finally, we encode prefixes and suffixes, up to three characters long, for w_0 (line 4).

Feature group GAZ. Gazetteer features (lines 5 & 6) are an efficient and effective way of building world knowledge into an NER model. A gazetteer is simply a list of phrases that belong to a particular semantic category. We use gazetteers from (i) GATE (Cunningham et al., 2002): countries, first/last names, trigger words; (ii) WordNet: the 46 lexicographical labels (food, location, person etc.); and (iii) Fortune 500: company names. The two gazetteer features are the binary features GAZ-B$_l$(k,i) and GAZ-I$_l$(k,i). GAZ-B$_l$ (resp. GAZ-I$_l$) is 1

iff w_k occurs as the first (resp. non-initial or internal) word in a phrase of length l that the gazetteer lists as belonging to category i where $k \in \{-1, 0, 1\}$.

4 Piggyback features

Feature groups URL, LEX, BOW, and MISC are piggyback features. We produce these by segmenting the input text into overlapping trigrams $w_1w_2w_3$, $w_2w_3w_4$, $w_3w_4w_5$ etc. Each trigram $w_{i-1}w_iw_{i+1}$ is submitted as a query to the search engine. For all experiments we used the publicly accessible Google Web Search API.[1] The search engine returns a *search result* for the query consisting of, in most cases, 10 *snippets*,[2] each of which contains 0, 1 or more *hits* of the search term w_i. We then compute features for the vector representation of w_i based on the snippets. We again refer to the target token and its neighbors (i.e., the search string) as $w_{-1}w_0w_1$. w_0 is the token that is to be classified (PER, LOC, ORG, or O) and the previous word and the next word serve as context that the search engine can exploit to provide snippets in which w_0 is used in the same NE category as in the input text. O is the tag of a token that is neither LOC, ORG nor PER.

In the definition of the features, we refer to the word in the snippet that matches w_0 as s_0, where the match is determined based on edit distance. The word immediately to the left (resp. right) of s_0 in a snippet is called s_{-1} (resp. s_1).

For non-binary features, we first calculate real values and then binarize them into 10 quantile bins.

Feature group URL. This group exploits NE information in URLs. The feature URL-SUBPART (line 7) is the fraction of URLs in the search result containing w_0 as a substring. To avoid spurious matches, we set the feature to 0 if $\text{length}(w_0) \leq 2$.

For URL-MI (line 8), each URL in the search result is split on special characters into parts (e.g., domain and subdomains). We refer to the set of all parts in the search result as URL-parts. The value of $\text{MI}_u(p, \text{PER})$ is computed on the search results of the training set as the mutual information (MI) between (i) w_0 being PER and (ii) p occurring as part of a URL in the search result. MI is defined as follows:

$$\text{MI}(p, \text{PER}) = \sum_{i \in \{\bar{p}, p\}} \sum_{j \in \{\bar{\text{PER}}, \text{PER}\}} P(i, j) \log \frac{P(i, j)}{P(i)P(j)}$$

For example, for the URL-part p = "staff" (e.g., in `bigcorp.com/staff.htm`), $P(\text{staff})$ is the proportion of search results that contain a URL with the part "staff", $P(\text{PER})$ is the proportion of search results where the search token w_0 is PER and $P(\text{staff,PER})$ is the proportion of search results where w_0 is PER and one of the URLs returned by the search engine has part "staff".

The value of the feature URL-MI is the average difference between the MI of PER and the other named entities. The feature is calculated in the same way for LOC, ORG, and O.

Our initial experiments that used binary features for URL parts were not successful. We then designed URL-MI to integrate all URL information specific to an NE class into one measurement in a way that gives higher weight to strong features and lower weight to weak features. The inner sum on line 8 is the sum of the three differences $\text{MI}(\text{PER}) - \text{MI}(\text{O})$, $\text{MI}(\text{PER}) - \text{MI}(\text{ORG})$, and $\text{MI}(\text{PER}) - \text{MI}(\text{LOC})$. Each of the three summands indicates the relative advantage a URL part p gives to PER vs O (or ORG and LOC). By averaging over all URL parts, one then obtains an assessment of the overall strength of evidence (in terms of MI) for the NE class in question.

Feature group LEX. These features assess how appropriate the words occurring in w_0's local contexts in the search result are for an NE class.

For NEIGHBOR (line 9), we calculate for each word v in the British National Corpus (BNC) the count NE-BNC(v, k), the number of times it occurs at position k with respect to an NE; and OTHER-BNC(v, k), the number of times it occurs at position k with respect to a non-NE. We instantiate the feature for $k = -1$ (left neighbor) and $k = 1$ (right neighbor). The value of NEIGHBOR(k) is defined as the average log ratio of NE-BNC(v, k) and OTHER-BNC(v, k), averaged over the set k-neighbors, the set of words that occur at position k with respect to s_0 in the search result.

In the experiments reported in this paper, we use a PoS-tagged version of the BNC, a balanced corpus of 100M words of British English, as a model

[1] Now deprecated in favor of the new Custom Search API.

[2] Less than 0.5% of the queries return fewer than 10 snippets.

of word distribution in general contexts and in NE contexts that is not specific to either target or source domain. In the BNC, NEs are tagged with just one PoS-tag, but there is no differentiation into subcategories. Note that the search engine could be used again for this purpose; for practical reasons we preferred a static resource for this first study where many design variants were explored.

The feature LEX-MI interprets words occurring before or after s_0 as indicators of named entitihood. The parameter d indicates the "direction" of the feature: before or after. $\text{MI}_d(v, \text{PER})$ is computed on the search results of the training set as the MI between (i) w_0 being PER and (ii) v occurring close to s_0 in the search result either to the left ($d = -1$) or to the right ($d = 1$) of s_0. Close refers to a window of 2 words. The value of LEX-MI(PER,d) is then the average difference between the MI of PER and the other NEs. The definition for LEX-MI(PER,d) is given on line 10. The feature is calculated in the same way for LOC, ORG, and O.

Feature group BOW. The features LEX-MI consider a small window for cooccurrence information and distinguish left and right context. For BOW features, we use a larger window and ignore direction. Our aim is to build a bag-of-words representation of the contexts of w_0 in the result snippets.

$\text{MI}_b(v, \text{PER})$ is computed on the search results of the training set as the MI between (i) w_0 being PER and (ii) v occurring anywhere in the search result. The value of BOW-MI(PER) is the average difference between the MI of PER and the other NEs (line 11). The average is computed over all words $v \in$ bow-words that occur in a particular search result. The feature is calculated in the same way for LOC, ORG, and O.

Feature group MISC. We collect the remaining piggyback features in the group MISC.

The UPPERCASE and ALLCAPS features (lines 12&13) compute the fraction of occurrences of w_0 in the search result with capitalization of only the first letter and all letters, respectively. We exclude titles: capitalization in titles is not a consistent clue for NE status.

The SPECIAL feature (line 14) returns 1 iff any character of w_0 is a number or a special character.

NEs are often surrounded by special characters in web pages, e.g., *Janis Joplin - Summertime*. The SPECIAL-TITLE feature (line 15) captures this by counting the occurrences of numbers and special characters in s_{-1} and s_1 in titles of the search result.

The TITLE-WORD feature (line 16) computes the fraction of occurrences of w_0 in the titles of the search result.

The NOMINAL-POS feature (line 17) calculates the proportion of nominal PoS tags (NN, NNS, NP, NPS) of s_0 in the search result, as determined by a PoS tagging of the snippets using TreeTagger (Schmid, 1994).

The basic idea behind the CONTEXT(k) feature (line 18) is that the occurrence of words of certain shapes and with certain parts of speech makes it either more or less likely that w_0 is an NE. For $k = -1$ (the word preceding s_0 in the search result), we test for words that are adjectives, indefinites, possessive pronouns or numerals (partly based on tagging, partly based on a manually compiled list of words). For $k = 1$ (the word following s_0), we test for words that contain numbers and special characters. This feature is complementary to the feature group LEX in that it is based on shape and PoS and does not estimate different parameters for each word.

The feature PHRASE-HIT(-1) (line 19) calculates the proportion of occurrences of w_0 in the search result where the left neighbor in the snippet is equal to the word preceding w_0 in the search string, i.e., $k = -1$: $s_{-1} = w_{-1}$. PHRASE-HIT(1) is the equivalent for the right neighbor. This feature helps identify phrases – search strings containing NEs are more likely to occur as a phrase in search results.

The ACRONYM feature (line 20) computes the proportion of the initials of $w_{-1}w_0$ or w_0w_1 or $w_{-1}w_0w_1$ occurring in the search result. For example, the abbreviation *GM* is likely to occur when searching for *general motors dealers*.

The binary feature EMPTY (line 21) returns 1 iff the search result is empty. This feature enables the classifier to distinguish true zero values (e.g., for the feature ALLCAPS) from values that are zero because the search engine found no hits.

5 Experimental data

In our experiments, we train an NER classifier on an in-domain data set and test it on two different out-of-domain data sets. We describe these data sets in

	CoNLL trn	CoNLL tst	IEER	KDD-D	KDD-T
LOC	4.1	4.1	1.9	11.9	10.6
ORG	4.9	3.7	3.2	8.2	8.3
PER	5.4	6.4	3.8	5.3	5.4
O	85.6	85.8	91.1	74.6	75.7

Table 2: Percentages of NEs in CoNLL, IEER, and KDD.

this section and the NER classifier and the details of the training regime in the next section, Section 6.

As training data for all models evaluated we used the CoNLL 2003 English NER dataset, a corpus of approximately 300,000 tokens of Reuters news from 1992 annotated with person, location, organization and miscellaneous NE labels (Sang and Meulder, 2003). As out-of-domain newswire evaluation data[3] we use the development test data from the NIST 1999 IEER named entity corpus, a dataset of 50,000 tokens of New York Times (NYT) and Associated Press Weekly news.[4] This corpus is annotated with person, location, organization, cardinal, duration, measure, and date labels. CoNLL and IEER are professionally edited and, in particular, properly capitalized news corpora. As capitalization is absent from queries we lowercased both CoNLL and IEER. We also reannotated the lowercased datasets with PoS categories using the retrained TnT PoS tagger (Brants, 2000) to avoid using non-plausible PoS information. Notice that this step is necessary as otherwise virtually no NNP/NNPS categories would be predicted on the query data because the lowercase NEs of web queries never occur in properly capitalized news; this causes an NER tagger trained on standard PoS to underpredict NEs (1–3% positive rate).

The 2005 KDD Cup is a query topic categorization task based on 800,000 queries (Li et al., 2005).[5] We use a random subset of 2000 queries as a source of web queries. By means of simple regular expressions we excluded from sampling queries that looked like urls or emails ($\approx$ 15%) as they are easy to identify and do not provide a significant challenge. We also excluded queries shorter than 10 characters (4%) and longer than 50 characters (2%) to provide annotators with enough context, but not an overly complex task. The annotation procedure was carried out using Amazon Mechanical Turk. We instructed workers to follow the CoNLL 2003 NER guidelines (augmented with several examples from queries that we annotated) and identify up to three NEs in a short text and copy and paste them into a box with associated multiple choice menu with the 4 CoNLL NE labels: LOC, MISC, ORG, and PER. Five workers annotated each query. In a first round we produced 1000 queries later used for development. We call this set KDD-D. We then expanded the guidelines with a few uncertain cases. In a second round, we generated another 1000 queries. This set will be referred to as KDD-T. Because annotator agreement is low on a per-token basis ($\kappa = .30$ for KDD-D, $\kappa = .34$ for KDD-T (Cohen, 1960)), we remove queries with less than 50% agreement, averaged over the tokens in the query. After this filtering, KDD-D and KDD-T contain 777 and 819 queries, respectively. Most of the rater disagreement involves the MISC NE class. This is not surprising as MISC is a sort of place-holder category that is difficult to define and identify in queries, especially by untrained AMT workers. We thus replaced MISC with the null label O. With these two changes, κ was .54 on KDD-D and .64 on KDD-T. This is sufficient for repeatable experiments.[6]

Table 2 shows the distribution of NE types in the 5 datasets. IEER has fewer NEs than CoNLL, KDD has more. PER is about as prevalent in KDD as in CoNLL, but LOC and ORG have higher percentages, reflecting the fact that people search frequently for locations and commercial organizations. These differences between source domain (CoNLL) and target domains (IEER, KDD) add to the difficulty of cross-domain generalization in this case.

6 Experimental setup

Recall that the input features for a token w_0 consist of standard NER features (BASE and GAZ) and features derived from the search result we obtain by

[3] A reviewer points out that we use the terms in-domain and out-of-domain somewhat liberally. We simply use "different domain" as a short-hand for "different distribution" without making any claim about the exact nature of the difference.

[4] `nltk.googlecode.com/svn/trunk/nltk_data`

[5] `www.sigkdd.org/kdd2005/kddcup.html`

[6] The two KDD sets, along with additional statistics on annotator agreement requested by a reviewer, are available at `ifnlp.org/~schuetze/piggyback11`.

running a search for $w_{-1}w_0w_1$ (URL, LEX, BOW, and MISC). Since the MISC NE class is not annotated in IEER and has low agreement on KDD in the experimental evaluation we focus on the four-class (PER, LOC, ORG, O) NER problem on all datasets. We use BIO encoding as in the original CoNLL task (Sang and Meulder, 2003).

								ALL	LOC	ORG	PER
CoNLL											
c1	l	BASE	GAZ					88.8*	91.9	77.9	93.0
c2	l		GAZ	URL		BOW	MISC	86.4*	90.7	74.0	90.9
c3	l	BASE		URL		BOW	MISC	92.3*	93.7	84.8	96.0
c4	l	BASE	GAZ			BOW	MISC	91.1*	93.3	82.2	94.9
c5	l	BASE	GAZ	URL			MISC	92.7*	**94.9**	84.5	95.9
c6	l	BASE	GAZ	URL		BOW		92.3*	94.2	84.4	95.8
c7	l	BASE	GAZ	URL		BOW	MISC	**93.0**	**94.9**	**85.1**	96.4
c8	l	BASE	GAZ	URL	LEX	BOW	MISC	92.9	94.7	84.9	**96.5**
c9	c	BASE	GAZ					92.9	95.3	87.7	94.6
IEER											
i1	l	BASE	GAZ					57.9*	71.0	37.7	59.9
i2	l		GAZ	URL	LEX	BOW	MISC	63.8*	76.2	26.0	75.9
i3	l	BASE		URL	LEX	BOW	MISC	64.9*	71.8	38.3	73.8
i4	l	BASE	GAZ		LEX	BOW	MISC	67.3	76.7	**41.2**	74.6
i5	l	BASE	GAZ	URL		BOW	MISC	67.8	76.7	40.4	75.8
i6	l	BASE	GAZ	URL	LEX		MISC	**68.1**	77.2	36.9	77.8
i7	l	BASE	GAZ	URL	LEX	BOW		66.6*	**77.4**	38.3	73.9
i8	l	BASE	GAZ	URL	LEX	BOW	MISC	**68.1**	**77.4**	36.2	**78.0**
i9	c	BASE	GAZ					68.6*	77.3	52.3	73.1
KDD-T											
k1	l	BASE	GAZ					34.6*	48.9	19.2	34.7
k2	l		GAZ	URL	LEX		MISC	40.4*	52.1	15.4	50.4
k3	l	BASE		URL	LEX		MISC	40.9*	50.0	20.1	48.0
k4	l	BASE	GAZ		LEX		MISC	41.6*	55.0	**25.2**	45.2
k5	l	BASE	GAZ	URL			MISC	43.0	**57.0**	15.8	50.9
k6	l	BASE	GAZ	URL	LEX			40.7*	55.5	15.8	42.9
k7	l	BASE	GAZ	URL	LEX		MISC	**43.8**	56.4	17.0	52.0
k8	l	BASE	GAZ	URL	LEX	BOW	MISC	**43.8**	56.5	17.4	**52.3**

Table 3: Evaluation results. l = text lowercased, c = original capitalization preserved. ALL scores significantly different from the best results for the three datasets (lines c7, i8, k7) are marked $*$ (see text).

We use SuperSenseTagger (Ciaramita and Altun, 2006)[7] as our NER tagger. It is a first-order conditional HMM trained with the perceptron algorithm (Collins, 2002), a discriminative model with excellent efficiency-performance trade-off (Sha and Pereira, 2003). The model is regularized by averaging (Freund and Schapire, 1999). For all models we used an appropriate development set for choosing the only hyperparameter, T, the number of training iterations on the source data. T must be tuned separately for each evaluation because different target domains have different overfitting patterns.

We train our NER system on an 80% sample of the CoNLL data. For our *in-domain* evaluation, we tune T on a 10% development sample of the CoNLL data and test on the remaining 10%. For our *out-of-domain* evaluation, we use the IEER and KDD test sets. Here T is tuned on the corresponding development sets. Since we do not train on IEER and KDD, these two data sets do not have training set portions. For each data set, we perform 63 runs, corresponding to the $2^6 - 1 = 63$ different non-empty combinations of the 6 feature groups. We report average F_1, generated by five-trial training and evaluation, with random permutations of the training data. We compute the scores using the original CoNLL phrase-based metric (Sang and Meulder, 2003). As a benchmark we use the baseline model with gazetteer features (BASE and GAZ). The robustness of this simple approach is well documented; e.g., Turian et al. (2010) show that the baseline model (gazetteer features without unsupervised features) produces an F_1 of .778 against .788 of the best unsupervised word representation feature.

7 Results and discussion

Table 3 summarizes the experimental results. In each column, the best numbers within a dataset for the "lowercased" runs are bolded (see below for discussion of the "capitalization" runs on lines c9 and i9). For all experiments, we selected a subset of the combinations of the feature groups. This subset always includes the best results and a number of other combinations where feature groups are added to or removed from the optimal combination.

Results for the CoNLL test set show that the 5 feature groups without LEX achieve optimal performance (line c7). Adding LEX improves performance on PER, but decreases overall performance (line c8). Removing GAZ, URL, BOW and MISC

[7] sourceforge.net/projects/supersensetag

from line c7, causes small comparable decreases in performance (lines c3–c6). These feature groups seem to have about the same importance in this experimental setting, but leaving out BASE decreases F_1 by a larger 6.6% (lines c7 vs c2).

The main result for CoNLL is that using piggyback features (line c7) improves F_1 of a standard NER system that uses only BASE and GAZ (line c1) by 4.2%. Even though the emphasis of this paper is on cross-domain robustness, we can see that our approach also has clear in-domain benefits.

The baseline in line c1 is the "lowercase" baseline as indicated by "l". We also ran a "capitalized" baseline ("c") on text with the original capitalization preserved and PoS-tagged in this unchanged form. Comparing lines c7 and c9, we see that piggyback features are able to recover all the performance that is lost when proper capitalization is unavailable. Lin and Wu (2009) report an F_1 score of 90.90 on the original split of the CoNLL data. Our F_1 scores $> 92\%$ can be explained by a combination of randomly partitioning the data and the fact that the four-class problem is easier than the five-class problem LOC-ORG-PER-MISC-O.

We use the t-test to compute significance on the two sets of five F_1 scores from the two experiments that are being compared (two-tailed, $p < .01$ for $t > 3.36$).[8] CoNLL scores that are significantly different from line c7 are marked with $*$.

For IEER, the system performs best for all six feature groups (line i8). Runs significantly different from i8 are marked $*$. When URL, LEX and BOW are removed from the set, performance does not decrease, or only slightly (lines i4, i5, i6), indicating that these three feature groups are least important. In contrast, there is significant evidence for the importance of BASE, GAZ, and MISC: removing them decreases performance by at least 1% (lines i2, i3, i7). The large increase of ORG F_1 when URL is not used is surprising (41.2% on line i4, best performance). The reason seems to be that URL features (and LEX to a lesser extent) do not generalize for ORG. Locations like *Madrid* in CoNLL are frequently tagged ORG when they refer to sports clubs like *Real Madrid*. This is rare in IEER and KDD.

[8]We make the assumption that the distribution of F_1 scores is approximately normal. See Cohen (1995), Noreen (1989) for a discussion of how this affects the validity of the t-test.

Compared to standard NER (using feature groups BASE and GAZ), our combined feature set achieves a performance that is by more than 10% higher (lines i8 vs i1). This demonstrates that piggyback features have robust cross-domain generalization properties. The comparison of lines i8 and i9 confirms that the features effectively compensate for the lack of capitalization, and perform almost as well as (although statistically worse than) a model trained on capitalized data.

The best run on KDD-D was the run with feature groups BASE, GAZ, URL, LEX and MISC. On line k7, we show results for this run for KDD-T and for runs that differ by one feature group (lines k2–k6, k8).[9] The overall best result (43.8%) is achieved when using all feature groups (line k8). Omitting BOW results in the same score for ALL (line k7). Apparently, the local LEX features already capture most useful cooccurrence information and looking at a wider window (as implemented by BOW) is of limited utility. On lines k2–k6, performance generally decreases on ALL and the three NE classes when dropping one of the five feature groups on line k7. One notable exception is an increase for ORG when feature group URL is dropped (line k4, 25.2%, the best performance for ORG of all runs). This is in line with our discussion of the same effect on IEER.

The key take-away from our results on KDD-T is that piggyback features are again (as for IEER) significantly better than standard feature groups BASE and GAZ. Search engine based adaptation has an advantage of 9.2% compared to standard NER (lines k7 vs k1). An F_1 below 45% may not yet be good enough for practical purposes. But even if additional work is necessary to boost the scores further, our model is an important step in this direction.

The low scores for KDD-T are also partially due to our processing of the AMT data. Our selection procedure is biased towards short entities whereas CoNLL guidelines favor long NEs. We can address this by forcing AMT raters to be more consistent with the CoNLL guidelines in the future.

We summarize the experimental results as follows. Piggyback features consistently improve NER for non-well-edited text when used together with standard NER features. While relative improve-

[9]KDD-D F_1 values were about 1% higher than for KDD-T.

ment due to piggyback features increases as out-of-domain data become more different from the in-domain training set, performance declines in absolute terms from .930 (CoNLL) to .681 (IEER) and .438 (KDD-T).

8 Conclusion

Robust cross-domain generalization is key in many NLP applications. In addition to surface and linguistic differences, differences in world knowledge pose a key challenge, e.g., the fact that *Java* refers to a location in one domain and to coffee in another. We have proposed a new way of addressing this challenge. Because search engines attempt to make optimal use of the context a word occurs in, hits shown to the user usually include other uses of the word in semantically similar snippets. These snippets can be used as a more robust and domain-independent representation of the context of the word/phrase than what is available in the input text.

Our first contribution is that we have shown that this basic idea of using search engines for robust domain-independent feature representations yields solid results for one specific NLP problem, NER. Piggyback features achieved an improvement of F_1 of about 10% compared to a baseline that uses BASE and GAZ features. Even in-domain, we were able to get a smaller, but still noticeable improvement of 4.2% due to piggyback features. These results are also important because there are many application domains with noisy text without reliable capitalization, e.g., automatically transcribed speech, tweets, SMS, social communities and blogs.

Our second contribution is that we address a type of NER that is of particular importance: NER for web queries. The query is the main source of information about the user's information need. Query analysis is important on the web because understanding the query, including the subtask of NER, is key for identifying the most relevant documents and the most relevant ads. NER for domains like Twitter and SMS has properties similar to web queries.

A third contribution of this paper is the release of an annotated dataset for web query NER. We hope that this dataset will foster more research on cross-domain generalization and domain adaptation – in particular for NER – and the difficult problem of web query understanding.

This paper is about cross-domain generalization. However, the general idea of using search to provide rich context information to NLP systems is applicable to a broad array of tasks. One of the main hurdles that NLP faces is that the single context a token occurs in is often not sufficient for reliable decisions, be they about attachment, disambiguation or higher-order semantic interpretation. Search makes dozens of additional relevant contexts available and can thus overcome this bottleneck. In the future, we hope to be able to show that other NLP tasks can also benefit from such an enriched context representation.

Future work. We used a web search engine in the experiments presented in this paper. Latencies when using one of the three main commercial search engines Bing, Google and Yahoo! in our scenario range from 0.2 to 0.5 seconds per token. These execution times are prohibitive for many applications. Search engines also tend to limit the number of queries per user and IP address. To gain widespread acceptance of the piggyback idea of using search results for robust NLP, we therefore must explore alternatives to search engines.

In future work, we plan to develop more efficient methods of using search results for cross-domain generalization to avoid the cost of issuing a large number of queries to search engines. Caching will be of obvious importance in this regard. Another avenue we are pursuing is to build a specialized search system for our application in a way similar to Cafarella and Etzioni (2005). While we need good coverage of a large variety of domains for our approach to work, it is not clear how big the index of the search engine must be for good performance. Conceivably, collections much smaller than those indexed by major search engines (e.g., the Google 1T 5-gram corpus or ClueWeb09) might give rise to features with similar robustness properties. It is important to keep in mind, however, that one of the key factors a search engine allows us to leverage is the notion of relevance which might not be always possible to model as accurately with other data.

Acknowledgments. This research was funded by a Google Research Award. We would like to thank Amir Najmi, John Blitzer, Richárd Farkas, Florian Laws, Slav Petrov and the anonymous reviewers for their comments.

References

Rie Kubota Ando. 2004. Exploiting unannotated corpora for tagging and chunking. In *ACL, Companion Volume*, pages 142–145.

Cory Barr, Rosie Jones, and Moira Regelson. 2008. The linguistic structure of English web-search queries. In *EMNLP*, pages 1021–1030.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine Learning*, 79:151–175.

Thorsten Brants. 2000. TnT – A statistical part-of-speech tagger. In *ANLP*, pages 224–231.

Michael J. Cafarella and Oren Etzioni. 2005. A search engine for natural language applications. In *WWW*, pages 442–452.

Nancy A. Chinchor, editor. 1998. *Proceedings of the Seventh Message Understanding Conference*. NIST.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 594–602.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46.

Paul R. Cohen. 1995. *Empirical methods for artificial intelligence*. MIT Press, Cambridge, MA, USA.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *EMNLP-CoNLL*, pages 708–716.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *ACL*, pages 168–175.

Richárd Farkas, György Szarvas, and Róbert Ormándi. 2007. Improving a state-of-the-art named entity recognition system using the world wide web. In *Industrial Conference on Data Mining*, pages 163–172.

Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating named entities in twitter data with crowdsourcing. In *NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 80–88.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37:277–296.

Atsushi Fujita and Satoshi Sato. 2008. A probabilistic model for measuring grammaticality and similarity of automatically generated paraphrases of predicate phrases. In *COLING*, pages 225–232.

Gregory Grefenstette, Yan Qu, and David A. Evans. 2004. Mining the web to create a language model for mapping between English names and phrases and Japanese. In *Web Intelligence*, pages 110–116.

Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. 2009. Named entity recognition in query. In *SIGIR*, pages 267–274.

Jun'ichi Kazama and Kentaro Torisawa. 2007. Exploiting Wikipedia as external knowledge for named entity recognition. In *EMNLP-CoNLL*, pages 698–707.

Mirella Lapata and Frank Keller. 2005. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*, 2(1):1–31.

Nolan Lawson, Kevin Eustice, Mike Perkowitz, and Meliha Yetisgen-Yildiz. 2010. Annotating large email datasets for named entity recognition with mechanical turk. In *NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 71–79.

Ying Li, Zijian Zheng, and Honghua (Kathy) Dai. 2005. KDD CUP 2005 report: Facing a great challenge. *SIGKDD Explorations Newsletter*, 7:91–99.

Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1030–1038.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses : An Introduction*. Wiley-Interscience.

Thierry Poibeau and Leila Kosseim. 2000. Proper name extraction from non-journalistic texts. In *CLIN*, pages 144–157.

J. J. Rocchio. 1971. Relevance feedback in information retrieval. In Gerard Salton, editor, *The Smart Retrieval System – Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall.

Mehran Sahami and Timothy D. Heilman. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *WWW*, pages 377–386.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL 2003 Shared Task*, pages 142–147.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 134–141.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL*, pages 384–394.

Peter D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *ACL*, pages 417–424.

Jinxi Xu and W. Bruce Croft. 1996. Query expansion using local and global document analysis. In *SIGIR*, pages 4–11.

Alexander Yates, Stefan Schoenmackers, and Oren Etzioni. 2006. Detecting parser errors using web-based semantic filters. In *EMNLP*, pages 27–34.

Meliha Yetisgen-Yildiz, Imre Solti, Fei Xia, and Scott Russell Halgrim. 2010. Preliminary experience with Amazon's Mechanical Turk for annotating medical named entities. In *NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 180–183.

Template-Based Information Extraction without the Templates

Nathanael Chambers and **Dan Jurafsky**
Department of Computer Science
Stanford University
{natec,jurafsky}@stanford.edu

Abstract

Standard algorithms for template-based information extraction (IE) require predefined template schemas, and often labeled data, to learn to extract their slot fillers (e.g., an *embassy* is the *Target* of a *Bombing* template). This paper describes an approach to template-based IE that removes this requirement and performs extraction without knowing the template structure in advance. Our algorithm instead learns the template structure automatically from raw text, inducing template schemas as sets of linked events (e.g., bombings include *detonate*, *set off*, and *destroy* events) associated with semantic roles. We also solve the standard IE task, using the induced syntactic patterns to extract role fillers from specific documents. We evaluate on the MUC-4 terrorism dataset and show that we induce template structure very similar to hand-created gold structure, and we extract role fillers with an F1 score of .40, approaching the performance of algorithms that require full knowledge of the templates.

1 Introduction

A *template* defines a specific type of event (e.g., a bombing) with a set of *semantic roles* (or slots) for the typical entities involved in such an event (e.g., perpetrator, target, instrument). In contrast to work in relation discovery that focuses on learning atomic facts (Banko et al., 2007a; Carlson et al., 2010), templates can extract a richer representation of a particular domain. However, unlike relation discovery, most template-based IE approaches assume foreknowledge of the domain's templates. Very little work addresses how to learn the template structure itself. Our goal in this paper is to perform the standard template filling task, but to first automatically induce the templates from an unlabeled corpus.

There are many ways to represent events, ranging from role-based representations such as frames (Baker et al., 1998) to sequential events in scripts (Schank and Abelson, 1977) and narrative schemas (Chambers and Jurafsky, 2009; Kasch and Oates, 2010). Our approach learns narrative-like knowledge in the form of IE templates; we learn sets of related events and semantic roles, as shown in this sample output from our system:

Bombing Template

{detonate, blow up, plant, explode, defuse, destroy}

Perpetrator:	Person who detonates, plants, blows up
Instrument:	Object that is planted, detonated, defused
Target:	Object that is destroyed, is blown up

A semantic role, such as *target*, is a cluster of syntactic functions of the template's event words (e.g., the objects of *detonate* and *explode*). Our goal is to characterize a domain by learning this template structure completely automatically. We learn templates by first clustering event words based on their proximity in a training corpus. We then use a novel approach to role induction that clusters the syntactic functions of these events based on selectional preferences and coreferring arguments. The induced roles are template-specific (e.g., perpetrator), not universal (e.g., agent or patient) or verb-specific.

After learning a domain's template schemas, we perform the standard IE task of role filling from individual documents, for example:

Perpetrator:	guerrillas
Instrument:	dynamite
Target:	embassy

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 976–986,
Portland, Oregon, June 19-24, 2011.

This extraction stage identifies entities using the learned syntactic functions of our roles. We evaluate on the MUC-4 terrorism corpus with results approaching those of supervised systems.

The core of this paper focuses on how to characterize a domain-specific corpus by learning rich template structure. We describe how to first expand the small corpus' size, how to cluster its events, and finally how to induce semantic roles. Section 5 then describes the extraction algorithm, followed by evaluations against previous work in section 6 and 7.

2 Previous Work

Many template extraction algorithms require full knowledge of the templates and labeled corpora, such as in rule-based systems (Chinchor et al., 1993; Rau et al., 1992) and modern supervised classifiers (Freitag, 1998; Chieu et al., 2003; Bunescu and Mooney, 2004; Patwardhan and Riloff, 2009). Classifiers rely on the labeled examples' surrounding context for features such as nearby tokens, document position, syntax, named entities, semantic classes, and discourse relations (Maslennikov and Chua, 2007). Ji and Grishman (2008) also supplemented labeled with unlabeled data.

Weakly supervised approaches remove some of the need for fully labeled data. Most still require the templates and their slots. One common approach is to begin with unlabeled, but clustered event-specific documents, and extract common word patterns as extractors (Riloff and Schmelzenbach, 1998; Sudo et al., 2003; Riloff et al., 2005; Patwardhan and Riloff, 2007). Filatova et al. (2006) integrate named entities into pattern learning (*PERSON won*) to approximate unknown semantic roles. Bootstrapping with seed examples of known slot fillers has been shown to be effective (Surdeanu et al., 2006; Yangarber et al., 2000). In contrast, this paper removes these data assumptions, learning instead from a corpus of unknown events and unclustered documents, without seed examples.

Shinyama and Sekine (2006) describe an approach to *template learning* without labeled data. They present *unrestricted relation discovery* as a means of discovering relations in unlabeled documents, and extract their fillers. Central to the algorithm is collecting multiple documents describing the same exact event (e.g. Hurricane Ivan), and observing repeated word patterns across documents connecting the same proper nouns. Learned patterns represent binary relations, and they show how to construct tables of extracted entities for these relations. Our approach draws on this idea of using unlabeled documents to discover relations in text, and of defining semantic roles by sets of entities. However, the limitations to their approach are that (1) redundant documents about specific events are required, (2) relations are binary, and (3) only slots with named entities are learned. We will extend their work by showing how to learn without these assumptions, obviating the need for redundant documents, and learning templates with any type and any number of slots.

Large-scale learning of scripts and narrative schemas also captures template-like knowledge from unlabeled text (Chambers and Jurafsky, 2008; Kasch and Oates, 2010). Scripts are sets of related event words and semantic roles learned by linking syntactic functions with coreferring arguments. While they learn interesting event structure, the structures are limited to frequent topics in a large corpus. We borrow ideas from this work as well, but our goal is to instead characterize a specific domain with limited data. Further, we are the first to apply this knowledge to the IE task of filling in template mentions in documents.

In summary, our work extends previous work on unsupervised IE in a number of ways. We are the first to learn MUC-4 templates, and we are the first to extract entities without knowing how many templates exist, without examples of slot fillers, and without event-clustered documents.

3 The Domain and its Templates

Our goal is to learn the general event structure of a domain, and then extract the instances of each learned event. In order to measure performance in both tasks (learning structure and extracting instances), we use the terrorism corpus of MUC-4 (Sundheim, 1991) as our target domain. This corpus was chosen because it is annotated with templates that describe all of the entities involved in each event. An example snippet from a *bombing* document is given here:

*The terrorists **used** explosives against the town hall. El Comercio reported that alleged Shining Path members also **attacked** public facilities in huarpacha, Ambo, tomayquichua, and kichki. Municipal official Sergio Horna was seriously **wounded** in an explosion in Ambo.*

The entities from this document fill the following slots in a MUC-4 bombing template.

Perp: Shining Path members **Victim**: Sergio Horna
Target: public facilities **Instrument**: explosives

We focus on these four string-based slots[1] from the MUC-4 corpus, as is standard in this task. The corpus consists of 1300 documents, 733 of which are labeled with at least one template. There are six types of templates, but only four are modestly frequent: *bombing* (208 docs), *kidnap* (83 docs), *attack* (479 docs), and *arson* (40 docs). 567 documents do not have any templates. Our learning algorithm does not know which documents contain (or do not contain) which templates. After learning event words that represent templates, we induce their slots, not knowing a priori how many there are, and then fill them in by extracting entities as in the standard task. In our example above, the three bold verbs (use, attack, wound) indicate the Bombing template, and their syntactic arguments fill its slots.

4 Learning Templates from Raw Text

Our goal is to learn templates that characterize a domain as described in unclustered, unlabeled documents. This presents a two-fold problem to the learner: it does not know how many events exist, and it does not know which documents describe which event (some may describe multiple events). We approach this problem with a three step process: (1) cluster the domain's event patterns to approximate the template topics, (2) build a new corpus *specific to each cluster* by retrieving documents from a larger unrelated corpus, (3) induce each template's slots using its new (larger) corpus of documents.

4.1 Clustering Events to Learn Templates

We cluster *event patterns* to create templates. An *event pattern* is either (1) a verb, (2) a noun in WordNet under the Event synset, or (3) a verb and the head word of its syntactic object. Examples of each include (1) 'explode', (2) 'explosion', and (3) 'explode:bomb'. We also tag the corpus with an NER system and allow patterns to include named entity types, e.g., 'kidnap:PERSON'. These patterns are crucially needed later to learn a template's slots. However, we first need an algorithm to cluster these patterns to learn the domain's core events. We consider two unsupervised algorithms: Latent Dirichlet Allocation (LDA) (Blei et al., 2003), and agglomerative clustering based on word distance.

4.1.1 LDA for Unknown Data

LDA is a probabilistic model that treats documents as mixtures of topics. It learns topics as discrete distributions (multinomials) over the event patterns, and thus meets our needs as it clusters patterns based on co-occurrence in documents. The algorithm requires the number of topics to be known ahead of time, but in practice this number is set relatively high and the resulting topics are still useful. Our best performing LDA model used 200 topics. We had mixed success with LDA though, and ultimately found our next approach performed slightly better on the document classification evaluation.

4.1.2 Clustering on Event Distance

Agglomerative clustering does not require foreknowledge of the templates, but its success relies on how event pattern similarity is determined.

Ideally, we want to learn that *detonate* and *destroy* belong in the same cluster representing a bombing. Vector-based approaches are often adopted to represent words as feature vectors and compute their distance with cosine similarity. Unfortunately, these approaches typically learn clusters of *synonymous* words that can miss detonate and destroy. Our goal is to instead capture world knowledge of co-occuring events. We thus adopt an assumption that *closeness* in the world is reflected by *closeness* in a text's discourse. We hypothesize that two patterns are related if they occur near each other in a document more often than chance.

Let $g(w_i, w_j)$ be the distance between two events (1 if in the same sentence, 2 in neighboring, etc). Let $C_{dist}(w_i, w_j)$ be the distance-weighted frequency of

[1]There are two Perpetrator slots in MUC-4: Organization and Individual. We consider their union as a single slot.

kidnap: kidnap, kidnap:PER, abduct, release, kidnapping, ransom, robbery, registration
bombing: explode, blow up, locate, place:bomb, detonate, damage, explosion, cause, damage, ...
attack: kill, shoot down, down, kill:civilian, kill:PER, kill:soldier, kill:member, killing, shoot:PER, wave, ...
arson: burn, search, burning, clip, collaborate, ...

Figure 1: The 4 clusters mapped to MUC-4 templates.

two events occurring together:

$$C_{dist}(w_i, w_j) = \sum_{d \in D} \sum_{w_i, w_j \in d} 1 - log_4(g(w_i, w_j)) \quad (1)$$

where d is a document in the set of all documents D. The base 4 logarithm discounts neighboring sentences by 0.5 and within the same sentence scores 1. Using this definition of distance, pointwise mutual information measures our similarity of two events:

$$pmi(w_i, w_j) = P_{dist}(w_i, w_j)/(P(w_i)P(w_j)) \quad (2)$$

$$P(w_i) = \frac{C(w_i)}{\sum_j C(w_j)} \quad (3)$$

$$P_{dist}(w_i, w_j) = \frac{C_{dist}(w_i, w_j)}{\sum_k \sum_l C_{dist}(w_k, w_l)} \quad (4)$$

We run agglomerative clustering with pmi over all event patterns. Merging decisions use the average link score between all new links across two clusters. As with all clustering algorithms, a stopping criterion is needed. We continue merging clusters until any single cluster grows beyond m patterns. We briefly inspected the clustering process and chose $m = 40$ to prevent learned scenarios from intuitively growing too large and ambiguous. Post-evaluation analysis shows that this value has wide flexibility. For example, the Kidnap and Arson clusters are unchanged in $30 < m < 80$, and Bombing unchanged in $30 < m < 50$. Figure 1 shows 3 clusters (of 77 learned) that characterize the main template types.

4.2 Information Retrieval for Templates

Learning a domain often suffers from a lack of training data. The previous section clustered events from the MUC-4 corpus, but its 1300 documents do not provide enough examples of verbs and argument counts to further learn the semantic roles in each cluster. Our solution is to assemble a larger *IR-corpus* of documents for each cluster. For example, MUC-4 labels 83 documents with Kidnap, but our learned cluster (*kidnap*, *abduct*, *release*, ...) retrieved 3954 documents from a general corpus.

We use the Associated Press and New York Times sections of the Gigaword Corpus (Graff, 2002) as our general corpus. These sections include approximately 3.5 million news articles spanning 12 years.

Our retrieval algorithm retrieves documents that score highly with a cluster's tokens. The document score is defined by two common metrics: word match, and word coverage. A document's match score is defined as the average number of times the words in cluster c appear in document d:

$$avgm(d, c) = \frac{\sum_{w \in c} \sum_{t \in d} 1\{w = t\}}{|c|} \quad (5)$$

We define *word coverage* as the number of seen cluster words. Coverage penalizes documents that score highly by repeating a single cluster word a lot. We only score a document if its coverage, $cvg(d, c)$, is at least 3 words (or less for tiny clusters):

$$ir(d, c) = \begin{cases} avgm(d, c) & \text{if } cvg(d, c) > min(3, |c|/4) \\ 0 & \text{otherwise} \end{cases}$$

A document d is retrieved for a cluster c if $ir(d, c) > 0.4$. Finally, we emphasize precision by pruning away 50% of a cluster's retrieved documents that are farthest in distance from the mean document of the retrieved set. Distance is the cosine similarity between bag-of-words vector representations. The confidence value of 0.4 was chosen from a manual inspection among a single cluster's retrieved documents. Pruning 50% was arbitrarily chosen to improve precision, and we did not experiment with other quantities. A search for optimum parameter values may lead to better results.

4.3 Inducing Semantic Roles (Slots)

Having successfully clustered event words and retrieved an *IR-corpus* for each cluster, we now address the problem of *inducing semantic roles*. Our learned roles will then extract entities in the next section and we will evaluate their per-role accuracy.

Most work on unsupervised role induction focuses on learning *verb-specific* roles, starting with seed examples (Swier and Stevenson, 2004; He and

Gildea, 2006) and/or knowing the number of roles (Grenager and Manning, 2006; Lang and Lapata, 2010). Our previous work (Chambers and Jurafsky, 2009) learned *situation-specific* roles over narrative schemas, similar to frame roles in FrameNet (Baker et al., 1998). Schemas link the syntactic relations of verbs by clustering them based on observing coreferring arguments in those positions. This paper extends this intuition by introducing a new vector-based approach to coreference similarity.

4.3.1 Syntactic Relations as Roles

We learn the roles of cluster C by clustering the syntactic relations R_C of its words. Consider the following example:

C = {*go off, explode, set off, damage, destroy*}
R_C = {*go_off:s, go_off:p_in, explode:s, set_off:s*}

where *verb:s* is the verb's subject, *:o* the object, and *p_in* a preposition. We ideally want to cluster R_C as:

bomb = {*go_off:s, explode:s, set_off:o, destroy:s*}
suspect = {*set_off:s*}
target = {*go_off:p_in, destroy:o*}

We want to cluster all subjects, objects, and prepositions. Passive voice is normalized to active[2].

We adopt two views of relation similarity: coreferring arguments and selectional preferences. Chambers and Jurafsky (2008) observed that coreferring arguments suggest a semantic relation between two predicates. In the sentence, *he ran and then he fell*, the subjects of run and fall corefer, and so they likely belong to the same scenario-specific semantic role. We applied this idea to a new vector similarity framework. We represent a relation as a vector of all relations with which their arguments coreferred. For instance, arguments of the relation *go_off:s* were seen coreferring with mentions in *plant:o*, *set_off:o* and *injure:s*. We represent *go_off:s* as a vector of these relation counts, calling this its *coref vector representation*.

Selectional preferences (SPs) are also useful in measuring similarity (Erk and Pado, 2008). A relation can be represented as a vector of its observed arguments during training. The SPs for *go_off:s* in our data include {*bomb, device, charge, explosion*}.

We measure similarity using cosine similarity between the vectors in both approaches. However, coreference and SPs measure different types of similarity. Coreference is a looser narrative similarity (bombings cause injuries), while SPs capture synonymy (plant and place have similar arguments). We observed that many narrative relations are not synonymous, and vice versa. We thus take the maximum of either cosine score as our final similarity metric between two relations. We then back off to the average of the two cosine scores if the max is not confident (less than 0.7); the average penalizes the pair. We chose the value of 0.7 from a grid search to optimize extraction results on the training set.

4.3.2 Clustering Syntactic Functions

We use agglomerative clustering with the above pairwise similarity metric. Cluster similarity is the average link score over all new links crossing two clusters. We include the following sparsity penalty $r(c_a, c_b)$ if there are too few links between clusters c_a and c_b.

$$score(c_a, c_b) = \sum_{w_i \in c_a} \sum_{w_j \in c_b} sim(w_i, w_j) * r(c_a, c_b) \quad (6)$$

$$r(c_a, c_b) = \frac{\sum_{w_i \in c_a} \sum_{w_j \in c_b} 1\{sim(w_i, w_j) > 0\}}{\sum_{w_i \in c_a} \sum_{w_j \in c_b} 1} \quad (7)$$

This penalizes clusters from merging when they share only a few high scoring edges. Clustering stops when the merged cluster scores drop below a threshold optimized to extraction performance on the training data.

We also begin with two assumptions about syntactic functions and semantic roles. The first assumes that the subject and object of a verb carry different semantic roles. For instance, the subject of *sell* fills a different role (Seller) than the object (Good). The second assumption is that each semantic role has a high-level entity type. For instance, the subject of *sell* is a Person or Organization, and the object is a Physical Object.

We implement the first assumption as a constraint in the clustering algorithm, preventing two clusters from merging if their union contains the same verb's subject and object.

We implement the second assumption by automatically labeling each syntactic function with a role type based on its observed arguments. The role types are broad general classes: *Person/Org*, *Physical Object*, or *Other*. A syntactic function is labeled as a

[2]We use the Stanford Parser at nlp.stanford.edu/software

Bombing Template (MUC-4)

Perpetrator *Person/Org* who detonates, blows up, plants, hurls, stages, is detained, is suspected, is blamed on, launches

Instrument A *physical object* that is exploded, explodes, is hurled, causes, goes off, is planted, damages, is set off, is defused

Target A *physical object* that is damaged, is destroyed, is exploded at, is damaged, is thrown at, is hit, is struck

Police *Person/Org* who raids, questions, discovers, investigates, defuses, arrests

N/A A *physical object* that is blown up, destroys

Attack/Shooting Template (MUC-4)

Perpetrator *Person/Org* who assassinates, patrols, ambushes, raids, shoots, is linked to

Victim *Person/Org* who is assassinated, is toppled, is gunned down, is executed, is evacuated

Target *Person/Org* who is hit, is struck, is downed, is set fire to, is blown up, surrounded

Instrument A *physical object* that is fired, injures, downs, is set off, is exploded

Kidnap Template (MUC-4)

Perpetrator *Person/Org* who releases, abducts, kidnaps, ambushes, holds, forces, captures, is imprisoned, frees

Target *Person/Org* who is kidnapped, is released, is freed, escapes, disappears, travels, is harmed, is threatened

Police *Person/Org* who rules out, negotiates, condemns, is pressured, finds, arrests, combs

Weapons Smuggling Template (NEW)

Perpetrator *Person/Org* who smuggles, is seized from, is captured, is detained

Police *Person/Org* who raids, seizes, captures, confiscates, detains, investigates

Instrument A *physical object* that is smuggled, is seized, is confiscated, is transported

Election Template (NEW)

Voter *Person/Org* who chooses, is intimidated, favors, is appealed to, turns out

Government *Person/Org* who authorizes, is chosen, blames, authorizes, denies

Candidate *Person/Org* who resigns, unites, advocates, manipulates, pledges, is blamed

Figure 2: Five learned example templates. All knowledge except the template/role names (e.g., 'Victim') is learned.

class if 20% of its arguments appear under the corresponding WordNet synset[3], or if the NER system labels them as such. Once labeled by type, we separately cluster the syntactic functions for each role type. For instance, Person functions are clustered separate from Physical Object functions. Figure 2 shows some of the resulting roles.

Finally, since agglomerative clustering makes hard decisions, related events to a template may have been excluded in the initial event clustering stage. To address this problem, we identify the 200 *nearby events* to each event cluster. These are simply the top scoring event patterns with the cluster's original events. We add their syntactic functions to their best matching roles. This expands the coverage of each learned role. Varying the 200 amount does not lead to wide variation in extraction performance. Once induced, the roles are evaluated by their entity extraction performance in Section 5.

4.4 Template Evaluation

We now compare our learned templates to those hand-created by human annotators for the MUC-4 terrorism corpus. The corpus contains 6 template types, but two of them occur in only 4 and 14 of the 1300 training documents. We thus only evaluate the 4 main templates (*bombing*, *kidnapping*, *attack*, and *arson*). The gold slots are shown in figure 3.

	Bombing	Kidnap	Attack	Arson
Perpetrator	x	x	x	x
Victim	x	x	x	x
Target	x		x	x
Instrument	x		x	

Figure 3: Slots in the hand-crafted MUC-4 templates.

We evaluate the four learned templates that score highest in the document classification evaluation (to be described in section 5.1), aligned with their MUC-4 types. Figure 2 shows three of our four templates, and two brand new ones that our algorithm learned. Of the four templates, we learned 12 of the 13 semantic roles as created for MUC. In addition, we learned a new role not in MUC for bombings, kidnappings, and arson: the *Police* or *Authorities* role. The annotators chose not to include this in their labeling, but this knowledge is clearly relevant when understanding such events, so we consider it correct. There is one additional Bombing and one Arson role that does not align with MUC-4, marked incorrect.

[3]Physical objects are defined as non-person physical objects

We thus report 92% slot recall, and precision as 14 of 16 (88%) learned slots.

We only measure agreement with the MUC template schemas, but our system learns other events as well. We show two such examples in figure 2: the Weapons Smuggling and Election Templates.

5 Information Extraction: Slot Filling

We now present how to apply our learned templates to information extraction. This section will describe how to extract slot fillers using our templates, but without knowing which templates are correct.

We could simply use a standard IE approach, for example, creating seed words for our new learned templates. But instead, we propose a new method that obviates the need for even a limited human labeling of seed sets. We consider each learned semantic role as a potential slot, and we extract slot fillers using the syntactic functions that were previously learned. Thus, the learned syntactic patterns (e.g., the subject of *release*) serve the dual purpose of both inducing the template slots, and extracting appropriate slot fillers from text.

5.1 Document Classification

A document is labeled for a template if two different conditions are met: (1) it contains at least one trigger phrase, and (2) its average per-token conditional probability meets a strict threshold.

Both conditions require a definition of the conditional probability of a template given a token. The conditional is defined as the token's importance relative to its uniqueness across all templates. This is not the usual conditional probability definition as IR-corpora are different sizes.

$$P(t|w) = \frac{P_{IR_t}(w)}{\sum_{s \in T} P_{IR_s}(w)} \quad (8)$$

where $P_{IR_t}(w)$ is the probability of pattern w in the IR-corpus of template t.

$$P_{IR_t}(w) = \frac{C_t(w)}{\sum_v C_t(v)} \quad (9)$$

where $C_t(w)$ is the number of times word w appears in the IR-corpus of template t. A template's trigger words are defined as words satisfying $P(t|w) > 0.2$.

	Kidnap	Bomb	Attack	Arson
Precision	.64	.83	.66	.30
Recall	.54	.63	.35	1.0
F1	**.58**	**.72**	**.46**	**.46**

Figure 4: Document classification results on test.

Trigger phrases are thus template-specific patterns that are highly indicative of that template.

After identifying triggers, we use the above definition to score a document with a template. A document is labeled with a template if it contains at least one trigger, and its average word probability is greater than a parameter optimized on the training set. A document can be (and often is) labeled with multiple templates.

Finally, we label the sentences that contain triggers and use them for extraction in section 5.2.

5.1.1 Experiment: Document Classification

The MUC-4 corpus links templates to documents, allowing us to evaluate our document labels. We treat each link as a gold label (kidnap, bomb, or attack) for that document, and documents can have multiple labels. Our learned clusters naturally do not have MUC labels, so we report results on the four clusters that score highest with each label.

Figure 4 shows the document classification scores. The bombing template performs best with an F1 score of .72. Arson occurs very few times, and Attack is lower because it is essentially an agglomeration of diverse events (discussed later).

5.2 Entity Extraction

Once documents are labeled with templates, we next extract entities into the template slots. Extraction occurs in the trigger sentences from the previous section. The extraction process is two-fold:

1. Extract all NPs that are arguments of patterns in the template's induced roles.

2. Extract NPs whose heads are observed frequently with one of the roles (e.g., 'bomb' is seen with Instrument relations in figure 2).

Take the following MUC-4 sentence as an example:

> *The two bombs were planted with the exclusive purpose of intimidating the owners of...*

The verb *plant* is in our learned bombing cluster, so step (1) will extract its passive subject *bombs* and map it to the correct instrument role (see figure 2). The human target, *owners*, is missed because *intimidate* was not learned. However, if *owner* is in the selectional preferences of the learned 'human target' role, step (2) correctly extracts it into that role.

These are two different, but complementary, views of semantic roles. The first is that a role is defined by the set of syntactic relations that describe it. Thus, we find all role relations and save their arguments (pattern extraction). The second view is that a role is defined by the arguments that fill it. Thus, we extract all arguments that filled a role in training, regardless of their current syntactic environment.

Finally, we filter extractions whose WordNet or named entity label does not match the learned slot's type (e.g., a Location does not match a Person).

6 Standard Evaluation

We trained on the 1300 documents in the MUC-4 corpus and tested on the 200 document TST3 and TST4 test set. We evaluate the four string-based slots: perpetrator, physical target, human target, and instrument. We merge MUC's two perpetrator slots (individuals and orgs) into one gold Perpetrator slot. As in Patwardhan and Riloff (2007; 2009), we ignore missed optional slots in computing recall. We induced clusters in training, performed IR, and induced the slots. We then extracted entities from the test documents as described in section 5.2.

The standard evaluation for this corpus is to report the F1 score for slot type accuracy, ignoring the template type. For instance, a perpetrator of a bombing and a perpetrator of an attack are treated the same. This allows supervised classifiers to train on all perpetrators at once, rather than template-specific learners. Although not ideal for our learning goals, we report it for comparison against previous work.

Several supervised approaches have presented results on MUC-4, but unfortunately we cannot compare against them. Maslennikov and Chua (2006; 2007) evaluated a random subset of test (they report .60 and .63 F1), and Xiao et al. (2004) did not evaluate all slot types (they report .57 F1).

Figure 5 thus shows our results with previous work that is comparable: the fully supervised and weakly supervised approaches of Patwardhan and Riloff (2009; 2007). We give two numbers for our system: mapping one learned template to Attack, and mapping five. Our learned templates for Attack have a different granularity than MUC-4. Rather than one broad Attack type, we learn several: Shooting, Murder, Coup, General Injury, and Pipeline Attack. We see these subtypes as strengths of our algorithm, but it misses the MUC-4 granularity of Attack. We thus show results when we apply the best five learned templates to Attack, rather than just one. The final F1 with these Attack subtypes is .40.

	P	R	F1
Patwardhan & Riloff-09 : Supervised	48	59	53
Patwardhan & Riloff-07 : Weak-Sup	42	48	44
Our Results (1 attack)	48	25	33
Our Results (5 attack)	**44**	**36**	**40**

Figure 5: MUC-4 extraction, ignoring template type.

F1 Score	Kidnap	Bomb	Arson	Attack
Results	.53	.43	.42	.16 / .25

Figure 6: Performance of individual templates. Attack compares our 1 vs 5 best templates.

Our precision is as good as (and our F1 score near) two algorithms that require knowledge of the templates and/or labeled data. Our algorithm instead learned this knowledge without such supervision.

7 Specific Evaluation

In order to more precisely evaluate each learned template, we also evaluated per-template performance. Instead of merging all slots across all template types, we score the slots within each template type. This is a stricter evaluation than Section 6; for example, bombing victims assigned to attacks were previously deemed correct[4].

Figure 6 gives our results. Three of the four templates score at or above .42 F1, showing that our lower score from the previous section is mainly due to the Attack template. Arson also unexpectedly

[4]We do not address the task of template instance identification (e.g., splitting two bombings into separate instances). This requires deeper discourse analysis not addressed by this paper.

	Precision	Recall	F1
Kidnap	.82	.47	.60 (+.07)
Bomb	.60	.36	.45 (+.02)
Arson	1.0	.29	.44 (+.02)
Attack	.36	.09	.15 (0.0)

Figure 7: Performance of each template type, but only evaluated on documents labeled with each type. All others are removed from test. The parentheses indicate F1 gain over evaluating on all test documents (figure 6).

scored well. It only occurs in 40 documents overall, suggesting our algorithm works with little evidence.

Per-template performace is good, and our .40 overall score from the previous section illustrates that we perform quite well in comparison to the .44-.53 range of weakly and fully supervised results.

These evaluations use the standard TST3 and TST4 test sets, including the documents that are not labeled with any templates. 74 of the 200 test documents are unlabeled. In order to determine where the system's false positives originate, we also measure performance only on the 126 test documents that have at least one template. Figure 7 presents the results on this subset. Kidnap improves most significantly in F1 score (7 F1 points absolute), but the others only change slightly. Most of the false positives in the system thus do not originate from the unlabeled documents (the 74 unlabeled), but rather from extracting incorrect entities from correctly identified documents (the 126 labeled).

8 Discussion

Template-based IE systems typically assume knowledge of the domain and its templates. We began by showing that domain knowledge isn't necessarily required; we learned the MUC-4 template structure with surprising accuracy, learning new semantic roles and several new template structures. We are the first to our knowledge to automatically induce MUC-4 templates. It is possible to take these learned slots and use a previous approach to IE (such as seed-based bootstrapping), but we presented an algorithm that instead uses our learned syntactic patterns. We achieved results with comparable precision, and an F1 score of .40 that approaches prior algorithms that rely on hand-crafted knowledge.

The extraction results are encouraging, but the template induction itself is a central contribution of this work. Knowledge induction plays an important role in moving to new domains and assisting users who may not know what a corpus contains. Recent work in Open IE learns atomic relations (Banko et al., 2007b), but little work focuses on structured scenarios. We learned more templates than just the main MUC-4 templates. A user who seeks to know what information is in a body of text would instantly recognize these as key templates, and could then extract the central entities.

We hope to address in the future how the algorithm's unsupervised nature hurts recall. Without labeled or seed examples, it does not learn as many patterns or robust classifiers as supervised approaches. We will investigate new text sources and algorithms to try and capture more knowledge. The final experiment in figure 7 shows that perhaps new work should first focus on pattern learning and entity extraction, rather than document identification.

Finally, while our pipelined approach (template induction with an IR stage followed by entity extraction) has the advantages of flexibility in development and efficiency, it does involve a number of parameters. We believe the IR parameters are quite robust, and did not heavily focus on improving this stage, but the two clustering steps during template induction require parameters to control stopping conditions and word filtering. While all learning algorithms require parameters, we think it is important for future work to focus on removing some of these to help the algorithm be even more robust to new domains and genres.

Acknowledgments

This work was supported by the National Science Foundation IIS-0811974, and this material is also based upon work supported by the Air Force Research Laboratory (AFRL) under prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the Air Force Research Laboratory (AFRL). Thanks to the Stanford NLP Group and reviewers for helpful suggestions.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In Christian Boitet and Pete Whitelock, editors, *ACL-98*, pages 86–90, San Francisco, California. Morgan Kaufmann Publishers.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007a. Learning relations from the web. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007b. Open information extraction from the web. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*.

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*.

Razvan Bunescu and Raymond Mooney. 2004. Collective information extraction with relational markov networks. In *Proceedings of the Association of Computational Linguistics (ACL)*, pages 438–445.

Andrew Carlson, J. Betteridge, R.C. Wang, E.R. Hruschka Jr., and T.M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of the Association of Computational Linguistics (ACL)*, Hawaii, USA.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Association of Computational Linguistics (ACL)*, Columbus, Ohio.

Hai Leong Chieu, Hwee Tou Ng, and Yoong Keok Lee. 2003. Closing the gap: Learning-based information extraction rivaling knowledge-engineering methods. In *Proceedings of the Association of Computational Linguistics (ACL)*.

Nancy Chinchor, David Lewis, and Lynette Hirschman. 1993. Evaluating message understanding systems: an analysis of the third message understanding conference. *Computational Linguistics*, 19:3:409–449.

Katrin Erk and Sebastian Pado. 2008. A structured vector space model for word meaning in context. In *Proceedings of the 2008 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Elena Filatova, Vasileios Hatzivassiloglou, and Kathleen McKeown. 2006. Automatic creation of domain templates. In *Proceedings of the Association of Computational Linguistics (ACL)*.

Dayne Freitag. 1998. Toward general-purpose learning for information extraction. In *Proceedings of the Association of Computational Linguistics (ACL)*, pages 404–408.

David Graff. 2002. English gigaword. *Linguistic Data Consortium*.

Trond Grenager and Christopher D. Manning. 2006. Unsupervised discovery of a statistical verb lexicon. In *Proceedings of the the 2006 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Shan He and Daniel Gildea. 2006. Self-training and co-training for semantic role labeling: Primary report. Technical Report 891, University of Rochester.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through unsupervised cross-document inference. In *Proceedings of the Association of Computational Linguistics (ACL)*.

Niels Kasch and Tim Oates. 2010. Mining script-like structures from the web. In *Proceedings of NAACL HLT*, pages 34–42.

Joel Lang and Mirella Lapata. 2010. Unsupervised induction of semantic roles. In *Proceedings of the North American Association of Computational Linguistics*.

Mstislav Maslennikov and Tat-Seng Chua. 2007. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the Association of Computational Linguistics (ACL)*.

Siddharth Patwardhan and Ellen Riloff. 2007. Effective ie with semantic affinity patterns and relevant regions. In *Proceedings of the 2007 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Lisa Rau, George Krupka, Paul Jacobs, Ira Sider, and Lois Childs. 1992. Ge nltoolset: Muc-4 test results and analysis. In *Proceedings of the Message Understanding Conference (MUC-4)*, pages 94–99.

Ellen Riloff and Mark Schmelzenbach. 1998. An empirical approach to conceptual case frame acquisition. In *Proceedings of the Sixth Workshop on Very Large Corpora*.

Ellen Riloff, Janyce Wiebe, and William Phillips. 2005. Exploiting subjectivity classification to improve information extraction. In *Proceedings of AAAI-05*.

Roger C. Schank and Robert P. Abelson. 1977. *Scripts, plans, goals and understanding*. Lawrence Erlbaum.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive ie using unrestricted relation discovery. In *Proceedings of NAACL*.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic ie pattern acquisition. In *Proceedings of the Association of Computational Linguistics (ACL)*, pages 224–231.

Beth M. Sundheim. 1991. Third message understanding evaluation and conference (muc-3): Phase 1 status report. In *Proceedings of the Message Understanding Conference*.

Mihai Surdeanu, Jordi Turmo, and Alicia Ageno. 2006. A hybrid approach for the acquisition of information extraction patterns. In *Proceedings of the EACL Workshop on Adaptive Text Extraction and Mining*.

Robert S. Swier and Suzanne Stevenson. 2004. Unsupervised semantic role labelling. In *Proceedings of the 2004 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Jing Xiao, Tat-Seng Chua, and Hang Cui. 2004. Cascading use of soft and hard matching pattern rules for weakly supervised information extraction. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Automatic acquisition of domain knowledge for information extraction. In *COLING*, pages 940–946.

Classifying Arguments by Scheme

Vanessa Wei Feng
Department of Computer Science
University of Toronto
Toronto, ON, M5S 3G4, Canada
weifeng@cs.toronto.edu

Graeme Hirst
Department of Computer Science
University of Toronto
Toronto, ON, M5S 3G4, Canada
gh@cs.toronto.edu

Abstract

Argumentation schemes are structures or templates for various kinds of arguments. Given the text of an argument with premises and conclusion identified, we classify it as an instance of one of five common schemes, using features specific to each scheme. We achieve accuracies of 63–91% in one-against-others classification and 80–94% in pairwise classification (baseline = 50% in both cases).

1 Introduction

We investigate a new task in the computational analysis of arguments: the classification of arguments by the *argumentation schemes* that they use. An argumentation scheme, informally, is a framework or structure for a (possibly defeasible) argument; we will give a more-formal definition and examples in Section 3. Our work is motivated by the need to determine the unstated (or implicitly stated) premises that arguments written in natural language normally draw on. Such premises are called *enthymemes*.

For instance, the argument in Example 1 consists of one explicit premise (the first sentence) and a conclusion (the second sentence):

Example 1 ***[Premise:]*** *The survival of the entire world is at stake.*
[Conclusion:] *The treaties and covenants aiming for a world free of nuclear arsenals and other conventional and biological weapons of mass destruction should be adhered to scrupulously by all nations.*

Another premise is left implicit — "*Adhering to those treaties and covenants is a means of realizing survival of the entire world*". This proposition is an enthymeme of this argument.

Our ultimate goal is to reconstruct the enthymemes in an argument, because determining these unstated assumptions is an integral part of understanding, supporting, or attacking an entire argument. Hence reconstructing enthymemes is an important problem in argument understanding. We believe that first identifying the particular argumentation scheme that an argument is using will help to bridge the gap between stated and unstated propositions in the argument, because each argumentation scheme is a relatively fixed "template" for arguing. That is, given an argument, we first classify its argumentation scheme; then we fit the stated propositions into the corresponding template; and from this we infer the enthymemes.

In this paper, we present an argument scheme classification system as a stage following argument detection and proposition classification. First in Section 2 and Section 3, we introduce the background to our work, including related work in this field, the two core concepts of argumentation schemes and scheme-sets, and the Araucaria dataset. In Section 4 and Section 5 we present our classification system, including the overall framework, data preprocessing, feature selection, and the experimental setups. In the remaining section, we present the essential approaches to solve the leftover problems of this paper which we will study in our future work, and discuss the experimental results, and potential directions for future work.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 987–996,
Portland, Oregon, June 19-24, 2011.

2 Related work

Argumentation has not received a great deal of attention in computational linguistics, although it has been a topic of interest for many years. Cohen (1987) presented a computational model of argumentative discourse. Dick (1987; 1991a; 1991b) developed a representation for retrieval of judicial decisions by the structure of their legal argument — a necessity for finding legal precedents independent of their domain. However, at that time no corpus of arguments was available, so Dick's system was purely theoretical. Recently, the Araucaria project at University of Dundee has developed a software tool for manual argument analysis, with a point-and-click interface for users to reconstruct and diagram an argument (Reed and Rowe, 2004; Rowe and Reed, 2008). The project also maintains an online repository, called AraucariaDB, of marked-up naturally occurring arguments collected by annotators worldwide, which can be used as an experimental corpus for automatic argumentation analysis (for details see Section 3.2).

Recent work on argument interpretation includes that of George, Zukerman, and Nieman (2007), who interpret constructed-example arguments (not naturally occurring text) as Bayesian networks. Other contemporary research has looked at the automatic detection of arguments in text and the classification of premises and conclusions. The work closest to ours is perhaps that of Mochales and Moens (2007; 2008; 2009a; 2009b). In their early work, they focused on automatic detection of arguments in legal texts. With each sentence represented as a vector of shallow features, they trained a multinomial naïve Bayes classifier and a maximum entropy model on the Araucaria corpus, and obtained a best average accuracy of 73.75%. In their follow-up work, they trained a support vector machine to further classify each argumentative clause into a premise or a conclusion, with an F_1 measure of 68.12% and 74.07% respectively. In addition, their context-free grammar for argumentation structure parsing obtained around 60% accuracy.

Our work is "downstream" from that of Mochales and Moens. Assuming the eventual success of their, or others', research program on detecting and classifying the components of an argument, we seek to determine how the pieces fit together as an instance of an argumentation scheme.

3 Argumentation schemes, scheme-sets, and annotation

3.1 Definition and examples

Argumentation schemes are structures or templates for forms of arguments. The arguments need not be deductive or inductive; on the contrary, most argumentation schemes are for *presumptive* or *defeasible* arguments (Walton and Reed, 2002). For example, *argument from cause to effect* is a commonly used scheme in everyday arguments. A list of such argumentation schemes is called a *scheme-set*.

It has been shown that argumentation schemes are useful in evaluating common arguments as fallacious or not (van Eemeren and Grootendorst, 1992). In order to judge the weakness of an argument, a set of critical questions are asked according to the particular scheme that the argument is using, and the argument is regarded as valid if it matches all the requirements imposed by the scheme.

Walton's set of 65 argumentation schemes (Walton et al., 2008) is one of the best-developed scheme-sets in argumentation theory. The five schemes defined in Table 1 are the most commonly used ones, and they are the focus of the scheme classification system that we will describe in this paper.

3.2 Araucaria dataset

One of the challenges for automatic argumentation analysis is that suitable annotated corpora are still very rare, in spite of work by many researchers. In the work described here, we use the Araucaria database[1], an online repository of arguments, as our experimental dataset. Araucaria includes approximately 660 manually annotated arguments from various sources, such as newspapers and court cases, and keeps growing. Although Araucaria has several limitations, such as rather small size and low agreement among annotators[2], it is nonetheless one of the best argumentative corpora available to date.

[1] http://araucaria.computing.dundee.ac.uk/doku.php#araucaria_argumentation_corpus

[2] The developers of Araucaria did not report on inter-annotator agreement, probably because some arguments are annotated by only one commentator.

Argument from example *Premise*: In this particular case, the individual *a* has property *F* and also property *G*. *Conclusion:* Therefore, generally, if *x* has property *F*, then it also has property *G*.
Argument from cause to effect *Major premise*: Generally, if *A* occurs, then *B* will (might) occur. *Minor premise*: In this case, *A* occurs (might occur). *Conclusion*: Therefore, in this case, *B* will (might) occur.
Practical reasoning *Major premise*: I have a goal *G*. *Minor premise*: Carrying out action *A* is a means to realize *G*. *Conclusion*: Therefore, I ought (practically speaking) to carry out this action *A*.
Argument from consequences *Premise*: If *A* is (is not) brought about, good (bad) consequences will (will not) plausibly occur. *Conclusion*: Therefore, *A* should (should not) be brought about.
Argument from verbal classification *Individual premise*: *a* has a particular property *F*. *Classification premise*: For all *x*, if *x* has property *F*, then *x* can be classified as having property *G*. *Conclusion*: Therefore, *a* has property *G*.

Table 1: The five most frequent schemes and their definitions in Walton's scheme-set.

Arguments in Araucaria are annotated in a XML-based format called "AML" (Argument Markup Language). A typical argument (see Example 2) consists of several AU nodes. Each AU node is a complete argument unit, composed of a conclusion proposition followed by optional premise proposition(s) in a linked or convergent structure. Each of these propositions can be further defined as a hierarchical collection of smaller AUs. INSCHEME is the particular scheme (e.g., *"Argument from Consequences"*) of which the current proposition is a member; enthymemes that have been made explicit are annotated as "missing = yes".

Example 2 ***Example of argument markup from Araucaria***

```
<TEXT>If we stop the free creation of art, we will stop
 the free viewing of art.</TEXT>
<AU>
  <PROP identifier="C" missing="yes">
    <PROPTEXT offset="-1">
      The prohibition of the free creation of art should
      not be brought about.</PROPTEXT>
    <INSCHEME scheme="Argument from Consequences"
     schid="0" />
  </PROP>
  <LA>
    <AU>
      <PROP identifier="A" missing="no">
        <PROPTEXT offset="0">
          If we stop the free creation of art, we will
          stop the free viewing of art.</PROPTEXT>
        <INSCHEME scheme="Argument from Consequences"
         schid="0" />
      </PROP>
    </AU>
    <AU>
      <PROP identifier="B" missing="yes">
        <PROPTEXT offset="-1">
          The prohibition of free viewing of art is not
          acceptable.</PROPTEXT>
        <INSCHEME scheme="Argument from Consequences"
        schid="0" />
      </PROP>
    </AU>
  </LA>
</AU>
```

There are three scheme-sets used in the annotations in Araucaria: Walton's scheme-set, Katzav and Reed's (2004) scheme-set, and Pollock's (1995) scheme-set. Each of these has a different set of schemes; and most arguments in Araucaria are marked up according to only one of them. Our experimental dataset is composed of only those arguments annotated in accordance with Walton's scheme-set, within which the five schemes shown in Table 1 constitute 61% of the total occurrences.

4 Methods

4.1 Overall framework

As we noted above, our ultimate goal is to reconstruct enthymemes, the unstated premises, in an argument by taking advantage of the stated propositions; and in order to achieve this goal we need to first determine the particular argumentation scheme that the argument is using. This problem is depicted in Figure 1. Our scheme classifier is the dashed round-cornered rectangle portion of this

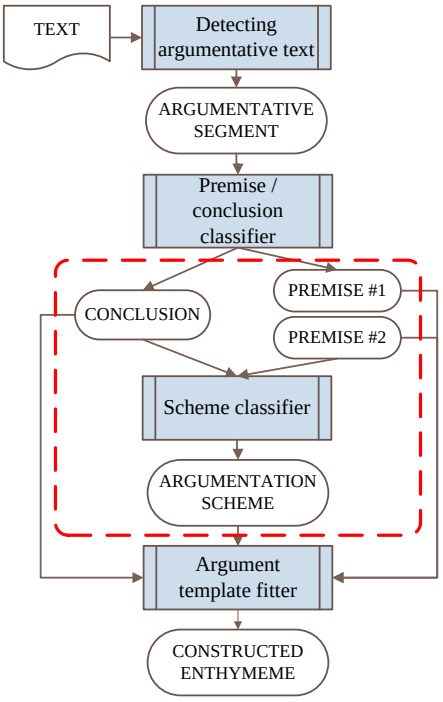


Figure 1: Overall framework of this research.

overall framework: its input is the extracted conclusion and premise(s) determined by an argument detector, followed by a premise / conclusion classifier, given an unknown text as the input to the entire system. And the portion below the dashed round-rectangle represents our long-term goal — to reconstruct the implicit premise(s) in an argument, given its argumentation scheme and its explicit conclusion and premise(s) as input. Since argument detection and classification are not the topic of this paper, we assume here that the input conclusion and premise(s) have already been retrieved, segmented, and classified, as for example by the methods of Mochales and Moens (see Section 2 above). And the scheme template fitter is the topic of our on-going work.

4.2 Data preprocessing

From all arguments in Araucaria, we first extract those annotated in accordance with Walton's scheme-set. Then we break each complex AU node into several simple AUs where no conclusion or premise proposition nodes have embedded AU nodes. From these generated simple arguments, we extract those whose scheme falls into one of the five most frequent schemes as described in Table 1. Furthermore, we remove all enthymemes that have been inserted by the annotator and ignore any argument with a missing conclusion, since the input to our proposed classifier, as depicted in Figure 1, cannot have any access to unstated argumentative propositions.

The resulting preprocessed dataset is composed of 393 arguments, of which 149, 106, 53, 44, and 41 respectively belong to the five schemes in the order shown in Table 1.

4.3 Feature selection

The features used in our work fall into two categories: general features and scheme-specific features.

4.3.1 General features

General features are applicable to arguments belonging to any of the five schemes (shown in Table 2).

For the features **conLoc**, **premLoc**, **gap**, and **lenRat**, we have two versions, differing in terms of their basic measurement unit: *sentence*-based and *token*-based. The final feature, **type**, indicates whether the premises contribute to the conclusion in a linked or convergent order. A *linked argument* (LA) is one that has two or more inter-dependent premise propositions, all of which are necessary to make the conclusion valid, whereas in a *convergent argument* (CA) exactly one premise proposition is sufficient to do so. Since it is observed that there exists a strong correlation between **type** and the particular scheme employed while arguing, we believe **type** can be a good indicator of argumentation scheme. However, although this feature is available to us because it is included in the Araucaria annotations, its value cannot be obtained from raw text as easily as other features mentioned above; but it is possible that we will in the future be able to determine it automatically by taking advantage of some scheme-independent cues such as the discourse relation between the conclusion and the premises.

4.3.2 Scheme-specific features

Scheme-specific features are different for each scheme, since each scheme has its own cue phrases or patterns. The features for each scheme are shown in Table 3 (for complete lists of features see Feng (2010)). In our experiments in Section 5 below, all these features are computed for all arguments; but

conLoc: the location (in token or sentence) of the conclusion in the text.

premLoc: the location (in token or sentence) of the first premise proposition.

conFirst: whether the conclusion appears before the first premise proposition.

gap: the interval (in token or sentence) between the conclusion and the first premise proposition.

lenRat: the ratio of the length (in token or sentence) of the premise(s) to that of the conclusion.

numPrem: the number of explicit premise propositions (PROP nodes) in the argument.

type: type of argumentation structure, i.e., linked or convergent.

Table 2: List of general features.

the features for any particular scheme are used only when it is the subject of a particular task. For example, when we classify *argument from example* in a one-against-others setup, we use the scheme-specific features of that scheme for all arguments; when we classify *argument from example* against *argument from cause to effect*, we use the scheme-specific features of those two schemes.

For the first three schemes (*argument from example*, *argument from cause to effect*, and *practical reasoning*), the scheme-specific features are selected cue phrases or patterns that are believed to be indicative of each scheme. Since these cue phrases and patterns have differing qualities in terms of their precision and recall, we do not treat them all equally. For each cue phrase or pattern, we compute "confidence", the degree of belief that the argument of interest belongs to a particular scheme, using the distribution characteristics of the cue phrase or pattern in the corpus, as described below.

For each argument $\mathcal{A}$, a vector $\mathbf{CV} = \{c_1, c_2, c_3\}$ is added to its feature set, where each c_i indicates the "confidence" of the existence of the specific features associated with each of the first three schemes, $scheme_i$. This is defined in Equation 1:

$$c_i = \frac{1}{N} \sum_{k=1}^{m_i} (P(scheme_i|cp_k) \cdot d_{ik}) \qquad (1)$$

Argument from example
8 keywords and phrases including *for example*, *such as*, *for instance*, etc.; 3 punctuation cues: ":", ";", and "—".

Argument from cause to effect
22 keywords and simple cue phrases including *result*, *related to*, *lead to*, etc.; 10 causal and non-causal relation patterns extracted from WordNet (Girju, 2003).

Practical reasoning
28 keywords and phrases including *want*, *aim*, *objective*, etc.; 4 modal verbs: *should*, *could*, *must*, and *need*; 4 patterns including imperatives and infinitives indicating the goal of the speaker.

Argument from consequences
The counts of positive and negative propositions in the conclusion and premises, calculated from the *General Inquirer*[2].

Argument from verbal classification
The maximal similarity between the *central word* pairs extracted from the conclusion and the premise; the counts of *copula*, *expletive*, and *negative modifier* dependency relations returned by the *Stanford parser*[3] in the conclusion and the premise.

[2] http://www.wjh.harvard.edu/~inquirer/
[3] http://nlp.stanford.edu/software/lex-parser.shtml

Table 3: List of scheme-specific features.

Here m_i is the number of scheme-specific cue phrases designed for $scheme_i$; $P(scheme_i|cp_k)$ is the prior probability that the argument $\mathcal{A}$ actually belongs to $scheme_i$, given that some particular cue phrase cp_k is found in $\mathcal{A}$; d_{ik} is a value indicating whether cp_k is found in $\mathcal{A}$; and the normalization factor N is the number of scheme-specific cue phrase patterns designed for $scheme_i$ with at least one support (at least one of the arguments belonging to $scheme_i$ contains that cue phrase). There are two ways to calculate d_{ik}, *Boolean* and *count*: in *Boolean* mode, d_{ik} is treated as 1 if $\mathcal{A}$ matches cp_k; in *count* mode, d_{ik} equals to the number of times $\mathcal{A}$ matches cp_k; and in both modes, d_{ik} is treated as 0 if cp_k is not found in $\mathcal{A}$.

For *argument from consequences*, since the arguer has an obvious preference for some particular consequence, sentiment orientation can be a good indicator for this scheme, which is quantified by the counts of positive and negative propositions in the conclusion and premise.

For *argument from verbal classification*, there exists a hypernymy-like relation between some pair of propositions (entities, concepts, or actions) located in the conclusion and the premise respectively. The existence of such a relation is quantified by the maximal Jiang-Conrath Similarity (Jiang and Conrath, 1997) between the "central word" pairs extracted from the conclusion and the premise. We parse each sentence of the argument with the Stanford dependency parser, and a word or phrase is considered to be a central word if it is the dependent or governor of several particular dependency relations, which basically represents the attribute or the action of an entity in a sentence, or the entity itself. For example, if a word or phrase is the dependent of the dependency relation *agent*, it is therefore considered as a "central word". In addition, an arguer tends to use several particular syntactic structures (*copula*, *expletive*, and *negative modifier*) when using this scheme, which can be quantified by the counts of those special relations in the conclusion and the premise(s).

5 Experiments

5.1 Training

We experiment with two kinds of classification: *one-against-others* and *pairwise*. We build a pruned C4.5 decision tree (Quinlan, 1993) for each different classification setup, implemented by Weka Toolkit 3.6[5] (Hall et al., 2009).

One-against-others classification A one-against-others classifier is constructed for each of the five most frequent schemes, using the general features and the scheme-specific features for the scheme of interest. For each classifier, there are two possible outcomes: *target_scheme* and other; 50% of the training dataset is arguments associated with *target_scheme*, while the rest is arguments of all the other schemes, which are treated as other. One-against-other classification thus tests the effectiveness of each scheme's specific features.

Pairwise classification A pairwise classifier is constructed for each of the ten possible pairings of the five schemes, using the general features and the scheme-specific features of the two schemes in the pair. For each of the ten classifiers, the training dataset is divided equally into arguments belonging to $scheme_1$ and arguments belonging to $scheme_2$, where $scheme_1$ and $scheme_2$ are two different schemes among the five. Only features associated with $scheme_1$ and $scheme_2$ are used.

5.2 Evaluation

We experiment with different combinations of general features and scheme-specific features (discussed in Section 4.3). To evaluate each experiment, we use the average accuracy over 10 pools of randomly sampled data (each with baseline at 50%[6]) with 10-fold cross-validation.

6 Results

We first present the best average accuracy (BAA) of each classification setup. Then we demonstrate the impact of the feature **type** (convergent or linked argument) on BAAs for different classification setups, since we believe **type** is strongly correlated with the particular argumentation scheme and its value is the only one directly retrieved from the annotations of the training corpus. For more details, see Feng (2010).

6.1 BAAs of each classification setup

target_scheme	*BAA*	d_{ik}	*base*	*type*
example	90.6	count	token	yes
cause	70.4	Boolean / count	token	no
reasoning	90.8	count	sentence	yes
consequences	62.9	–	sentence	yes
classification	63.2	–	token	yes

Table 4: Best average accuracies (BAAs) (%) of one-against-others classification.

[5] http://cs.waikato.ac.nz/ml/weka

[6] We also experiment with using general features only, but the results are consistently below or around the sampling baseline of 50%; therefore, we do not use them as a baseline here.

	example	*cause*	*reasoning*	*consequences*
cause	80.6			
reasoning	93.1	94.2		
consequences	86.9	86.7	97.9	
classification	86.0	85.6	98.3	64.2

Table 5: Best average accuracies (BAAs) (%) of pairwise classification.

Table 4 presents the best average accuracies of one-against-others classification for each of the five schemes. The subsequent three columns list the particular strategies of features incorporation under which those BAAs are achieved (the complete set of possible choices is given in Section 4.3.):

- $\mathbf{d_{ik}}$: *Boolean* or *count* — the strategy of combining scheme-specific cue phrases or patterns using either *Boolean* or *count* for d_{ik}.
- **base**: *sentence* or *token* — the basic unit of applying location- or length-related general features.
- **type**: *yes* or *no* — whether **type** (convergent or linked argument) is incorporated into the feature set.

As Table 4 shows, one-against-others classification achieves high accuracy for *argument from example* and *practical reasoning*: 90.6% and 90.8%. The BAA of *argument from cause to effect* is only just over 70%. However, with the last two schemes (*argument from consequences* and *argument from verbal classification*), accuracy is only in the low 60s; there is little improvement of our system over the majority baseline of 50%. This is probably due at least partly to the fact that these schemes do not have such obvious cue phrases or patterns as the other three schemes which therefore may require more world knowledge encoded, and also because the available training data for each is relatively small (44 and 41 instances, respectively). The BAA for each scheme is achieved with inconsistent choices of base and d_{ik}, but the accuracies that resulted from different choices vary only by very little.

Table 5 shows that our system is able to correctly differentiate between most of the different scheme pairs, with accuracies as high as 98%. It has poor performance (64.0%) only for the pair *argument from consequences* and *argument from verbal classification*; perhaps not coincidentally, these are the two schemes for which performance was poorest in the one-against-others task.

6.2 Impact of type on classification accuracy

As we can see from Table 6, for one-against-others classifications, incorporating **type** into the feature vectors improves classification accuracy in most cases: the only exception is that the best average accuracy of one-against-others classification between *argument from cause to effect* and *others* is obtained without involving **type** into the feature vector — but the difference is negligible, i.e., 0.5 percentage points with respect to the average difference. **Type** also has a relatively small impact on *argument from verbal classification* (2.6 points), compared to its impact on *argument from example* (22.3 points), *practical reasoning* (8.1 points), and *argument from consequences* (7.5 points), in terms of the maximal differences.

Similarly, for pairwise classifications, as shown in Table 7, **type** has significant impact on BAAs, especially on the pairs of *practical reasoning* versus *argument from cause to effect* (17.4 points), *practical reasoning* versus *argument from example* (22.6 points), and *argument from verbal classification* versus *argument from example* (20.2 points), in terms of the maximal differences; but it has a relatively small impact on *argument from consequences* versus *argument from cause to effect* (0.8 point), and *argument from verbal classification* versus *argument from consequences* (1.1 points), in terms of average differences.

7 Future Work

In future work, we will look at automatically classifying **type** (i.e., whether an argument is linked or convergent), as **type** is the only feature directly retrieved from annotations in the training corpus that has a strong impact on improving classification accuracies.

Automatically classifying **type** will not be easy, because sometimes it is subjective to say whether a premise is sufficient by itself to support the conclusion or not, especially when the argument is about

target_scheme	***BAA-t***	***BAA-no t***	***max diff***	***min diff***	***avg diff***
example	90.6	71.6	22.3	10.6	14.7
cause	70.4	70.9	−0.5	−0.6	−0.5
reasoning	90.8	83.2	8.1	7.5	7.7
consequences	62.9	61.9	7.5	−0.6	4.2
classification	63.2	60.7	2.6	0.4	2.0

Table 6: Accuracy (%) with and without **type** in one-against-others classification. *BAA-t* is best average accuracy with **type**, and *BAA-no t* is best average accuracy without **type**. *max diff*, *min diff*, and *avg diff* are maximal, minimal, and average differences between each experimental setup with **type** and without **type** while the remaining conditions are the same.

$scheme_1$	$scheme_2$	***BAA-t***	***BAA-no t***	***max diff***	***min diff***	***avg diff***
cause	example	80.6	69.7	10.9	7.1	8.7
reasoning	example	93.1	73.1	22.8	19.1	20.1
reasoning	cause	94.2	80.5	17.4	8.7	13.9
consequences	example	86.9	76.0	13.8	6.9	10.1
consequences	cause	87.7	86.7	3.8	−1.5	−0.1
consequences	reasoning	97.9	97.9	10.6	0.0	0.8
classification	example	86.0	74.6	20.2	3.7	7.1
classification	cause	85.6	76.8	9.0	3.7	7.1
classification	reasoning	98.3	89.3	8.9	4.2	8.3
classification	consequences	64.0	60.0	6.5	−1.3	1.1

Table 7: Accuracy (%) with and without **type** in pairwise classification. Column headings have the same meanings as in Table 6.

personal opinions or judgments. So for this task, we will initially focus on arguments that are (or at least seem to be) empirical or objective rather than value-based. It will also be non-trivial to determine whether an argument is convergent or linked — whether the premises are independent of one another or not. Cue words and discourse relations between the premises and the conclusion will be one helpful factor; for example, *besides* generally flags an independent premise. And one premise may be regarded as linked to another if either would become an enthymeme if deleted; but determining this in the general case, without circularity, will be difficult.

We will also work on the argument template fitter, which is the final component in our overall framework. The task of the argument template fitter is to map each explicitly stated conclusion and premise into the corresponding position in its scheme template and to extract the information necessary for enthymeme reconstruction. Here we propose a syntax-based approach for this stage, which is similar to tasks in information retrieval. This can be best explained by the argument in Example 1, which uses the particular argumentation scheme *practical reasoning*.

We want to fit the *Premise* and the *Conclusion* of this argument into the *Major premise* and the *Conclusion* slots of the definition of *practical reasoning* (see Table 1), and construct the following conceptual mapping relations:

1. *Survival of the entire world* ⟶ a goal *G*
2. *Adhering to the treaties and covenants aiming for a world free of nuclear arsenals and other conventional and biological weapons of mass destruction* ⟶ action *A*

Thereby we will be able to reconstruct the missing *Minor premise* — the enthymeme in this argument:

> Carrying out *adhering to the treaties and covenants aiming for a world free of nuclear arsenals and other conventional and biological*

weapons of mass destruction is a means of realizing *survival of the entire world*.

8 Conclusion

The argumentation scheme classification system that we have presented in this paper introduces a new task in research on argumentation. To the best of our knowledge, this is the first attempt to classify argumentation schemes.

In our experiments, we have focused on the five most frequently used schemes in Walton's scheme-set, and conducted two kinds of classification: in one-against-others classification, we achieved over 90% best average accuracies for two schemes, with other three schemes in the 60s to 70s; and in pairwise classification, we obtained 80% to 90% best average accuracies for most scheme pairs. The poor performance of our classification system on other experimental setups is partly due to the lack of training examples or to insufficient world knowledge.

Completion of our scheme classification system will be a step towards our ultimate goal of reconstructing the enthymemes in an argument by the procedure depicted in Figure 1. Because of the significance of enthymemes in reasoning and arguing, this is crucial to the goal of understanding arguments. But given the still-premature state of research of argumentation in computational linguistics, there are many practical issues to deal with first, such as the construction of richer training corpora and improvement of the performance of each step in the procedure.

Acknowledgments

This work was financially supported by the Natural Sciences and Engineering Research Council of Canada and by the University of Toronto. We are grateful to Suzanne Stevenson for helpful comments and suggestions.

References

Robin Cohen. 1987. Analyzing the structure of argumentative discourse. *Computational Linguistics*, 13(1–2):11–24.

Judith Dick. 1987. Conceptual retrieval and case law. In *Proceedings, First International Conference on Artificial Intelligence and Law*, pages 106–115, Boston, May.

Judith Dick. 1991a. *A Conceptual, Case-relation Representation of Text for Intelligent Retrieval*. Ph.D. thesis, Faculty of Library and Information Science, University of Toronto, April.

Judith Dick. 1991b. Representation of legal text for conceptual retrieval. In *Proceedings, Third International Conference on Artificial Intelligence and Law*, pages 244–252, Oxford, June.

Vanessa Wei Feng. 2010. Classifying arguments by scheme. Technical report, Department of Computer Science, University of Toronto, November. `http://ftp.cs.toronto.edu/pub/gh/Feng-MSc-2010.pdf`.

Sarah George, Ingrid Zukerman, and Michael Niemann. 2007. Inferences, suppositions and explanatory extensions in argument interpretation. *User Modeling and User-Adapted Interaction*, 17(5):439–474.

Roxana Girju. 2003. Automatic detection of causal relations for question answering. In *Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question Answering*, pages 76–83, Morristown, NJ, USA. Association for Computational Linguistics.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: an update. *SIGKDD Explorations Newsletter*, 11(1):10–18.

Jay J. Jiang and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *International Conference Research on Computational Linguistics (ROCLING X)*, pages 19–33.

Joel Katzav and Chris Reed. 2004. On argumentation schemes and the natural classification of arguments. *Argumentation*, 18(2):239–259.

Raquel Mochales and Marie-Francine Moens. 2008. Study on the structure of argumentation in case law. In *Proceedings of the 2008 Conference on Legal Knowledge and Information Systems*, pages 11–20, Amsterdam, The Netherlands. IOS Press.

Raquel Mochales and Marie-Francine Moens. 2009a. Argumentation mining: the detection, classification and structure of arguments in text. In *ICAIL '09: Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pages 98–107, New York, NY, USA. ACM.

Raquel Mochales and Marie-Francine Moens. 2009b. Automatic argumentation detection and its role in law and the semantic web. In *Proceedings of the 2009 Conference on Law, Ontologies and the Semantic Web*, pages 115–129, Amsterdam, The Netherlands. IOS Press.

Marie-Francine Moens, Erik Boiy, Raquel Mochales Palau, and Chris Reed. 2007. Automatic detection

of arguments in legal texts. In *ICAIL '07: Proceedings of the 11th International Conference on Artificial Intelligence and Law*, pages 225–230, New York, NY, USA. ACM.

John L. Pollock. 1995. *Cognitive Carpentry: A Blueprint for How to Build a Person*. Bradford Books. The MIT Press, May.

J. Ross Quinlan. 1993. C4.5: Programs for machine learning. *Machine Learning*, 16(3):235–240.

Chris Reed and Glenn Rowe. 2004. Araucaria: Software for argument analysis, diagramming and representation. *International Journal of Artificial Intelligence Tools*, 14:961–980.

Glenn Rowe and Chris Reed. 2008. Argument diagramming: The Araucaria project. In *Knowledge Cartography*, pages 163–181. Springer London.

Frans H. van Eemeren and Rob Grootendorst. 1992. *Argumentation, Communication, and Fallacies: A Pragma-Dialectical Perspective*. Routledge.

Douglas Walton and Chris Reed. 2002. Argumentation schemes and defeasible inferences. In *Workshop on Computational Models of Natural Argument, 15th European Conference on Artificial Intelligence*, pages 11–20, Amsterdam, The Netherlands. IOS Press.

Douglas Walton, Chris Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press.

Automatically Evaluating Text Coherence Using Discourse Relations

Ziheng Lin, Hwee Tou Ng and **Min-Yen Kan**
Department of Computer Science
National University of Singapore
13 Computing Drive
Singapore 117417
{linzihen,nght,kanmy}@comp.nus.edu.sg

Abstract

We present a novel model to represent and assess the discourse coherence of text. Our model assumes that coherent text implicitly favors certain types of discourse relation transitions. We implement this model and apply it towards the text ordering ranking task, which aims to discern an original text from a permuted ordering of its sentences. The experimental results demonstrate that our model is able to significantly outperform the state-of-the-art coherence model by Barzilay and Lapata (2005), reducing the error rate of the previous approach by an average of 29% over three data sets against human upper bounds. We further show that our model is synergistic with the previous approach, demonstrating an error reduction of 73% when the features from both models are combined for the task.

1 Introduction

The coherence of a text is usually reflected by its discourse structure and relations. In Rhetorical Structure Theory (RST), Mann and Thompson (1988) observed that certain RST relations tend to favor one of two possible canonical orderings. Some relations (*e.g.*, Concessive and Conditional) favor arranging their satellite span before the nucleus span. In contrast, other relations (*e.g.*, Elaboration and Evidence) usually order their nucleus before the satellite. If a text that uses non-canonical relation orderings is rewritten to use canonical orderings, it often improves text quality and coherence.

This notion of preferential ordering of discourse relations is observed in natural language in general, and generalizes to other discourse frameworks aside from RST. The following example shows a Contrast relation between the two sentences.

(1) [Everyone agrees that most of the nation's old bridges need to be repaired or replaced. $]_{S_1}$ [But there's disagreement over how to do it. $]_{S_2}$

Here the second sentence provides contrasting information to the first. If this order is violated without rewording (*i.e.*, if the two sentences are swapped), it produces an incoherent text (Marcu, 1996).

In addition to the intra-relation ordering, such preferences also extend to inter-relation ordering:

(2) [The Constitution does not expressly give the president such power. $]_{S_1}$ [However, the president does have a duty not to violate the Constitution. $]_{S_2}$ [The question is whether his only means of defense is the veto. $]_{S_3}$

The second sentence above provides a contrast to the previous sentence and an explanation for the next one. This pattern of Contrast-followed-by-Cause is rather common in text (Pitler et al., 2008). Ordering the three sentences differently results in incoherent, cryptic text.

Thus coherent text exhibits measurable preferences for specific intra- and inter-discourse relation ordering. Our key idea is to use the converse of this phenomenon to assess the coherence of a text. In this paper, we detail our model to capture the coherence of a text based on the statistical distribution of the discourse structure and relations. Our method specifically focuses on the discourse relation transitions between adjacent sentences, modeling them in a discourse role matrix.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 997–1006,
Portland, Oregon, June 19-24, 2011.

Our study makes additional contributions. We implement and validate our model on three data sets, which show robust improvements over the current state-of-the-art for coherence assessment. We also provide the first assessment of the upper-bound of human performance on the standard task of distinguishing coherent from incoherent orderings. To the best our knowledge, this is also the first study in which we show output from an automatic discourse parser helps in coherence modeling.

2 Related Work

The study of coherence in discourse has led to many linguistic theories, of which we only discuss algorithms that have been reduced to practice.

Barzilay and Lapata (2005; 2008) proposed an entity-based model to represent and assess *local* textual coherence. The model is motivated by Centering Theory (Grosz et al., 1995), which states that subsequent sentences in a locally coherent text are likely to continue to focus on the same entities as in previous sentences. Barzilay and Lapata operationalized Centering Theory by creating an entity grid model to capture discourse entity transitions at the sentence-to-sentence level, and demonstrated their model's ability to discern coherent texts from incoherent ones. Barzilay and Lee (2004) proposed a domain-dependent HMM model to capture topic shift in a text, where topics are represented by hidden states and sentences are observations. The *global* coherence of a text can then be summarized by the overall probability of topic shift from the first sentence to the last. Following these two directions, Soricut and Marcu (2006) and Elsner et al. (2007) combined the entity-based and HMM-based models and demonstrated that these two models are complementary to each other in coherence assessment.

Our approach differs from these models in that it introduces and operationalizes another indicator of discourse coherence, by modeling a text's discourse relation transitions. Karamanis (2007) has tried to integrate local discourse relations into the Centering-based coherence metrics for the task of information ordering, but was not able to obtain improvement over the baseline method, which is partly due to the much smaller data set and the way the discourse relation information is utilized in heuristic constraints and rules.

To implement our proposal, we need to identify the text's discourse relations. This task, *discourse parsing*, has been a recent focus of study in the natural language processing (NLP) community, largely enabled by the availability of large-scale discourse annotated corpora (Wellner and Pustejovsky, 2007; Elwell and Baldridge, 2008; Lin et al., 2009; Pitler et al., 2009; Pitler and Nenkova, 2009; Lin et al., 2010; Wang et al., 2010). The Penn Discourse Treebank (PDTB) (Prasad et al., 2008) is such a corpus which provides a discourse-level annotation on top of the Penn Treebank, following a predicate-argument approach (Webber, 2004). Crucially, the PDTB provides annotations not only on explicit (*i.e.*, signaled by discourse connectives such as *because*) discourse relations, but also implicit (*i.e.*, inferred by readers) ones.

3 Using Discourse Relations

To utilize discourse relations of a text, we first apply automatic discourse parsing on the input text. While any discourse framework, such as the Rhetorical Structure Theory (RST), could be applied in our work to encode discourse information, we have chosen to work with the Discourse Lexicalized Tree Adjoining Grammar (D-LTAG) by Webber (2004) as embodied in the PDTB, as a PDTB-styled discourse parser[1] developed by Lin et al. (2010) has recently become freely available.

This parser tags each explicit/implicit relation with two levels of relation types. In this work, we utilize the four PDTB Level-1 types: Temporal (Temp), Contingency (Cont), Comparison (Comp), and Expansion (Exp). This parser automatically identifies the discourse relations, labels the argument spans, and classifies the relation types, including identifying common entity and no relation (EntRel and NoRel) as types.

A simple approach to directly model the connections among discourse relations is to use the sequence of discourse relation transitions. Text (2) in Section 1 can be represented by $S_1 \stackrel{Comp}{\longrightarrow} S_2 \stackrel{Cont}{\longrightarrow} S_3$, for instance, when we use Level-1 types. In such a basic approach, we can compile a distribu-

[1] http://wing.comp.nus.edu.sg/~linzihen/parser/

tion of the n-gram discourse relation transition sequences in gold standard coherent text, and a similar one for incoherent text. For example, the above text would generate the transition bigram Comp→Cont. We can build a classifier to distinguish one from the other through learned examples or using a suitable distribution distance measure (*e.g.*, KL Divergence).

In our pilot work where we implemented such a basic model with n-gram features for relation transitions, the performance was very poor. Our analysis revealed a serious shortcoming: as the discourse relation transitions in short texts are few in number, we have very little data to base the coherence judgment on. However, when faced with even short text excerpts, humans can distinguish coherent texts from incoherent ones, as exemplified in our example texts. The basic approach also does not model the intra-relation preference. In Text (1), a Comparison (Comp) relation would be recorded between the two sentences, irregardless of whether S_1 or S_2 comes first. However, it is clear that the ordering of ($S_1 \prec S_2$) is more coherent.

4 A Refined Approach

The central problem with the basic approach is in its sparse modeling of discourse relations. In developing an improved model, we need to better exploit the discourse parser's output to provide more circumstantial evidence to support the system's coherence decision.

In this section, we introduce the concept of a discourse role matrix which aims to capture an expanded set of discourse relation transition patterns. We describe how to represent the coherence of a text with its discourse relations and how to transform such information into a matrix representation. We then illustrate how we use the matrix to formulate a preference ranking problem.

4.1 Discourse Role Matrix

Figure 1 shows a text and its gold standard PDTB discourse relations. When a term appears in a discourse relation, the discourse role of this term is defined as the discourse relation type plus the argument span in which the term is located (*i.e.*, the argument tag). For instance, consider the term "cananea" in the first relation. Since the relation type is a

[Japan normally depends heavily on the Highland Valley and **Cananea** mines as well as the Bougainville mine in Papua New Guinea. $]_{S_1}$ [Recently, Japan has been buying copper elsewhere. $]_{S_2}$ [[But as Highland Valley and **Cananea** begin operating, $]_{C_{3.1}}$ [they are expected to resume their roles as Japan's suppliers. $]_{C_{3.2}}$ $]_{S_3}$ [[According to Fred Demler, metals economist for Drexel Burnham Lambert, New York, $]_{C_{4.1}}$ ["Highland Valley has already started operating $]_{C_{4.2}}$ [and **Cananea** is expected to do so soon." $]_{C_{4.3}}$ $]_{S_4}$

5 discourse relations are present in the above text:

1. Implicit Comparison between S_1 as Arg1, and S_2 as Arg2
2. Explicit Comparison using "but" between S_2 as Arg1, and S_3 as Arg2
3. Explicit Temporal using "as" within S_3 (Clause $C_{3.1}$ as Arg1, and $C_{3.2}$ as Arg2)
4. Implicit Expansion between S_3 as Arg1, and S_4 as Arg2
5. Explicit Expansion using "and" within S_4 (Clause $C_{4.2}$ as Arg1, and $C_{4.3}$ as Arg2)

Figure 1: An excerpt with four contiguous sentences from wsj_0437, showing five gold standard discourse relations. "Cananea" is highlighted for illustration.

S#	Terms				
	copper	cananea	operat	depend	…
S_1	nil	Comp.Arg1	nil	Comp.Arg1	
S_2	Comp.Arg2 Comp.Arg1	nil	nil	nil	
S_3	nil	Comp.Arg2 Temp.Arg1 Exp.Arg1	Comp.Arg2 Temp.Arg1 Exp.Arg1	nil	
S_4	nil	Exp.Arg2	Exp.Arg1 Exp.Arg2	nil	

Table 1: Discourse role matrix fragment for Figure 1. Rows correspond to sentences, columns to stemmed terms, and cells contain extracted discourse roles.

Comparison and "cananea" is found in the Arg1 span, the discourse role of "cananea" is defined as Comp.Arg1. When terms appear in different relations and/or argument spans, they obtain different discourse roles in the text. For instance, "cananea" plays a different discourse role of Temp.Arg1 in the third relation in Figure 1. In the fourth relation, since "cananea" appears in both argument spans, it has two additional discourse roles, Exp.Arg1 and

Exp.Arg2. The discourse role matrix thus represents the different discourse roles of the terms across the continuous text units. We use sentences as the text units, and define terms to be the stemmed forms of the open class words: nouns, verbs, adjectives, and adverbs. We formulate the discourse role matrix such that it encodes the discourse roles of the terms across adjacent sentences.

Table 1 shows a fragment of the matrix representation of the text in Figure 1. Columns correspond to the extracted terms; rows, the contiguous sentences. A cell C_{T_i,S_j} then contains the set of the discourse roles of the term T_i that appears in sentence S_j. For example, the term "cananea" from S_1 takes part in the first relation, so the cell $C_{cananea,S_1}$ contains the role Comp.Arg1. A cell may be empty (*nil*, as in $C_{cananea,S_2}$) or contain multiple discourse roles (as in $C_{cananea,S_3}$, as "cananea" in S_3 participates in the second, third, and fourth relations). Given these discourse relations, building the matrix is straightforward: we note down the relations that a term T_i from a sentence S_j participates in, and record its discourse roles in the respective cell.

We hypothesize that the sequence of discourse role transitions in a coherent text provides clues that distinguish it from an incoherent text. The discourse role matrix thus provides the foundation for computing such role transitions, on a per term basis. In fact, each column of the matrix corresponds to a lexical chain (Morris and Hirst, 1991) for a particular term across the whole text. The key differences from the traditional lexical chains are that our chain nodes' entities are simplified (they share the same stemmed form, instead being connected by WordNet relations), but are further enriched by being typed with discourse relations.

We compile the set of sub-sequences of discourse role transitions for every term in the matrix. These transitions tell us how the discourse role of a term varies through the progression of the text. For instance, "cananea" functions as Comp.Arg1 in S_1 and Comp.Arg2 in S_3, and plays the role of Exp.Arg1 and Exp.Arg2 in S_3 and S_4, respectively. As we have six relation types (Temp(oral), Cont(ingency), Comp(arison), Exp(ansion), EntRel and NoRel) and two argument tags (Arg1 and Arg2) for each type, we have a total of $6 \times 2 = 12$ possible discourse roles, plus a *nil* value. We define a *discourse role transition* as the sub-sequence of discourse roles for a term in multiple consecutive sentences. For example, the discourse role transition of "cananea" from S_1 to S_2 is Comp.Arg1→nil. As a cell may contain multiple discourse roles, a transition may produce multiple sub-sequences. For example, the length 2 sub-sequences for "cananea" from S_3 to S_4, are Comp.Arg2→Exp.Arg2, Temp.Arg1→Exp.Arg2, and Exp.Arg1→Exp.Arg2.

Each sub-sequence has a probability that can be computed from the matrix. To illustrate the calculation, suppose the matrix fragment in Table 1 is the entire discourse role matrix. Then since there are in total 25 length 2 sub-sequences and the sub-sequence Comp.Arg2→Exp.Arg2 has a count of two, its probability is $2/25 = 0.08$. A key property of our approach is that, while discourse transitions are captured locally on a per-term basis, the probabilities of the discourse transitions are aggregated globally, across all terms. We believe that the overall distribution of discourse role transitions for a coherent text is distinguishable from that for an incoherent text. Our model captures the distributional differences of such sub-sequences in coherent and incoherent text in training to determine an unseen text's coherence. To evaluate the coherence of a text, we extract sub-sequences with various lengths from the discourse role matrix as features[2] and compute the sub-sequence probabilities as the feature values.

To further refine the computation of the sub-sequence distribution, we follow (Barzilay and Lapata, 2005) and divide the matrix into a salient matrix and a non-salient matrix. Terms (columns) with a frequency greater than a threshold form the salient matrix, while the rest form the non-salient matrix. The sub-sequence distributions are then calculated separately for these two matrices.

4.2 Preference Ranking

While some texts can be said to be simply coherent or incoherent, often it is a matter of degree. A text can be less coherent when compared to one text, but more coherent when compared to another. As such, since the notion of coherence is relative, we feel that coherence assessment is better represented as

[2]Sub-sequences consisting of only *nil* values are not used as features.

a ranking problem rather than a classification problem. Given a pair of texts, the system ranks them based on how coherent they are. Applications of such a system include differentiating a text from its permutation (*i.e.*, the sentence ordering of the text is shuffled) and identifying a more well-written essay from a pair. Such a system can easily generalize from pairwise ranking into listwise, suitable for the ordinal ranking of a set of texts. Coherence scoring equations can also be deduced (Lapata and Barzilay, 2005) from such a model, yielding coherence scores.

To induce a model for preference ranking, we use the SVM^{light} package[3] by (Joachims, 1999) with the preference ranking configuration for training and testing. All parameters are set to their default values.

5 Experiments

We evaluate our coherence model on the task of *text ordering ranking*, a standard coherence evaluation task used in both (Barzilay and Lapata, 2005) and (Elsner et al., 2007). In this task, the system is asked to decide which of two texts is more coherent. The pair of texts consists of a source text and one of its permutations (*i.e.*, the text's sentence order is randomized). Assuming that the original text is always more discourse-coherent than its permutation, an ideal system will prefer the original to the permuted text. A system's accuracy is thus the number of times the system correctly chooses the original divided by the total number of test pairs.

In order to acquire a large data set for training and testing, we follow the approach in (Barzilay and Lapata, 2005) to create a collection of synthetic data from *Wall Street Journal* (WSJ) articles in the Penn Treebank. All of the WSJ articles are randomly split into a training and a testing set; 40 articles are held out from the training set for development. For each article, its sentences are permuted up to 20 times to create a set of permutations[4]. Each permutation is paired with its source text to form a pair.

We also evaluate on two other data collections (cf. Table 2), provided by (Barzilay and Lapata, 2005), for a direct comparison with their entity-based model. These two data sets consist of Associated Press articles about earthquakes from the North American News Corpus, and narratives from the National Transportation Safety Board. These collections are much smaller than the WSJ data, as each training/testing set contains only up to 100 source articles. Similar to the WSJ data, we construct pairs by permuting each source article up to 20 times.

		WSJ	Earthquakes	Accidents
Train	# Articles	1040	97	100
	# Pairs	19120	1862	1996
	Avg. # Sents	22.0	10.4	11.5
Test	# Articles	1079	99	100
	# Pairs	19896	1956	1986

Table 2: Details of the WSJ, Earthquakes, and Accidents data sets, showing the number of training/testing articles, number of pairs of articles, and average length of an article (in sentences).

Our model has two parameters: (1) the term frequency (TF) that is used as a threshold to identify salient terms, and (2) the lengths of the subsequences that are extracted as features. These parameters are tuned on the development set, and the best ones that produce the optimal accuracy are TF $>= 2$ and lengths of the sub-sequences $<= 3$.

We must also be careful in using the automatic discourse parser. We note that the discourse parser of Lin et al. (2010) comes trained on the PDTB, which provides annotations on top of the whole WSJ data. As we also use the WSJ data for evaluation, we must avoid parsing an article that has already been used in training the parser to prevent training on the test data. We re-train the parser with 24 WSJ sections and use the trained parser to parse the sentences in our WSJ collection from the remaining section. We repeat this re-training/parsing process for all 25 sections. Because the Earthquakes and Accidents data do not overlap with the WSJ training data, we use the parser as distributed to parse these two data sets. Since the discourse parser utilizes paragraph boundaries but a permuted text does not have such boundaries, we ignore paragraph boundaries and treat the source text as if it has only one paragraph. This is to make sure that we do not give the system extra information because of this difference between the source and permuted text.

[3] `http://svmlight.joachims.org/`

[4] Short articles may produce less than 20 permutations.

5.1 Human Evaluation

While the text ordering ranking task has been used in previous studies, two key questions about this task have remained unaddressed in the previous work: (1) to what extent is the assumption that the source text is more coherent than its permutation correct? and (2) how well do humans perform on this task? The answer to the first is needed to validate the correctness of this synthetic task, while the second aims to obtain the upper bound for evaluation. We conduct a human evaluation to answer these questions.

We randomly select 50 source text/permutation pairs from each of the WSJ, Earthquakes, and Accidents training sets. We observe that some of the source texts have formulaic structures in their initial sentences that give away the correct ordering. Sources from the Earthquakes data always begin with a headline sentence and a location-newswire sentence, and many sources from the Accidents data start with two sentences of "This is preliminary ... errors. Any errors ... completed." We remove these sentences from the source and permuted texts, to avoid the subjects judging based on these clues instead of textual coherence. For each set of 50 pairs, we assigned two human subjects (who are not authors of this paper) to perform the ranking. The subjects are told to identify the source text from the pair. When both subjects rank a source text higher than its permutation, we interpret it as the subjects agreeing that the source text is more coherent than the permutation. Table 3 shows the inter-subject agreements.

WSJ	Earthquakes	Accidents	Overall
90.0	90.0	94.0	91.3

Table 3: Inter-subject agreements on the three data sets.

While our study is limited and only indicative, we conclude from these results that the task is tractable. Also, since our subjects' judgments correlate highly with the gold standard, the assumption that the original text is always more coherent than the permuted text is supported. Importantly though, human performance is not perfect, suggesting fair upper bound limits on system performance. We note that the Accidents data set is relatively easier to rank, as it has a higher upper bound than the other two.

5.2 Baseline

Barzilay and Lapata (2005) showed that their entity-based model is able to distinguish a source text from its permutation accurately. Thus, it can serve as a good comparison point for our discourse relation-based model. We compare against their Syntax+Salience setting. Since they did not automatically determine the coreferential information of a permuted text but obtained that from its corresponding source text, we do not perform automatic coreference resolution in our reimplementation of their system. For fair comparison, we follow their experiment settings as closely as possible. We re-use their Earthquakes and Accidents dataset as is, using their exact permutations and pre-processing. For the WSJ data, we need to perform our own pre-processing, thus we employed the Stanford parser[5] to perform sentence segmentation and constituent parsing, followed by entity extraction.

5.3 Results

We perform a series of experiments to answer the following four questions:

1. Does our model outperform the baseline?
2. How do the different features derived from using relation types, argument tags, and salience information affect performance?
3. Can the combination of the baseline and our model outperform the single models?
4. How does system performance of these models compare with human performance on the task?

Baseline results are shown in the first row of Table 4. The results on the Earthquakes and Accidents data are quite similar to those published in (Barzilay and Lapata, 2005) (they reported 83.4% on Earthquakes and 89.7% on Accidents), validating the correctness of our reimplementation of their method.

Row 2 in Table 4 shows the overall performance of the proposed refined model, answering Question 1. The model setting of Type+Arg+Sal means that the model makes use of the discourse roles consisting of 1) relation types and 2) argument tags (*e.g.*,

[5] `http://nlp.stanford.edu/software/lex-parser.shtml`

	WSJ	Earthquakes	Accidents
Baseline	85.71	83.59	89.93
Type+Arg+Sal	**88.06****	**86.50****	**89.38**
Arg+Sal	88.28**	85.89*	87.06
Type+Sal	87.06**	82.98	86.05
Type+Arg	85.98	82.67	87.87
Baseline & Type+Arg+Sal	89.25**	89.72**	91.64**

Table 4: Test set ranking accuracy. The first row shows the baseline performance, the next four show our model with different settings, and the last row is a combined model. Double (**) and single (*) asterisks indicate that the respective model significantly outperforms the baseline at $p < 0.01$ and $p < 0.05$, respectively. We follow Barzilay and Lapata (2008) and use the Fisher Sign test.

the discourse role Comp.Arg2 consists of the type Comp(arison) and the tag Arg2), and 3) two distinct feature sets from salient and non-salient terms. Comparing these accuracies to the baseline, our model significantly outperforms the baseline with $p < 0.01$ in the WSJ and Earthquakes data sets with accuracy increments of 2.35% and 2.91%, respectively. In Accidents, our model's performance is slightly lower than the baseline, but the difference is not statistically significant.

To answer Question 2, we perform feature ablation testing. We eliminate each of the information sources from the full model. In **Row 3**, we first delete relation types from the discourse roles, which causes discourse roles to only contain the argument tags. A discourse role such as Comp.Arg2 becomes Arg2 after deleting the relation type. Comparing Row 3 to Row 2, we see performance reductions on the Earthquakes and Accidents data after eliminating type information. **Row 4** measures the effect of omitting argument tags (Type+Sal). In this setting, the discourse role Comp.Arg2 reduces to Comp. We see a large reduction in performance across all three data sets. This model is also most similar to the basic naïve model in Section 3. These results suggest that the argument tag information plays an important role in our discourse role transition model. **Row 5** omits the salience information (Type+Arg), which also markedly reduces performance. This result supports the use of salience, in line with the conclusion drawn in (Barzilay and Lapata, 2005).

To answer Question 3, we train and test a combined model using features from both the baseline and our model (shown as **Row 6** in Table 4). The entity-based model of Barzilay and Lapata (2005) connects the local entity transition with textual coherence, while our model looks at the patterns of discourse relation transitions. As these two models focus on different aspects of coherence, we expect that they are complementary to each other. The combined model in all three data sets gives the highest performance in comparison to all single models, and it significantly outperforms the baseline model with $p < 0.01$. This confirms that the combined model is linguistically richer than the single models as it integrates different information together, and the entity-based model and our model are synergistic.

To answer Question 4, when compared to the human upper bound (Table 3), the performance gaps for the baseline model are relatively large, while those for our full model are more acceptable in the WSJ and Earthquakes data. For the combined model, the error rates are significantly reduced in all three data sets. The average error rate reductions against 100% are 9.57% for the full model and 26.37% for the combined model. If we compute the average error rate reductions against the human upper bounds (rather than an oracular 100%), the average error rate reduction for the full model is 29% and that for the combined model is 73%. While these are only indicative results, they do highlight the significant gains that our model is making towards reaching human performance levels.

We further note that some of the permuted texts may read as coherently as the original text. This phenomenon has been observed in several natural language synthesis tasks such as generation and summarization, in which a single gold standard is inadequate to fully assess performance. As such, both automated systems and humans may actually perform better than our performance measures indicate. We leave it to future work to measure the impact of this phenomenon.

6 Analysis and Discussion

When we compare the accuracies of the full model in the three data sets (Row 2), the accuracy in the Accidents data is the highest (89.38%), followed by

that in the WSJ (88.06%), with Earthquakes at the lowest (86.50%). To explain the variation, we examine the ratio between the number of the relations in the article and the article length (*i.e.*, number of sentences). This ratio is 1.22 for the Accidents source articles, 1.2 for the WSJ, and 1.08 for Earthquakes. The relation/length ratio gives us an idea of how often a sentence participates in discourse relations. A high ratio means that the article is densely interconnected by discourse relations, and may make distinguishing this article from its permutation easier compared to that for a loosely connected article.

We expect that when a text contains more discourse relation types (*i.e.*, Temporal, Contingency, Comparison, Expansion) and less EntRel and NoRel types, it is easier to compute how coherent this text is. This is because compared to EntRel and NoRel, these four discourse relations can combine to produce meaningful transitions, such as the example Text (2). To examine how this affects performance, we calculate the average ratio between the number of the four discourse relations in the permuted text and the length for the permuted text. The ratio is 0.58 for those that are correctly ranked by our system, and 0.48 for those that are incorrectly ranked, which supports our hypothesis.

We also examined the learning curves for our Type+Arg+Sal model, the baseline model, and the combined model on the data sets, as shown in Figure 2(a)–2(c). In the WSJ data, the accuracies for all three models increase rapidly as more pairs are added to the training set. After 2,000 pairs, the increase slows until 8,000 pairs, after which the curve is nearly flat. From the curves, our model consistently performs better than the baseline with a significant gap, and the combined model also consistently and significantly outperforms the other two. Only about half of the total training data is needed to reach optimal performance for all three models. The learning curves in the Earthquakes data show that the performance for all models is always increasing as more training pairs are utilized. The Type+Arg+Sal and combined models start with lower accuracies than the baseline, but catch up with it at 1,000 and 400 pairs, respectively, and consistently outperform the baseline beyond this point. On the other hand, the learning curves for the Type+Arg+Sal and baseline models in Accidents do not show any one curve consistently better than the other: our model outperforms in the middle segment but underperforms in the first and last segments. The curve for the combined model shows a consistently significant gap between it and the other two curves after the point at 400 pairs.

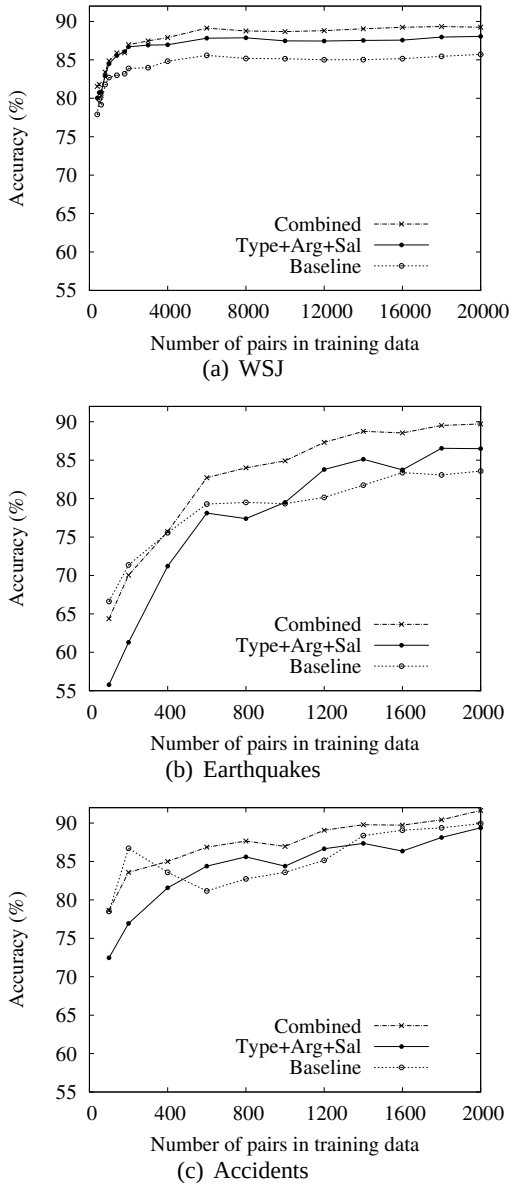


Figure 2: Learning curves for the Type+Arg+Sal, the baseline, and the combined models on the three data sets.

With the performance of the model as it is, how can future work improve upon it? We point out one weakness that we plan to explore. We use the full Type+Arg+Sal model trained on the WSJ training

data to test Text (2) from the introduction. As (2) has 3 sentences, permuting it gives rise to 5 permutations. The model is able to correctly rank four of these 5 pairs. The only permutation it fails on is ($S_3 \prec S_1 \prec S_2$), when the last sentence is moved to the beginning. A very good clue of coherence in Text (2) is the explicit Comp relation between S_1 and S_2. Since this clue is retained in ($S_3 \prec S_1 \prec S_2$), it is difficult for the system to distinguish this ordering from the source. In contrast, as this clue is not present in the other four permutations, it is easier to distinguish them as incoherent. By modeling longer range discourse relation transitions, we may be able to discern these two cases.

While performance on identifying explicit discourse relations in the PDTB is as high as 93% (Pitler et al., 2008), identifying implicit ones has been shown to be a difficult task with accuracy of 40% at Level-2 types (Lin et al., 2009). As the overall performance of the PDTB parser is still less accurate than we hope it to be, we expect that our proposed model will give better performance than it does now, when the current PDTB parser performance is improved.

7 Conclusion

We have proposed a new model for discourse coherence that leverages the observation that coherent texts preferentially follow certain discourse structures. We posit that these structures can be captured in and represented by the patterns of discourse relation transitions. We first demonstrate that simply using the sequence of discourse relation transition leads to sparse features and is insufficient to distinguish coherent from incoherent text. To address this, our method transforms the discourse relation transitions into a discourse role matrix. The matrix schematically represents term occurrences in text units and associates each occurrence with its discourse roles in the text units. In our approach, n-gram sub-sequences of transitions per term in the discourse role matrix then constitute the more fine-grained evidence used in our model to distinguish coherence from incoherence.

When applied to distinguish a source text from a sentence-reordered permutation, our model significantly outperforms the previous state-of-the-art, the entity-based local coherence model. While the entity-based model captures repetitive mentions of entities, our discourse relation-based model gleans its evidence from the argumentative and discourse structure of the text. Our model is complementary to the entity-based model, as it tackles the same problem from a different perspective. Experiments validate our claim, with a combined model outperforming both single models.

The idea of modeling coherence with discourse relations and formulating it in a discourse role matrix can also be applied to other NLP tasks. We plan to apply our methodology to other tasks, such as summarization, text generation and essay scoring, which also need to produce and assess discourse coherence.

References

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: an entity-based approach. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 141–148, Morristown, NJ, USA. Association for Computational Linguistics.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34:1–34, March.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceedings of the Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics Annual Meeting 2004*.

Micha Elsner, Joseph Austerweil, and Eugene Charniak. 2007. A unified local and global model for discourse coherence. In *Proceedings of the Conference on Human Language Technology and North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2007)*, Rochester, New York, USA, April.

Robert Elwell and Jason Baldridge. 2008. Discourse connective argument identification with connective specific rankers. In *Proceedings of the IEEE International Conference on Semantic Computing (ICSC 2010)*, Washington, DC, USA.

Barbara J. Grosz, Scott Weinstein, and Aravind K. Joshi. 1995. Centering: a framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225, June.

Thorsten Joachims. 1999. Making large-scale support vector machine learning practical. In Bernhard

Schlkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 169–184. MIT Press, Cambridge, MA, USA.

Nikiforos Karamanis. 2007. Supplementing entity coherence with local rhetorical relations for information ordering. *Journal of Logic, Language and Information*, 16:445–464, October.

Mirella Lapata and Regina Barzilay. 2005. Automatic evaluation of text coherence: Models and representations. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, UK.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the Penn Discourse Treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*, Singapore.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2010. A PDTB-styled end-to-end discourse parser. Technical Report TRB8/10, School of Computing, National University of Singapore, August.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Daniel Marcu. 1996. Distinguishing between coherent and incoherent texts. In *The Proceedings of the Student Conference on Computational Linguistics in Montreal*, pages 136–143.

Jane Morris and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17:21–48, March.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, Singapore.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind Joshi. 2008. Easily identifiable discourse relations. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008) Short Papers*, Manchester, UK.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP 2009)*, Singapore.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse Treebank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*.

Radu Soricut and Daniel Marcu. 2006. Discourse generation using utility-trained coherence models. In *Proceedings of the COLING/ACL Main Conference Poster Sessions*, pages 803–810, Morristown, NJ, USA. Association for Computational Linguistics.

WenTing Wang, Jian Su, and Chew Lim Tan. 2010. Kernel based discourse relation recognition with temporal ordering information. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, Uppsala, Sweden, July.

Bonnie Webber. 2004. D-LTAG: Extending lexicalized TAG to discourse. *Cognitive Science*, 28(5):751–779.

Ben Wellner and James Pustejovsky. 2007. Automatically identifying the arguments of discourse connectives. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, Prague, Czech Republic.

Underspecifying and Predicting Voice for Surface Realisation Ranking

Sina Zarrieß, Aoife Cahill and Jonas Kuhn
Institut für maschinelle Sprachverarbeitung
Universität Stuttgart, Germany
{sina.zarriess,aoife.cahill,jonas.kuhn}@ims.uni-stuttgart.de

Abstract

This paper addresses a data-driven surface realisation model based on a large-scale reversible grammar of German. We investigate the relationship between the surface realisation performance and the character of the input to generation, i.e. its degree of underspecification. We extend a syntactic surface realisation system, which can be trained to choose among word order variants, such that the candidate set includes active and passive variants. This allows us to study the interaction of voice and word order alternations in realistic German corpus data. We show that with an appropriately underspecified input, a linguistically informed realisation model trained to regenerate strings from the underlying semantic representation achieves 91.5% accuracy (over a baseline of 82.5%) in the prediction of the original voice.

1 Introduction

This paper[1] presents work on modelling the usage of voice and word order alternations in a free word order language. Given a set of meaning-equivalent candidate sentences, such as in the simplified English Example (1), our model makes predictions about which candidate sentence is most appropriate or natural given the context.

(1) Context: *The Parliament started the debate about the state budget in April.*

a. It wasn't until June that the Parliament approved it.
b. It wasn't until June that it was approved by the Parliament.
c. It wasn't until June that it was approved.

We address the problem of predicting the usage of linguistic alternations in the framework of a *surface realisation ranking* system. Such ranking systems are practically relevant for the real-world application of grammar-based generators that usually generate several grammatical surface sentences from a given abstract input, e.g. (Velldal and Oepen, 2006). Moreover, this framework allows for detailed experimental studies of the interaction of specific linguistic features. Thus it has been demonstrated that for free word order languages like German, word order prediction quality can be improved with carefully designed, linguistically informed models capturing information-structural strategies (Filippova and Strube, 2007; Cahill and Riester, 2009).

This paper is situated in the same framework, using rich linguistic representations over corpus data for machine learning of realisation ranking. However, we go beyond the task of finding the correct ordering for an almost fixed set of word forms. Quite obviously, word order is only one of the means at a speaker's disposal for expressing some content in a contextually appropriate form; we add systematic alternations like the voice alternation (active vs. passive) to the picture. As an alternative way of promoting or demoting the prominence of a syntactic argument, its interaction with word ordering strategies in real corpus data is of high theoretical interest (Aissen, 1999; Aissen, 2003; Bresnan et al., 2001).

Our main goals are (i) to establish a corpus-based surface realisation framework for empirically investigating interactions of voice and word order in German, (ii) to design an input representation for generation capturing voice alternations in a variety of contexts, (iii) to better understand the relationship between the performance of a generation ranking model and the type of realisation candidates available in its input. In working towards these goals, this paper addresses the question of evaluation. We conduct a pilot human evaluation on the voice al-

[1]This work has been supported by the Deutsche Forschungsgemeinschaft (DFG; German Research Foundation) in SFB 732 *Incremental specification in context*, project D2 (PIs: Jonas Kuhn and Christian Rohrer).

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1007–1017,
Portland, Oregon, June 19-24, 2011.

ternation data and relate our findings to our results established in the automatic ranking experiments.

Addressing interactions among a range of grammatical and discourse phenomena on realistic corpus data turns out to be a major methodological challenge for data-driven surface realisation. The set of candidate realisations available for ranking will influence the findings, and here, existing surface realisers vary considerably. Belz et al. (2010) point out the differences across approaches in the type of syntactic and semantic information present and absent in the input representation; and it is the type of underspecification that determines the number (and character) of available candidate realisations and, hence, the complexity of the realisation task.

We study the effect of varying degrees of underspecification explicitly, extending a syntactic generation system by a semantic component capturing voice alternations. In regeneration studies involving underspecified underlying representations, corpus-oriented work reveals an additional methodological challenge. When using standard semantic representations, as common in broad-coverage work in semantic parsing (i.e., from the point of view of analysis), alternative variants for sentence realisation will often receive slightly different representations: In the context of (1), the continuation (1-c) is presumably more natural than (1-b), but with a standard sentence-bounded semantic analysis, only (1-a) and (1-b) would receive equivalent representations.

Rather than waiting for the availability of robust and reliable techniques for detecting the reference of implicit arguments in analysis (or for contextually aware reasoning components), we adopt a relatively simple heuristic approach (see Section 3.1) that approximates the desired equivalences by augmented representations for examples like (1-c). This way we can overcome an extremely skewed distribution in the naturally occurring meaning-equivalent active vs. passive sentences, a factor which we believe justifies taking the risk of occasional overgeneration.

The paper is structured as follows: Section 2 situates our methodology with respect to other work on surface realisation and briefly summarises the relevant theoretical linguistic background. In Section 3, we present our generation architecture and the design of the input representation. Section 4 describes the setup for the experiments in Section 5. In Section 6, we present the results from the human evaluation.

2 Related Work

2.1 Generation Background

The first widely known data-driven approach to surface realisation, or tactical generation, (Langkilde and Knight, 1998) used language-model n-gram statistics on a word lattice of candidate realisations to guide a ranker. Subsequent work explored ways of exploiting linguistically annotated data for trainable generation models (Ratnaparkhi, 2000; Marciniak and Strube, 2005; Belz, 2005, a.o.). Work on data-driven approaches has led to insights into the importance of linguistic features for sentence linearisation decisions (Ringger et al., 2004; Filippova and Strube, 2009). The availability of discriminative learning techniques for the ranking of candidate analyses output by broad-coverage grammars with rich linguistic representations, originally in parsing (Riezler et al., 2000; Riezler et al., 2002), has also led to a revival of interest in linguistically sophisticated reversible grammars as the basis for surface realisation (Velldal and Oepen, 2006; Cahill et al., 2007). The grammar generates candidate analyses for an underlying representation and the ranker's task is to predict the contextually appropriate realisation.

The work that is most closely related to ours is Velldal (2008). He uses an MRS representation derived by an HPSG grammar that can be underspecified for information status. In his case, the underspecification is encoded in the grammar and not directly controlled. In multilingually oriented linearisation work, Bohnet et al. (2010) generate from semantic corpus annotations included in the CoNLL'09 shared task data. However, they note that these annotations are not suitable for full generation since they are often incomplete. Thus, it is not clear to which degree these annotations are actually underspecified for certain paraphrases.

2.2 Linguistic Background

In competition-based linguistic theories (Optimality Theory and related frameworks), the use of argument alternations is construed as an effect of markedness hierarchies (Aissen, 1999; Aissen, 2003). Argument functions (subject, object, ...) on

the one hand and the various properties that argument phrases can bear (person, animacy, definiteness) on the other are organised in markedness hierarchies. Wherever possible, there is a tendency to *align* the hierarchies, i.e., use prominent functions to realise prominently marked argument phrases. For instance, Bresnan et al. (2001) find that there is a statistical tendency in English to passivise a verb if the patient is higher on the person scale than the agent, but an active is grammatically possible.

Bresnan et al. (2007) correlate the use of the English dative alternation to a number of features such as givenness, pronominalisation, definiteness, constituent length, animacy of the involved verb arguments. These features are assumed to reflect the discourse acessibility of the arguments.

Interestingly, the properties that have been used to model argument alternations in strict word order languages like English have been identified as factors that influence word order in free word order languages like German, see Filippova and Strube (2007) for a number of pointers. Cahill and Riester (2009) implement a model for German word order variation that approximates the information status of constituents through morphological features like definiteness, pronominalisation etc. We are not aware of any corpus-based generation studies investigating how these properties relate to argument alternations in free word order languages.

3 Generation Architecture

Our data-driven methodology for investigating factors relevant to surface realisation uses a regeneration set-up[2] with two main components: a) a grammar-based component used to parse a corpus sentence and map it to all its meaning-equivalent surface realisations, b) a statistical ranking component used to select the correct, i.e. contextually most appropriate surface realisation. Two variants of this set-up that we use are sketched in Figure 1.

We generally use a hand-crafted, broad-coverage LFG for German (Rohrer and Forst, 2006) to parse a corpus sentence into a f(unctional) structure[3] and generate all surface realisations from a given

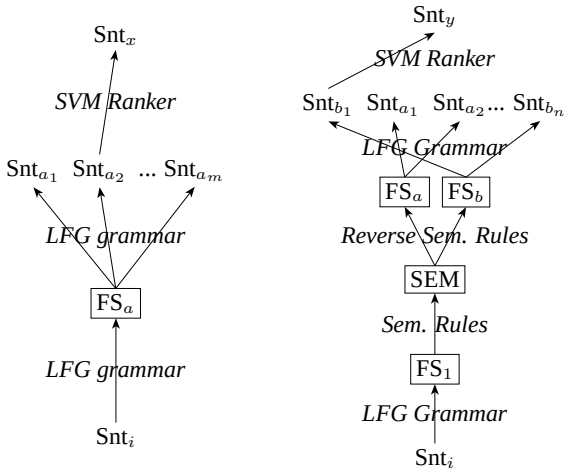


Figure 1: Generation pipelines

f-structure, following the generation approach of Cahill et al. (2007). F-structures are attribute-value matrices representing grammatical functions and morphosyntactic features; their theoretical motivation lies in the abstraction over details of surface realisation. The grammar is implemented in the XLE framework (Crouch et al., 2006), which allows for reversible use of the same declarative grammar in the parsing and generation direction.

To obtain a more abstract underlying representation (in the pipeline on the right-hand side of Figure 1), the present work uses an additional semantic construction component (Crouch and King, 2006; Zarrieß, 2009) to map LFG f-structures to meaning representations. For the reverse direction, the meaning representations are mapped to f-structures which can then be mapped to surface strings by the XLE generator (Zarrieß and Kuhn, 2010).

For the final realisation ranking step in both pipelines, we used SVMrank, a Support Vector Machine-based learning tool (Joachims, 1996). The ranking step is thus technically independent from the LFG-based component. However, the grammar is used to produce the training data, pairs of corpus sentences and the possible alternations.

The two pipelines allow us to vary the degree to which the generation input is underspecified. An f-structure abstracts away from word order, i.e. the candidate set will contain just word order alternations. In the semantic input, syntactic function and voice are underspecified, so a larger set of surface realisation candidates is generated. Figure 2 illustrates the two representation levels for an active and

[2]Compare the bidirectional competition set-up in some Optimality-Theoretic work, e.g., (Kuhn, 2003).

[3]The choice among alternative f-structures is done with a discriminative model (Forst, 2007).

a passive sentence. The subject of the passive and the object of the active f-structure are mapped to the same role (patient) in the meaning representation.

3.1 Issues with "naive" underspecification

In order to create an underspecified voice representation that does indeed leave open the realisation options available to the speaker/writer, it is often not sufficient to remove just the syntactic function information. For instance, the subject of the active sentence (2) is an arbitrary reference pronoun *man "one"* which cannot be used as an oblique agent in a passive, sentence (2-b) is ungrammatical.

(2) a. Man hat den Kanzler gesehen.
One has the chancellor seen.
b. *Der Kanzler wurde von man gesehen.
The chancellor was by one seen.

So, when combined with the grammar, the meaning representation for (2) in Figure 2 contains implicit information about the voice of the original corpus sentence; the candidate set will not include any passive realisations. However, a passive realisation without the oblique agent in the *by*-phrase, as in Example (3), is a very natural variant.

(3) Der Kanzler wurde gesehen.
The chancellor was seen.

The reverse situation arises frequently too: passive sentences where the agent role is not overtly realised. Given the standard, "analysis-oriented" meaning representation for Sentence (4) in Figure 2, the realiser will not generate an active realisation since the agent role cannot be instantiated by any phrase in the grammar. However, depending on the exact context there are typically options for realising the subject phrase in an active with very little descriptive content.

Ideally, one would like to account for these phenomena in a meaning representation that underspecifies the lexicalisation of discourse referents, and also captures the reference of implicit arguments. Especially the latter task has hardly been addressed in NLP applications (but see Gerber and Chai (2010)). In order to work around that problem, we implemented some simple heuristics which underspecify the realisation of certain verb arguments. These rules define: 1. a set of pronouns (generic and neutral pronouns, universal quantifiers) that correspond to "trivial" agents in active and implicit agents in passive sentences; 2. a set of prepositional adjuncts in passive sentences that correspond to subjects in active sentence (e.g. causative and instrumental prepositions like *durch* "*by means of*"); 3. certain syntactic contexts where special underspecification devices are needed, e.g. coordinations or embeddings, see Zarrieß and Kuhn (2010) for examples. In the following, we will distinguish 1-role transitives where the agent is "trivial" or implicit from 2-role transitives with a non-implicit agent.

	Active	Passive
2-role trans.	71% (82%)	10% (2%)
1-role trans.	11% (0%)	8% (16%)

Table 1: Distribution of voices in SEM_h (SEM_n)

By means of the extended underspecification rules for voice, the sentences in (2) and (3) receive an identical meaning representation. As a result, our surface realiser can produce an active alternation for (3) and a passive alternation for (2). In the following, we will refer to the extended representations as SEM_h ("heuristic semantics"), and to the original representations as SEM_n ("naive semantics").

We are aware of the fact that these approximations introduce some noise into the data and do not always represent the underlying referents correctly. For instance, the implicit agent in a passive need not be "trivial" but can correspond to an actual discourse referent. However, we consider these heuristics as a first step towards capturing an important discourse function of the passive alternation, namely the deletion of the agent role. If we did not treat the passives with an implicit agent on a par with certain actives, we would have to ignore a major portion of the passives occurring in corpus data.

Table 1 summarises the distribution of the voices for the heuristic meaning representation SEM_h on the data-set we will introduce in Section 4, with the distribution for the naive representation SEM_n in parentheses.

4 Experimental Set-up

Data To obtain a sizable set of realistic corpus examples for our experiments on voice alternations, we created our own dataset of input sentences and representations, instead of building on treebank examples as Cahill et al. (2007) do. We extracted 19,905 sentences, all containing at least one transitive verb,

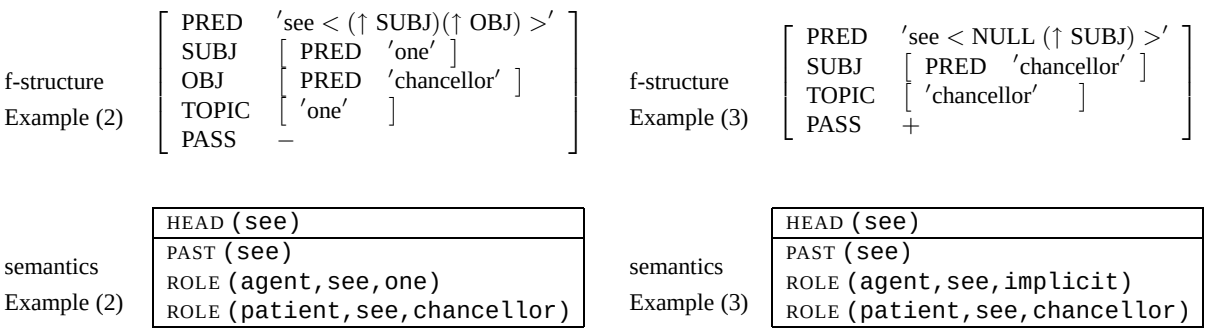


Figure 2: F-structure pair for passive-active alternation

from the HGC, a huge German corpus of newspaper text (204.5 million tokens). The sentences are automatically parsed with the German LFG grammar. The resulting f-structure parses are transferred to meaning representations and mapped back to f-structure charts. For our generation experiments, we only use those f-structure charts that the XLE generator can map back to a set of surface realisations. This results in a total of 1236 test sentences and 8044 sentences in our training set. The data loss is mostly due to the fact the XLE generator often fails on incomplete parses, and on very long sentences. Nevertheless, the average sentence length (17.28) and number of surface realisations (see Table 2) are higher than in Cahill et al. (2007).

Labelling For the training of our ranking model, we have to tell the learner how closely each surface realisation candidate resembles the original corpus sentence. We distinguish the rank categories: "1" identical to the corpus string, "2" identical to the corpus string ignoring punctuation, "3" small edit distance (< 4) to the corpus string ignoring punctuation, "4" different from the corpus sentence. In one of our experiments (Section 5.1), we used the rank category "5" to explicitly label the surface realisations derived from the alternation f-structure that does not correspond to the parse of the original corpus sentence. The intermediate rank categories "2" and "3" are useful since the grammar does not always regenerate the exact corpus string, see Cahill et al. (2007) for explanation.

Features The linguistic theories sketched in Section 2.2 correlate morphological, syntactic and semantic properties of constituents (or discourse referents) with their order and argument realisation. In our system, this correlation is modelled by a combination of linguistic properties that can be extracted from the f-structure or meaning representation and of the surface order that is read off the sentence string. Standard n-gram features are also used as features.[4] The feature model is built as follows: for every lemma in the f-structure, we extract a set of morphological properties (definiteness, person, pronominal status etc.), the voice of the verbal head, its syntactic and semantic role, and a set of informations status features following Cahill and Riester (2009). These properties are combined in two ways: a) Precedence features: relative order of properties in the surface string, e.g. "theme $<$ agent in passive", "1st person $<$ 3rd person"; b) "scale alignment" features (ScalAl.): combinations of voice and role properties with morphological properties, e.g. "subject is singular", "agent is 3rd person in active voice" (these are surface-independent, identical for each alternation candidate).

The model for which we present our results is based on sentence-internal features only; as Cahill and Riester (2009) showed, these feature carry a considerable amount of implicit information about the discourse context (e.g. in the shape of referring expressions). We also implemented a set of explicitly inter-sentential features, inspired by Centering Theory (Grosz et al., 1995). This model did not improve over the intra-sentential model.

Evaluation Measures In order to assess the general quality of our generation ranking models, we

[4]The language model is trained on the German data release for the 2009 ACL Workshop on Machine Translation shared task, 11,991,277 total sentences.

		FS	SEM$_n$	SEM$_h$
Avg. # strings		36.7	68.2	75.8
Random Match		16.98	10.72	7.28
LM	Match	15.45	15.04	11.89
	BLEU	0.68	0.68	0.65
	NIST	13.01	12.95	12.69
Ling. Model	Match	27.91	27.66	26.38
	BLEU	0.764	0.759	0.747
	NIST	13.18	13.14	13.01

Table 2: Evaluation of Experiment 1

use several standard measures: a) exact match: how often does the model select the original corpus sentence, b) BLEU: n-gram overlap between top-ranked and original sentence, c) NIST: modification of BLEU giving more weight to less frequent n-grams. Second, we are interested in the model's performance wrt. specific linguistic criteria. We report the following accuracies: d) Voice: how often does the model select a sentence realising the correct voice, e) Precedence: how often does the model generate the right order of the verb arguments (agent and patient), and f) Vorfeld: how often does the model correctly predict the verb arguments to appear in the sentence initial position before the finite verb, the so-called *Vorfeld*. See Sections 5.3 and 6 for a discussion of these measures.

5 Experiments

5.1 Exp. 1: Effect of Underspecified Input

We investigate the effect of the input's underspecification on a state-of-the-art surface realisation ranking model. This model implements the entire feature set described in Section 4 (it is further analysed in the subsequent experiments). We built 3 datasets from our alternation data: FS - candidates generated from the f-structure; SEM$_n$ - realisations from the naive meaning representations; SEM$_h$ - candidates from the heuristically underspecified meaning representation. Thus, we keep the set of original corpus sentences (=the target realisations) constant, but train and test the model on different candidate sets.

In Table 2, we compare the performance of the linguistically informed model described in Section 4 on the candidates sets against a random choice and a language model (LM) baseline. The differences in BLEU between the candidate sets and models are statistically significant.[5] In general, the linguistic model largely outperforms the LM and is less sensitive to the additional confusion introduced by the SEM$_h$ input. Its BLEU score and match accuracy decrease only slightly (though statistically significantly).

		FS	SEM$_n$	SEM$_h$	SEM$_n*$
All Trans.	Voice Acc.	100	98.06	91.05	97.59
	Voice Spec.	100	22.8	0	0
	Majority BL		82.4		98.1
2-role Trans.	Voice Acc.	100	97.7	91.8	97.59
	Voice Spec.	100	8.33	0	0
	Majority BL		88.5		98.1
1-role Trans.	Voice Acc.	100	100	90.0	-
	Voice Spec.	100	100	0	-
	Majority BL		53.9		-

Table 3: Accuracy of Voice Prediction by Ling. Model in Experiment 1

In Table 3, we report the performance of the linguistic model on the different candidate sets with respect to voice accuracy. Since the candidate sets differ in the proportion of items that underspecify the voice (see "Voice Spec." in Table 3), we also report the accuracy on the SEM$_n*$ test set, which is a subset of SEM$_n$ excluding the items where the voice is specified. Table 3 shows that the proportion of active realisations for the SEM$_n*$ input is very high, and the model does not outperform the majority baseline (which always selects active). In contrast, the SEM$_h$ model clearly outperforms the majority baseline.

Example (4) is a case from our development set where the SEM$_n$ model incorrectly predicts an active (4-a), and the SEM$_h$ correctly predicts a passive (4-b).

(4) a. 26 kostspielige Studien erwähnten die Finanzierung.
26 expensive studies mentioned the funding.

b. Die Finanzierung wurde von 26 kostspieligen Studien erwähnt.
The funding was by 26 expensive studies mentioned.

This prediction is according to the markedness hierarchy: the patient is singular and definite, the agent

[5] According to a bootstrap resampling test, $p < 0.05$

Features	Match	BLEU	Voice	Prec.	VF
Prec.	16.3	0.70	88.43	64.1	59.1
ScalAl.	10.4	0.64	90.37	58.9	56.3
Union	26.4	0.75	91.50	80.2	70.9

Table 4: Evaluation of Experiment 2

is plural and indefinite. Counterexamples are possible, but there is a clear statistical preference – which the model was able to pick up.

On the one hand, the rankers can cope surprisingly well with the additional realisations obtained from the meaning representations. According to the global sentence overlap measures, their quality is not seriously impaired. On the other hand, the design of the representations has a substantial effect on the prediction of the alternations. The SEM$_n$ does not seem to learn certain preferences because of the extremely imbalanced distribution in the input data. This confirms the hypothesis sketched in Section 3.1, according to which the degree of the input's underspecification can crucially change the behaviour of the ranking model.

5.2 Exp. 2: Word Order and Voice

We examine the impact of certain feature types on the prediction of the variation types in our data. We are particularly interested in the interaction of voice and word order (precedence) since linguistic theories (see Section 2.2) predict similar information-structural factors guiding their use, but usually do not consider them in conjunction.

In Table 4, we report the performance of ranking models trained on the different feature subsets introduced in Section 4. The union of the features corresponds to the model trained on SEM$_h$ in Experiment 1. At a very broad level, the results suggest that the precedence and the scale alignment features interact both in the prediction of voice and word order.

The most pronounced effect on voice accuracy can be seen when comparing the precedence model to the union model. Adding the surface-independent scale alignment features to the precedence features leads to a big improvement in the prediction of word order. This is not a trivial observation since a) the surface-independent features do not discriminate between the word orders and b) the precedence features are built from the same properties (see Section 4). Thus, the SVM learner discovers dependencies between relative precedence preferences and abstract properties of a verb argument which cannot be encoded in the precedence alone.

It is worth noting that the precedence features improve the voice prediction. This indicates that wherever the application context allows it, voice should not be specified at a stage prior to word order. Example (5) is taken from our development set, illustrating a case where the union model predicted the correct voice and word order (5-a), and the scale alignment model top-ranked the incorrect voice and word order. The active verb arguments in (5-b) are both case-ambigous and placed in the non-canonical order (object $<$ subject), so the semantic relation can be easily misunderstood. The passive in (5-a) is unambiguous since the agent is realised in a PP (and placed in the Vorfeld).

(5) a. Von den deutschen Medien wurden die Ausländer
By the German media were the foreigners
nur erwähnt, wenn es Zoff gab.
only mentioned, when there trouble was.

b. Wenn es Zoff gab, erwähnten die Ausländer
When there trouble was, mentioned the foreigners
nur die deutschen Medien.
only the German media.

Moreover, our results confirm Filippova and Strube (2007) who find that it is harder to predict the correct Vorfeld occupant in a German sentence, than to predict the relative order of the constituents.

5.3 Exp. 3: Capturing Flexible Variation

The previous experiment has shown that there is a certain inter-dependence between word order and voice. This experiment addresses this interaction by varying the way the training data for the ranker is labelled. We contrast two ways of labelling the sentences (see Section 4): a) all sentences that are not (nearly) identical to the reference sentence have the rank category "4", irrespective of their voice (referred to as unlabelled model), b) the sentences that do not realise the correct voice are ranked lower than sentences with the correct voice ("4" vs. "5"), referred to as labelled model. Intuitively, the latter way of labelling tells the ranker that all sentences in the incorrect voice are worse than all sentences in the correct voice, independent of the word order. Given the first labelling strategy, the ranker can decide in an unsupervised way which combinations of word order and voice are to be preferred.

Model	Match	BLEU	NIST	Top 1 Voice	Top 1 Prec.	Top 1 Prec.+Voice	Top 2 Prec.+Voice	Top 3 Prec.+Voice
Labelled, no LM	21.52	0.73	12.93	91.9	76.25	71.01	78.35	82.31
Unlabelled, no LM	26.83	0.75	13.01	91.5	80.19	74.51	84.28	88.59
Unlabeled + LM	27.35	0.75	13.08	91.5	79.6	73.92	79.74	82.89

Table 5: Evaluation of Experiment 3

In Table 5, it can be seen that the unlabelled model improves over the labelled on all the sentence overlap measures. The improvements are statistically significant. Moreover, we compare the n-best accuracies achieved by the models for the joint prediction of voice and argument order. The unlabelled model is very flexible with respect to the word order-voice interaction: the accuracy dramatically improves when looking at the top 3 sentences. Table 5 also reports the performance of an unlabelled model that additionally integrates LM scores. Surprisingly, these scores have a very small positive effect on the sentence overlap features and no positive effect on the voice and precedence accuracy. The n-best evaluations even suggest that the LM scores negatively impact the ranker: the accuracy for the top 3 sentences increases much less as compared to the model that does not integrate LM scores.[6]

The n-best performance of a realisation ranker is practically relevant for re-ranking applications such as Velldal (2008). We think that it is also conceptually interesting. Previous evaluation studies suggest that the original corpus sentence is not always the only optimal realisation of a given linguistic input (Cahill and Forst, 2010; Belz and Kow, 2010). Humans seem to have varying preferences for word order contrasts in certain contexts. The n-best evaluation could reflect the behaviour of a ranking model with respect to the range of variations encountered in real discourse. The pilot human evaluation in the next Section deals with this question.

6 Human Evaluation

Our experiment in Section 5.3 has shown that the accuracy of our linguistically informed ranking model dramatically increases when we consider the three best sentences rather than only the top-ranked sentence. This means that the model sometimes predicts almost equal naturalness for different voice realisations. Moreover, in the case of word order, we know from previous evaluation studies, that humans sometimes prefer different realisations than the original corpus sentences. This Section investigates agreement in human judgements of voice realisation.

Whereas previous studies in generation mainly used human evaluation to compare different systems, or to correlate human and automatic evaluations, our primary interest is the agreement or correlation between human rankings. In particular, we explore the hypothesis that this agreement is higher in certain contexts than in others. In order to select these contexts, we use the predictions made by our ranking model.

The questionnaire for our experiment comprised 24 items falling into 3 classes: a) items where the 3 best sentences predicted by the model have the same voice as the original sentence ("Correct"), b) items where the 3 top-ranked sentences realise different voices ("Mixed"), c) items where the model predicted the incorrect voice in all 3 top sentences ("False"). Each item is composed of the original sentence, the 3 top-ranked sentences (if not identical to the corpus sentence) and 2 further sentences such that each item contains different voices. For each item, we presented the previous context sentence.

The experiment was completed by 8 participants, all native speakers of German, 5 had a linguistic background. The participants were asked to rank each sentence on a scale from 1-6 according to its naturalness and plausibility in the given context. The participants were explicitly allowed to use the same rank for sentences they find equally natural. The participants made heavy use of this option: out of the 192 annotated items, only 8 are ranked such that no two sentences have the same rank.

We compare the human judgements by correlat-

[6](Nakanishi et al., 2005) also note a negative effect of including LM scores in their model, pointing out that the LM was not trained on enough data. The corpus used for training our LM might also have been too small or distinct in genre.

ing them with Spearman's ρ. This measure is considered appropriate for graded annotation tasks in general (Erk and McCarthy, 2009), and has also been used for analysing human realisation rankings (Velldal, 2008; Cahill and Forst, 2010). We normalise the ranks according to the procedure in Velldal (2008). In Table 6, we report the correlations obtained from averaging over all pairwise correlations between the participants and the correlations restricted to the item and sentence classes. We used bootstrap re-sampling on the pairwise correlations to test that the correlations on the different item classes significantly differ from each other.

The correlations in Table 6 suggest that the agreement between annotators is highest on the false items, and lowest on the mixed items. Humans tended to give the best rank to the original sentence more often on the false items (91%) than on the others. Moreover, the agreement is generally higher on the sentences realising the correct voice.

These results seem to confirm our hypothesis that the general level of agreement between humans differs depending on the context. However, one has to be careful in relating the effects in our data solely to voice preferences. Since the sentences were chosen automatically, some examples contain very unnatural word orders that probably guided the annotators' decisions more than the voice. This is illustrated by Example (6) showing two passive sentences from our questionnaire which differ only in the position of the adverb *besser "better"*. Sentence (6-a) is completely implausible for a native speaker of German, whereas Sentence (6-b) sounds very natural.

(6) a. Durch das neue Gesetz sollen **besser**
By the new law should better
Eigenheimbesitzer geschützt werden.
house owners protected be.

b. Durch das neue Gesetz sollen Eigenheimbesitzer
By the new law should house owners
besser geschützt werden.
better protected be.

This observation brings us back to our initial point that the surface realisation task is especially challenging due to the interaction of a range of semantic and discourse phenomena. Obviously, this interaction makes it difficult to single out preferences for a specific alternation type. Future work will have to establish how this problem should be dealt with in the design of human evaluation experiments.

	Items			
	All	Correct	Mixed	False
"All" sent.	0.58	0.6	0.54	0.62
"Correct" sent.	0.64	0.63	0.56	0.72
"False" sent.	0.47	0.57	0.48	0.44
Top-ranked corpus sent.	84%	78%	83%	91%

Table 6: Human Evaluation

7 Conclusion

We have presented a grammar-based generation architecture which implements the surface realisation of meaning representations abstracting from voice and word order. In order to be able to study voice alternations in a variety of contexts, we designed heuristic underspecification rules which establish, for instance, the alternation relation between an active with a generic agent and a passive that does not overtly realise the agent. This strategy leads to a better balanced distribution of the alternations in the training data, such that our linguistically informed generation ranking model achieves high BLEU scores and accurately predicts active and passive. In future work, we will extend our experiments to a wider range of alternations and try to capture inter-sentential context more explicitly. Moreover, it would be interesting to carry over our methodology to a purely statistical linearisation system where the relation between an input representation and a set of candidate realisations is not so clearly defined as in a grammar-based system.

Our study also addressed the interaction of different linguistic variation types, i.e. word order and voice, by looking at different types of linguistic features and exploring different ways of labelling the training data. However, our SVM-based learning framework is not well-suited to directly assess the correlation between a certain feature (or feature combination) and the occurrence of an alternation. Therefore, it would be interesting to relate our work to the techniques used in theoretical papers, e.g. (Bresnan et al., 2007), where these correlations are analysed more directly.

References

Judith Aissen. 1999. Markedness and subject choice in optimality theory. *Natural Language and Linguistic Theory*, 17(4):673–711.

Judith Aissen. 2003. Differential Object Marking: Iconicity vs. Economy. *Natural Language and Linguistic Theory*, 21:435–483.

Anja Belz and Eric Kow. 2010. Comparing rating scales and preference judgements in language evaluation. In *Proceedings of the 6th International Natural Language Generation Conference (INLG'10)*.

Anja Belz, Mike White, Josef van Genabith, Deirdre Hogan, and Amanda Stent. 2010. Finding common ground: Towards a surface realisation shared task. In *Proceedings of the 6th International Natural Language Generation Conference (INLG'10)*.

Anja Belz. 2005. Statistical generation: Three methods compared and evaluated. In *Proceedings of Tenth European Workshop on Natural Language Generation (ENLG-05)*, pages 15–23.

Bernd Bohnet, Leo Wanner, Simon Mill, and Alicia Burga. 2010. Broad coverage multilingual deep sentence generation with a stochastic multi-level realizer. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, Beijing, China.

Joan Bresnan, Shipra Dingare, and Christopher D. Manning. 2001. Soft Constraints Mirror Hard Constraints: Voice and Person in English and Lummi. In *Proceedings of the LFG '01 Conference*.

Joan Bresnan, Anna Cueni, Tatiana Nikitina, and Harald Baayen. 2007. Predicting the Dative Alternation. In G. Boume, I. Kraemer, and J. Zwarts, editors, *Cognitive Foundations of Interpretation*. Amsterdam: Royal Netherlands Academy of Science.

Aoife Cahill and Martin Forst. 2010. Human Evaluation of a German Surface Realisation Ranker. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 112 – 120, Athens, Greece. Association for Computational Linguistics.

Aoife Cahill and Arndt Riester. 2009. Incorporating Information Status into Generation Ranking. In *Proceedings of the 47th Annual Meeting of the ACL*, pages 817–825, Suntec, Singapore, August. Association for Computational Linguistics.

Aoife Cahill, Martin Forst, and Christian Rohrer. 2007. Stochastic realisation ranking for a free word order language. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, pages 17–24, Saarbrücken, Germany, June. DFKI GmbH. Document D-07-01.

Dick Crouch and Tracy Holloway King. 2006. Semantics via F-Structure Rewriting. In Miriam Butt and Tracy Holloway King, editors, *Proceedings of the LFG06 Conference*.

Dick Crouch, Mary Dalrymple, Ron Kaplan, Tracy King, John Maxwell, and Paula Newman. 2006. XLE Documentation. Technical report, Palo Alto Research Center, CA.

Katrin Erk and Diana McCarthy. 2009. Graded Word Sense Assignment. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 440 – 449, Singapore.

Katja Filippova and Michael Strube. 2007. Generating constituent order in German clauses. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 07)*, Prague, Czech Republic.

Katja Filippova and Michael Strube. 2009. Tree linearization in English: Improving language model based approaches. In *Companion Volume to the Proceedings of Human Language Technologies Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 09, short).*, Boulder, Colorado.

Martin Forst. 2007. Filling Statistics with Linguistics – Property Design for the Disambiguation of German LFG Parses. In *ACL 2007 Workshop on Deep Linguistic Processing*, pages 17–24, Prague, Czech Republic, June. Association for Computational Linguistics.

Matthew Gerber and Joyce Chai. 2010. Beyond nombank: A study of implicit argumentation for nominal predicates. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Barbara J. Grosz, Aravind Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Thorsten Joachims. 1996. Training linear svms in linear time. In M. Butt and T. H. King, editors, *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, CSLI Proceedings Online.

Jonas Kuhn. 2003. *Optimality-Theoretic Syntax—A Declarative Approach*. CSLI Publications, Stanford, CA.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the ACL/COLING-98*, pages 704–710, Montreal, Quebec.

Tomasz Marciniak and Michael Strube. 2005. Using an annotated corpus as a knowledge source for language generation. In *Proceedings of Workshop on Using Corpora for Natural Language Generation*, pages 19–24, Birmingham, UK.

Hiroko Nakanishi, Yusuke Miyao, and Junichi Tsujii. 2005. Probabilistic models for disambiguation of an

HPSG-based chart generator. In *Proceedings of IWPT 2005*.

Adwait Ratnaparkhi. 2000. Trainable methods for surface natural language generation. In *Proceedings of NAACL 2000*, pages 194–201, Seattle, WA.

Stefan Riezler, Detlef Prescher, Jonas Kuhn, and Mark Johnson. 2000. Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and EM training. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'00), Hong Kong*, pages 480–487.

Stefan Riezler, Dick Crouch, Ron Kaplan, Tracy King, John Maxwell, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02), Pennsylvania, Philadelphia*.

Eric K. Ringger, Michael Gamon, Robert C. Moore, David Rojas, Martine Smets, and Simon Corston-Oliver. 2004. Linguistically Informed Statistical Models of Constituent Structure for Ordering in Sentence Realization. In *Proceedings of the 2004 International Conference on Computational Linguistics*, Geneva, Switzerland.

Christian Rohrer and Martin Forst. 2006. Improving coverage and parsing quality of a large-scale LFG for German. In *Proceedings of LREC-2006*.

Erik Velldal and Stephan Oepen. 2006. Statistical ranking in tactical generation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia.

Erik Velldal. 2008. *Empirical Realization Ranking*. Ph.D. thesis, University of Oslo, Department of Informatics.

Sina Zarrieß and Jonas Kuhn. 2010. Reversing F-structure Rewriting for Generation from Meaning Representations. In *Proceedings of the LFG10 Conference*, Ottawa.

Sina Zarrieß. 2009. Developing German Semantics on the basis of Parallel LFG Grammars. In *Proceedings of the 2009 Workshop on Grammar Engineering Across Frameworks (GEAF 2009)*, pages 10–18, Suntec, Singapore, August. Association for Computational Linguistics.

Recognizing Authority in Dialogue with an Integer Linear Programming Constrained Model

Elijah Mayfield
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
elijah@cmu.edu

Carolyn Penstein Rosé
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
cprose@cs.cmu.edu

Abstract

We present a novel computational formulation of speaker *authority* in discourse. This notion, which focuses on how speakers position themselves relative to each other in discourse, is first developed into a reliable coding scheme (0.71 agreement between human annotators). We also provide a computational model for automatically annotating text using this coding scheme, using supervised learning enhanced by constraints implemented with Integer Linear Programming. We show that this constrained model's analyses of speaker authority correlates very strongly with expert human judgments (r^2 coefficient of 0.947).

1 Introduction

In this work, we seek to formalize the ways speakers position themselves in discourse. We do this in a way that maintains a notion of discourse structure, and which can be aggregated to evaluate a speaker's overall stance in a dialogue. We define the body of work in positioning to include any attempt to formalize the processes by which speakers attempt to influence or give evidence of their relations to each other. Constructs such as Initiative and Control (Whittaker and Stenton, 1988), which attempt to operationalize the authority over a discourse's structure, fall under the umbrella of positioning. As we construe positioning, it also includes work on detecting certainty and confusion in speech (Liscombe et al., 2005), which models a speaker's understanding of the information in their statements. Work in dialogue act tagging is also relevant, as it seeks to describe the actions and moves with which speakers display these types of positioning (Stolcke et al., 2000).

To complement these bodies of work, we choose to focus on the question of how speakers position themselves as *authoritative* in a discourse. This means that we must describe the way speakers introduce new topics or discussions into the discourse; the way they position themselves relative to that topic; and how these functions interact with each other. While all of the tasks mentioned above focus on specific problems in the larger rhetorical question of speaker positioning, none explicitly address this framing of authority. Each does have valuable ties to the work that we would like to do, and in section 2, we describe prior work in each of those areas, and elaborate on how each relates to our questions.

We measure this as an *authoritativeness ratio*. Of the contentful dialogue moves made by a speaker, in what fraction of those moves is the speaker positioned as the primary authority on that topic? To measure this quantitatively, we introduce the Negotiation framework, a construct from the field of systemic functional linguistics (SFL), which addresses specifically the concepts that we are interested in. We present a reproducible formulation of this sociolinguistics research in section 3, along with our preliminary findings on reliability between human coders, where we observe inter-rater agreement of 0.71. Applying this coding scheme to data, we see strong correlations with important motivational constructs such as Self-Efficacy (Bandura, 1997) as well as learning gains.

Next, we address automatic coding of the Negotiation framework, which we treat as a two-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1018–1026,
Portland, Oregon, June 19-24, 2011.

dimensional classification task. One dimension is a set of codes describing the authoritative status of a contribution[1]. The other dimension is a segmentation task. We impose constraints on both of these models based on the structure observed in the work of SFL. These constraints are formulated as boolean statements describing what a correct label sequence looks like, and are imposed on our model using an Integer Linear Programming formulation (Roth and Yih, 2004). In section 5, this model is evaluated on a subset of the MapTask corpus (Anderson et al., 1991) and shows a high correlation with human judgements of authoritativeness (r^2 = 0.947). After a detailed error analysis, we will conclude the paper in section 6 with a discussion of our future work.

2 Background

The Negotiation framework, as formulated by the SFL community, places a special emphasis on how speakers function in a discourse as sources or recipients of information or action. We break down this concept into a set of codes, one code per contribution. Before we break down the coding scheme more concretely in section 3, it is important to understand why we have chosen to introduce a new framework, rather than reusing existing computational work.

Much work has examined the emergence of discourse structure from the choices speakers make at the linguistic and intentional level (Grosz and Sidner, 1986). For instance, when a speaker asks a question, it is expected to be followed with an answer. In discourse analysis, this notion is described through dialogue games (Carlson, 1983), while conversation analysis frames the structure in terms of adjacency pairs (Schegloff, 2007). These expectations can be viewed under the umbrella of conditional relevance (Levinson, 2000), and the exchanges can be labelled *discourse segments*.

In prior work, the way that people influence discourse structure is described through the two tightly-related concepts of *initiative* and *control*. A speaker who begins a discourse segment is said to have initiative, while control accounts for which speaker is being addressed in a dialogue (Whittaker and Stenton, 1988). As initiative passes back and forth between discourse participants, control over the conversation similarly transfers from one speaker to another (Walker and Whittaker, 1990). This relation is often considered synchronous, though evidence suggests that the reality is not straightforward (Jordan and Di Eugenio, 1997).

Research in initiative and control has been applied in the form of mixed-initiative dialogue systems (Smith, 1992). This is a large and active field, with applications in tutorial dialogues (Core, 2003), human-robot interactions (Peltason and Wrede, 2010), and more general approaches to effective turn-taking (Selfridge and Heeman, 2010). However, that body of work focuses on influencing discourse structure through positioning. The question that we are asking instead focuses on how speakers view their authority as a source of information about the topic of the discourse.

In particular, consider questioning in discourse. In mixed-initiative analysis of discourse, asking a question always gives you control of a discourse. There is an expectation that your question will be followed by an answer. A speaker might already know the answer to a question they asked - for instance, when a teacher is verifying a student's knowledge. However, in most cases asking a question represents a lack of authority, treating the other speakers as a source for that knowledge. While there have been preliminary attempts to separate out these specific types of positioning in initiative, such as Chu-Carroll and Brown (1998), it has not been studied extensively in a computational setting.

Another similar thread of research is to identify a speaker's *certainty*, that is, the confidence of a speaker and how that self-evaluation affects their language (Pon-Barry and Shieber, 2010). Substantial work has gone into automatically identifying levels of speaker certainty, for example in Liscombe et al. (2005) and Litman et al. (2009). The major difference between our work and this body of literature is that work on certainty has rarely focused on how state translates into interaction between speakers (with some exceptions, such as the application of certainty to tutoring dialogues (Forbes-Riley and Litman, 2009)). Instead, the focus is on the person's self-evaluation, independent of the influence on the speaker's positioning within a discourse.

Dialogue act tagging seeks to describe the moves people make to express themselves in a discourse.

[1] We treat each line in our corpus as a single contribution.

This task involves defining the role of each contribution based on its function (Stolcke et al., 2000). We know that there are interesting correlations between these acts and other factors, such as learning gains (Litman and Forbes-Riley, 2006) and the relevance of a contribution for summarization (Wrede and Shriberg, 2003). However, adapting dialogue act tags to the question of how speakers position themselves is not straightforward. In particular, the granularity of these tagsets, which is already a highly debated topic (Popescu-Belis, 2008), is not ideal for the task we have set for ourselves. Many dialogue acts can be used in authoritative or non-authoritative ways, based on context, and can position a speaker as either giver or receiver of information. Thus these more general tagsets are not specific enough to the role of authority in discourse.

Each of these fields of prior work is highly valuable. However, none were designed to specifically describe how people present themselves as a source or recipient of knowledge in a discourse. Thus, we have chosen to draw on a different field of sociolinguistics. Our formalization of that theory is described in the next section.

3 The Negotiation Framework

We now present the Negotiation framework[2], which we use to answer the questions left unanswered in the previous section. Within the field of SFL, this framework has been continually refined over the last three decades (Berry, 1981; Martin, 1992; Martin, 2003). It attempts to describe how speakers use their role as a source of knowledge or action to position themselves relative to others in a discourse.

Applications of the framework include distinguishing between focus on teacher knowledge and student reasoning (Veel, 1999) and distribution of authority in juvenile trials (Martin et al., 2008). The framework can also be applied to problems similar to those studied through the lens of initiative, such as the distinction between authority over discourse structure and authority over content (Martin, 2000).

A challenge of applying this work to language technologies is that it has historically been highly qualitative, with little emphasis placed on reproducibility. We have formulated a pared-down, reproducible version of the framework, presented in Section 3.1. Evidence of the usefulness of that formulation for identifying authority, and of correlations that we can study based on these codes, is presented briefly in Section 3.2.

3.1 Our Formulation of Negotiation

The codes that we can apply to a contribution using the Negotiation framework are comprised of four main codes, K1, K2, A1, and A2, and two additional codes, ch and o. This is a reduction over the many task-specific or highly contextual codes used in the original work. This was done to ensure that a machine learning classification task would not be overwhelmed with many infrequent classes.

The main codes are divided by two questions. First, is the contribution related to exchanging information, or to exchanging services and actions? If the former, then it is a K move (knowledge); if the latter, then an A move (action). Second, is the contribution acting as a primary actor, or secondary? In the case of knowledge, this often correlates to the difference between assertions (K1) and queries (K2). For instance, a statement of fact or opinion is a K1:

g	K1	well i've got a great viewpoint here just below the east lake

By contrast, asking for someone else's knowledge or opinion is a K2:

g	K2	what have you got underneath the east lake
f	K1	rocket launch

In the case of action, the codes usually correspond to narrating action (A1) and giving instructions (A2), as below:

g	A2	go almost to the edge of the lake
f	A1	yeah

A challenge move (ch) is one which directly contradicts the content or assertion of the previous line, or makes that previous contribution irrelevant. For instance, consider the exchange below, where an instruction is rejected because its presuppositions are broken by the challenging statement.

g	A2	then head diagonally down towards the bottom of the dead tree
f	ch	i have don't have a dead tree i have a dutch elm

[2] All examples are drawn from the MapTask corpus and involve an instruction giver (g) and follower (f). Within examples, discourse segment boundaries are shown by horizontal lines.

All moves that do not fit into one of these categories are classified as other (o). This includes back-channel moves, floor-grabbing moves, false starts, and any other non-contentful contributions.

This theory makes use of discourse segmentation. Research in the SFL community has focused on intra-segment structure, and empirical evidence from this research has shown that exchanges between speakers follow a very specific pattern:

o* X2? o* X1+ o*

That is to say, each segment contains a primary move (a K1 or an A1) and an optional preceding secondary move, with other non-contentful moves interspersed throughout. A single statement of fact would be a K1 move comprising an entire segment, while a single question/answer pair would be a K2 move followed by a K1. Longer exchanges of many lines obviously also occur.

We iteratively developed a coding manual which describes, in a reproducible way, how to apply the codes listed above. The six codes we use, along with their frequency in our corpus, are given in Table 1. In the next section, we evaluate the reliability and utility of hand-coded data, before moving on to automation in section 4.

3.2 Preliminary Evaluation

This coding scheme was evaluated for reliability on two corpora using Cohen's kappa (Cohen, 1960). Within the social sciences community, a kappa above 0.7 is considered acceptable. Two conversations were each coded by hand by two trained annotators. The first conversation was between three students in a collaborative learning task; inter-rater reliability kappa for Negotiation labels was 0.78. The second conversation was from the MapTask corpus, and kappa was 0.71. Further data was labelled by hand by one trained annotator.

In our work, we label conversations using the coding scheme above. To determine how well these codes correlate with other interesting factors, we choose to assign a quantitative measure of authoritativeness to each speaker. This measure can then be compared to other features of a speaker. To do this, we use the coded labels to assign an *Authoritativeness Ratio* to each speaker. First, we define a function $A(S, c, L)$ for a speaker, a contribution, and a set of labels $L \subseteq \{K1, K2, A1, A2, o, ch\}$ as:

Code	Meaning	Count	Percent
K1	Primary Knower	984	22.5
K2	Secondary Knower	613	14.0
A1	Primary Actor	471	10.8
A2	Secondary Actor	708	16.2
ch	Challenge	129	2.9
o	Other	1469	33.6
	Total	4374	100.0

Table 1: The six codes in our coding scheme, along with their frequency in our corpus of twenty conversations.

$$A(S, c, L) = \begin{cases} 1 & c \text{ spoken by } S \text{ with label } l \in L \\ 0 & \text{otherwise.} \end{cases}$$

We then define the Authoritativeness ratio $Auth(S)$ for a speaker S in a dialogue consisting of contributions $c_1 ... c_n$ as:

$$Auth(S) = \frac{\sum_{i=1}^{n} A(S, c_i, \{K1, A2\})}{\sum_{i=1}^{n} A(S, c_i, \{K1, K2, A1, A2\})}$$

The intuition behind this ratio is that we are only interested in the four main label types in our analysis - at least for an initial description of authority, we do not consider the non-contentful o moves. Within these four main labels, there are clearly two that appear "dominant" - statements of fact or opinion, and commands or instructions - and two that appear less dominant - questions or requests for information, and narration of an action. We sum these together to reach a single numeric value for each speaker's projection of authority in the dialogue.

The full details of our external validations of this approach are available in Howley et al. (2011). To summarize, we considered two data sets involving student collaborative learning. The first data set consisted of pairs of students interacting over two days, and was annotated for aggressive behavior, to assess warning factors in social interactions. Our analysis

showed that aggressive behavior correlated with authoritativeness ratio ($p < .05$), and that less aggressive students became less authoritative in the second day ($p < .05$, effect size $.15\sigma$). The second data set was analyzed for Self-Efficacy - the confidence of each student in their own ability (Bandura, 1997) - as well as actual learning gains based on pre- and post-test scores. We found that the Authoritativeness ratio was a significant predictor of learning gains ($r^2 = .41$, $p < .04$). Furthermore, in a multiple regression, we determined that the Authoritativeness ratio of both students in a group predict the average Self-Efficacy of the pair ($r^2 = .12$, $p < .01$).

4 Computational Model

We know that our coding scheme is useful for making predictions about speakers. We now judge whether it can be reproduced fully automatically. Our model must select, for each contribution c_i in a dialogue, the most likely classification label l_i from $\{K1, K2, A1, A2, o, ch\}$. We also build in parallel a segmentation model to select s_i from the set $\{new, same\}$. Our baseline approach to both problems is to use a bag-of-words model of the contribution, and use machine learning for classification.

Certain types of interactions, explored in section 4.1, are difficult or impossible to classify without context. We build a contextual feature space, described in section 4.2, to enhance our baseline bag-of-words model. We can also describe patterns that appear in discourse segments, as detailed in section 3.1. In our coding manual, these instructions are given as rules for how segments should be coded by humans. Our hypothesis is that by enforcing these rules in the output of our automatic classifier, performance will increase. In section 4.3 we formalize these constraints using Integer Linear Programming.

4.1 Challenging cases

We want to distinguish between phenomena such as in the following two examples.

f	K2	so I'm like on the bank on the bank of the east lake
g	K1	yeah

In this case, a one-token contribution is indisputably a K1 move, answering a yes/no question. However, in the dialogue below, it is equally inarguable that the same move is an A1:

g	A2	go almost to the edge of the lake
f	A1	yeah

Without this context, these moves would be indistinguishable to a model. With it, they are both easily classified correctly.

We also observed that markers for segmentation of a segment vary between contentful initiations and non-contentful ones. For instance, filler noises can often initiate segments:

g	o	hmm...
g	K2	do you have a farmer's gate?
f	K1	no

Situations such as this are common. This is also a challenge for segmentation, as demonstrated below:

g	K1	oh oh it's on the right-hand side of my great viewpoint
f	o	okay yeah
g	o	right eh
g	A2	go almost to the edge of the lake
f	A1	yeah

A long statement or instruction from one speaker is followed up with a terse response (in the same segment) from the listener. However, after that backchannel move, a short floor-grabbing move is often made to start the next segment. This is a distinction that a bag-of-words model would have difficulty with. This is markedly different from contentful segment initiations:

g	A2	come directly down below the stone circle and we come up
f	ch	I don't have a stone circle
g	o	you don't have a stone circle

All three of these lines look like statements, which often initiate new segments. However, only the first should be marked as starting a new segment. The other two are topically related, in the second line by contradicting the instruction, and in the third by repeating the previous person's statement.

4.2 Contextual Feature Space Additions

To incorporate the insights above into our model, we append features to our bag-of-words model. First, in our classification model we include both lexical bigrams and part-of-speech bigrams to encode further lexical knowledge and some notion of syntactic structure. To account for restatements and topic shifts, we add a feature based on cosine similarity (using term vectors weighted by TF-IDF calculated

over training data). We then add a feature for the predicted label of the previous contribution - after each contribution is classified, the next contribution adds a feature for the automatic label. This requires our model to function as an on-line classifier.

We build two segmentation models, one trained on contributions of less than four tokens, and another trained on contributions of four or more tokens, to distinguish between characteristics of contentful and non-contentful contributions. To the short-contribution model, we add two additional features. The first represents the ratio between the length of the current contribution and the length of the previous contribution. The second represents whether a change in speaker has occurred between the current and previous contribution.

4.3 Constraints using Integer Linear Programming

We formulate our constraints using Integer Linear Programming (ILP). This formulation has an advantage over other sequence labelling formulations, such as Viterbi decoding, in its ability to enforce structure through constraints. We then enhance this classifier by adding constraints, which allow expert knowledge of discourse structure to be enforced in classification. We can use these constraints to eliminate label options which would violate the rules for a segment outlined in our coding manual.

Each classification decision is made at the contribution level, jointly optimizing the Negotiation label and segmentation label for a single contribution, then treating those labels as given for the next contribution classification.

To define our objective function for optimization, for each possible label, we train a one vs. all SVM, and use the resulting regression for each label as a score, giving us six values $\vec{l_i}$ for our Negotiation label and two values $\vec{s_i}$ for our segmentation label. Then, subject to the constraints below, we optimize:

$$\arg\max_{l \in \vec{l_i}, s \in \vec{s_i}} l + s$$

Thus, at each contribution, if the highest-scoring Negotiation label breaks a constraint, the model can optimize whether to drop to the next-most-likely label, or start a new segment.

Recall from section 3.1 that our discourse segments follow strict rules related to ordering and repetition of contributions. Below, we list the constraints that we used in our model to enforce that pattern, along with a brief explanation of the intuition behind each.

$$\begin{aligned} &\forall c_i \in s, (l_i = K2) \Rightarrow \\ &\forall j < i, c_j \in t \Rightarrow (l_j \neq K1) \end{aligned} \tag{1}$$

$$\begin{aligned} &\forall c_i \in s, (l_i = A2) \Rightarrow \\ &\forall j < i, c_j \in t \Rightarrow (l_j \neq A1) \end{aligned} \tag{2}$$

The first constraints enforce the rule that a primary move cannot occur before a secondary move in the same segment. For instance, a question must initiate a new segment if it follows a statement.

$$\begin{aligned} &\forall c_i \in s, (l_i \in \{A1, A2\}) \Rightarrow \\ &\forall j < i, c_j \in s \Rightarrow (l_j \notin \{K1, K2\}) \end{aligned} \tag{3}$$

$$\begin{aligned} &\forall c_i \in s, (l_i \in \{K1, K2\}) \Rightarrow \\ &\forall j < i, c_j \in s \Rightarrow (l_j \notin \{A1, A2\}) \end{aligned} \tag{4}$$

These constraints specify that A moves and K moves cannot cooccur in a segment. An instruction for action and a question requesting information must be considered separate segments.

$$\begin{aligned} &\forall c_i \in s, (l_i = A1) \Rightarrow ((l_{i-1} = A1) \vee \\ &\forall j < i, c_j \in s \Rightarrow (l_j \neq A1)) \end{aligned} \tag{5}$$

$$\begin{aligned} &\forall c_i \in s, (l_i = K1) \Rightarrow ((l_{i-1} = K1) \vee \\ &\forall j < i, c_j \in s \Rightarrow (l_j \neq K1)) \end{aligned} \tag{6}$$

This pair states that two primary moves cannot occur in the same segment unless they are contiguous, in rapid succession.

$$\begin{aligned} &\forall c_i \in s, (l_i = A1) \Rightarrow \\ &\forall j < i, c_j \in s, (l_j = A2) \Rightarrow (S_i \neq S_j) \end{aligned} \tag{7}$$

$$\begin{aligned} &\forall c_i \in s, (l_i = K1) \Rightarrow \\ &\forall j < i, c_j \in s, (l_j = K2) \Rightarrow (S_i \neq S_j) \end{aligned} \tag{8}$$

The last set of constraints enforce the intuitive notion that a speaker cannot follow their own secondary move with a primary move in that segment (such as answering their own question).

Computationally, an advantage of these constraints is that they do not extend past the current segment in history. This means that they usually are only enforced over the past few moves, and do not enforce any global constraint over the structure of the whole dialogue. This allows the constraints to be flexible to various conversational styles, and tractable for fast computation independent of the length of the dialogue.

5 Evaluation

We test our models on a twenty conversation subset of the MapTask corpus detailed in Table 1. We compare the use of four models in our results.

- **Baseline:** This model uses a bag-of-words feature space as input to an SVM classifier. No segmentation model is used and no ILP constraints are enforced.
- **Baseline+ILP:** This model uses the baseline feature space as input to both classification and segmentation models. ILP constraints are enforced between these models.
- **Contextual:** This model uses our enhanced feature space from section 4.2, with no segmentation model and no ILP constraints enforced.
- **Contextual+ILP:** This model uses the enhanced feature spaces for both Negotiation labels and segment boundaries from section 4.2 to enforce ILP constraints.

For segmentation, we evaluate our models using exact-match accuracy. We use multiple evaluation metrics to judge classification. The first and most basic is accuracy - the percentage of accurately chosen Negotiation labels. Secondly, we use Cohen's Kappa (Cohen, 1960) to judge improvement in accuracy over chance. The final evaluation is the r^2 coefficient computed between predicted and actual Authoritativeness ratios per speaker. This represents how much variance in authoritativeness is accounted for in the predicted ratios. This final metric is the most important for measuring reproducibility of human analyses of speaker authority in conversation.

We use SIDE for feature extraction (Mayfield and Rosé, 2010), SVM-Light for machine learning (Joachims, 1999), and Learning-Based Java for ILP inference (Rizzolo and Roth, 2010). Performance is evaluated by 20-fold cross-validation, where each fold is trained on 19 conversations and tested on the remaining one. Statistical significance was calculated using a student's paired t-test. For accuracy and kappa, $n = 20$ (one data point per conversation) and for r^2, $n = 40$ (one data point per speaker).

Model	**Accuracy**	**Kappa**	r^2
Baseline	59.7%	0.465	0.354
Baseline+ILP	61.6%	0.488	0.663
Segmentation	*72.3%*		
Contextual	66.7%	0.565	0.908
Contextual+ILP	68.4%	0.584	0.947
Segmentation	*74.9%*		

Table 2: Performance evaluation for our models. Each line is significantly improved in both accuracy and r^2 error from the previous line ($p < .01$).

5.1 Results

All classification results are given in Table 2 and charts showing correlation between predicted and actual speaker Authoritativeness ratios are shown in Figure 1. We observe that the baseline bag-of-words model performs well above random chance (kappa of 0.465); however, its accuracy is still very low and its ability to predict Authoritativeness ratio of a speaker is not particularly high (r^2 of 0.354 with ratios from manually labelled data). We observe a significant improvement when ILP constraints are applied to this model.

The contextual model described in section 4.2 performs better than our baseline constrained model. However, the gains found in the contextual model are somewhat orthogonal to the gains from using ILP constraints, as applying those constraints to the contextual model results in further performance gains (and a high r^2 coefficient of 0.947).

Our segmentation model was evaluated based on exact matches in boundaries. Switching from baseline to contextual features, we observe an improvement in accuracy of 2.6%.

5.2 Error Analysis

An error analysis of model predictions explains the large effect on correlation despite relatively smaller

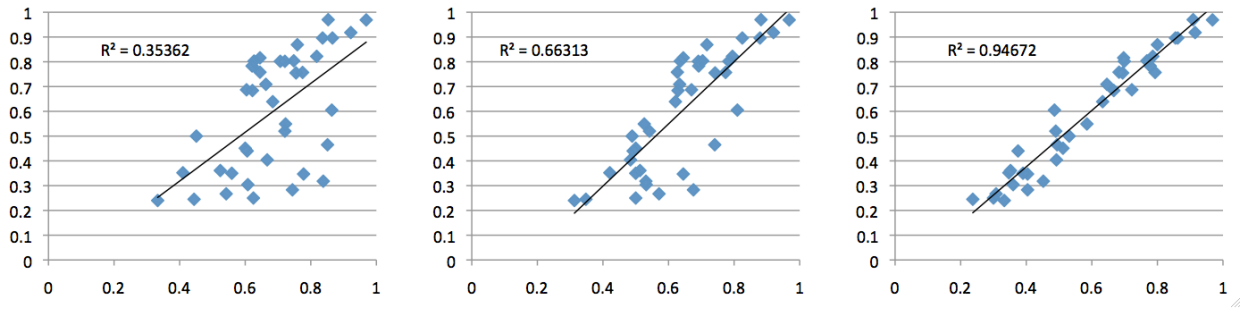


Figure 1: Plots of predicted (x axis) and actual (y axis) Authoritativeness ratios for speakers across 20 conversations, for the Baseline (left), Baseline+Constraints (center), and Contextual+Constraints (right) models.

changes in accuracy. Our Authoritativeness ratio does not take into account moves labelled o or ch. What we find is that the most advanced model still makes many mistakes at determining whether a move should be labelled as o or a core move. This error rate is, however, fairly consistent across the four core move codes. When a move is determined (correctly) to not be an o move, the system is highly accurate in distinguishing between the four core labels.

The one systematic confusion that continues to appear most frequently in our results is the inability to distinguish between a segment containing an $A2$ move followed by an $A1$ move, and a segment containing a $K1$ move followed by an o move. The surface structure of these types of exchanges is very similar. Consider the following two exchanges:

g	A2	if you come down almost to the bottom of the map that I've got
f	A1	uh-huh

f	K1	but the meadow's below my broken gate
g	o	right yes

These two exchanges on a surface level are highly similar. Out of context, making this distinction is very hard even for human coders, so it is not surprising then that this pattern is the most difficult one to recognize in this corpus. It contributes most of the remaining confusion between the four core codes.

6 Conclusions

In this work we have presented one formulation of authority in dialogue. This formulation allows us to describe positioning in discourse in a way that is complementary to prior work in mixed-initiative dialogue systems and analysis of speaker certainty. Our model includes a simple understanding of discourse structure while also encoding information about the types of moves used, and the certainty of a speaker as a source of information. This formulation is reproducible by human coders, with an inter-rater reliability of 0.71.

We have then presented a computational model for automatically applying these codes per contribution. In our best model, we see a good 68.4% accuracy on a six-way individual contribution labelling task. More importantly, this model replicates human analyses of authoritativeness very well, with an r^2 coefficient of 0.947.

There is room for improvement in our model in future work. Further use of contextual features will more thoroughly represent the information we want our model to take into account. Our segmentation accuracy is also fairly low, and further examination of segmentation accuracy using a more sophisticated evaluation metric, such as WindowDiff (Pevzner and Hearst, 2002), would be helpful.

In general, however, we now have an automated model that is reliable in reproducing human judgments of authoritativeness. We are now interested in how we can apply this to the larger questions of positioning we began this paper by asking, especially in describing speaker positioning at various instants throughout a single discourse. This will be the main thrust of our future work.

Acknowledgements

This research was supported by NSF grants SBE-0836012 and HCC-0803482.

References

Anne Anderson, Miles Bader, Ellen Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, et al. 1991. The HCRC Map Task Corpus. In *Language and Speech*.

Albert Bandura. 1997. *Self-efficacy: The Exercise of Control*

Margaret Berry. 1981. Towards Layers of Exchange Structure for Directive Exchanges. In *Network* 2.

Lauri Carlson. 1983. *Dialogue Games: An Approach to Discourse Analysis*.

Jennifer Chu-Carroll and Michael Brown. 1998. An Evidential Model for Tracking Initiative in Collaborative Dialogue Interactions. In *User Modeling and User-Adapted Interaction*.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. In *Educational and Psychological Measurement*.

Mark Core and Johanna Moore and Claus Zinn. 2003. The Role of Initiative in Tutorial Dialogue. In *Proceedings of EACL*.

Kate Forbes-Riley and Diane Litman. 2009. Adapting to Student Uncertainty Improves Tutoring Dialogues. In *Proceedings of Artificial Intelligence in Education*.

Barbara Grosz and Candace Sidner. 1986. Attention, Intentions, and the Structure of Discourse. In *Computational Linguistics*.

Iris Howley and Elijah Mayfield and Carolyn Penstein Rosé. 2011. Missing Something? Authority in Collaborative Learning. In *Proceedings of Computer-Supported Collaborative Learning*.

Thorsten Joachims. 1999. Making large-Scale SVM Learning Practical. In *Advances in Kernel Methods - Support Vector Learning*.

Pamela Jordan and Barbara Di Eugenio. 1997. Control and Initiative in Collaborative Problem Solving Dialogues. In *Proceedings of AAAI Spring Symposium on Computational Models for Mixed Initiative Interactions*.

Stephen Levinson. 2000. *Pragmatics*.

Jackson Liscombe, Julia Hirschberg, and Jennifer Venditti. 2005. Detecting Certainness in Spoken Tutorial Dialogues. In *Proceedings of Interspeech*.

Diane Litman and Kate Forbes-Riley. 2006. Correlations betweeen Dialogue Acts and Learning in Spoken Tutoring Dialogue. In *Natural Language Engineering*.

Diane Litman, Mihai Rotaru, and Greg Nicholas. 2009. Classifying Turn-Level Uncertainty Using Word-Level Prosody. In *Proceedings of Interspeech*.

James Martin. 1992. *English Text: System and Structure.*

James Martin. 2000. Factoring out Exchange: Types of Structure. In *Working with Dialogue*.

James Martin and David Rose. 2003. *Working with Discourse: Meaning Beyond the Clause*.

James Martin, Michele Zappavigna, and Paul Dwyer. 2008. Negotiating Shame: Exchange and Genre Structure in Youth Justice Conferencing. In *Proceedings of European Systemic Functional Linguistics*.

Elijah Mayfield and Carolyn Penstein Rosé. 2010. An Interactive Tool for Supporting Error Analysis for Text Mining. In *Proceedings of Demo Session at NAACL*.

Julia Peltason and Britta Wrede. 2010. Modeling Human-Robot Interaction Based on Generic Interaction Patterns. In *AAAI Report on Dialog with Robots*.

Lev Pevzner and Marti Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. In *Computational Linguistics*.

Heather Pon-Barry and Stuart Shieber. 2010. Assessing Self-awareness and Transparency when Classifying a Speakers Level of Certainty. In *Speech Prosody*.

Andrei Popescu-Belis. 2008. Dimensionality of Dialogue Act Tagsets: An Empirical Analysis of Large Corpora. In *Language Resources and Evaluation*.

Nick Rizzolo and Dan Roth. 2010. Learning Based Java for Rapid Development of NLP Systems. In *Language Resources and Evaluation*.

Dan Roth and Wen-Tau Yih. 2004. A Linear Programming Formulation for Global Inference in Natural Language Tasks. In *Proceedings of CoNLL*.

Emanuel Schegloff. 2007. *Sequence Organization in Interaction: A Primer in Conversation Analysis*.

Ethan Selfridge and Peter Heeman. 2010. Importance-Driven Turn-Bidding for Spoken Dialogue Systems. In *Proceedings of ACL*.

Ronnie Smith. 1992. A computational model of expectation-driven mixed-initiative dialog processing. Ph.D. Dissertation.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, et al. 2000. Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech. In *Computational Linguistics*.

Robert Veel. 1999. Language, Knowledge, and Authority in School Mathematics. In *Pedagogy and the Shaping of Consciousness: Linguistics and Social Processes*

Marilyn Walker and Steve Whittaker. 1990. Mixed Initiative in Dialogue: An Investigation into Discourse Structure. In *Proceedings of ACL*.

Steve Whittaker and Phil Stenton. 1988. Cues and Control in Expert-Client Dialogues. In *Proceedings of ACL*.

Britta Wrede and Elizabeth Shriberg. 2003. The Relationship between Dialogue Acts and Hot Spots in Meetings. In *IEEE Workshop on Automatic Speech Recognition and Understanding*.

Reordering Metrics for MT

Alexandra Birch a.birch@ed.ac.uk
Miles Osborne miles@inf.ed.ac.uk
University of Edinburgh
10 Crichton Street
Edinburgh, EH8 9AB, UK

Abstract

One of the major challenges facing statistical machine translation is how to model differences in word order between languages. Although a great deal of research has focussed on this problem, progress is hampered by the lack of reliable metrics. Most current metrics are based on matching lexical items in the translation and the reference, and their ability to measure the quality of word order has not been demonstrated. This paper presents a novel metric, the LRscore, which explicitly measures the quality of word order by using permutation distance metrics. We show that the metric is more consistent with human judgements than other metrics, including the BLEU score. We also show that the LRscore can successfully be used as the objective function when training translation model parameters. Training with the LRscore leads to output which is preferred by humans. Moreover, the translations incur no penalty in terms of BLEU scores.

1 Introduction

Research in machine translation has focused broadly on two main goals, improving word choice and improving word order in translation output. Current machine translation metrics rely upon indirect methods for measuring the quality of the word order, and their ability to capture the quality of word order is poor (Birch et al., 2010).

There are currently two main approaches to evaluating reordering. The first is exemplified by the BLEU score (Papineni et al., 2002), which counts the number of matching n-grams between the reference and the hypothesis. Word order is captured by the proportion of longer n-grams which match. This method does not consider the position of matching words, and only captures ordering differences if there is an exact match between the words in the translation and the reference. Another approach is taken by two other commonly used metrics, METEOR (Banerjee and Lavie, 2005) and TER (Snover et al., 2006). They both search for an alignment between the translation and the reference, and from this they calculate a penalty based on the number of differences in order between the two sentences. When block moves are allowed the search space is very large, and matching stems and synonyms introduces errors. Importantly, none of these metrics capture the distance by which words are out of order. Also, they conflate reordering performance with the quality of the lexical items in the translation, making it difficult to tease apart the impact of changes. More sophisticated metrics, such as the RTE metric (Padó et al., 2009), use higher level syntactic or semantic analysis to determine the grammaticality of the output. These approaches require annotation and can be very slow to run. For most research, shallow metrics are more appropriate.

We introduce a novel shallow metric, the Lexical Reordering Score (LRscore), which explicitly measures the quality of word order in machine translations and interpolates it with a lexical metric. This results in a simple, decomposable metric which makes it easy for researchers to pinpoint the effect of their changes. In this paper we show that the LRscore is more consistent with human judgements

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1027–1035,
Portland, Oregon, June 19-24, 2011.

than other metrics for five out of eight different language pairs. We also apply the LRscore during Minimum Error Rate Training (MERT) to see whether information on reordering allows the translation model to produce better reorderings. We show that humans prefer the output of systems trained with the LRscore 52.5% as compared to 43.9% when training with the BLEU score. Furthermore, training with the LRscore does not result in lower BLEU scores.

The rest of the paper proceeds as follows. Section 2 describes the reordering and lexical metrics that are used and how they are combined. Section 3 presents the experiments on consistency with human judgements and describes how to train the language independent parameter of the LRscore. Section 4 reports the results of the experiments on MERT. Finally we discuss related work and conclude.

2 The LRscore

In this section we present the LRscore which measures reordering using permutation distance metrics. These reordering metrics have been demonstrated to correlate strongly with human judgements of word order quality (Birch et al., 2010). The LRscore combines the reordering metrics with lexical metrics to provide a complete metric for evaluating machine translations.

2.1 Reordering metrics

The relative ordering of words in the source and target sentences is encoded in alignments. We can interpret alignments as permutations which allows us to apply research into metrics for ordered encodings to measuring and evaluating reorderings. We use distance metrics over permutations to evaluate reordering performance. Figure 1 shows three permutations. Each position represents a source word and each value indicates the relative positions of the aligned target words. In Figure 1 (a) represents the identity permutation, which would result from a monotone alignment, (b) represents a small reordering consisting of two words whose orders are inverted, and (c) represents a large reordering where the two halves of the sentence are inverted in the target.

A translation can potentially have many valid word orderings. However, we can be reasonably certain that the ordering of the reference sentence must be acceptable. We therefore compare the ordering of a translation with that of the reference sentence. Where multiple references exist, we select the closest, i.e. the one that gives the best score. The underlying assumption is that most reasonable word orderings should be fairly similar to the reference, which is a necessary assumption for all automatic machine translation metrics.

(a) (1 2 3 4 5 6 7 8 9 10)
(b) (1 2 3 4 •6 •5 •7 8 9 10)
(c) (6 7 8 9 10 •1 2 3 4 5)

Figure 1. Three permutations: (a) monotone (b) with a small reordering and (b) with a large reordering. Bullet points highlight non-sequential neighbours.

Permutations encode one-one relations, whereas alignments contain null alignments and one-many, many-one and many-many relations. We make some simplifying assumptions to allow us to work with permutations. Source words aligned to null are assigned the target word position immediately after the target word position of the previous source word. Where multiple source words are aligned to the same target word or phrase, a many-to-one relation, the target ordering is assumed to be monotone. When one source word is aligned to multiple target words, a one-to-many relation, the source word is assumed to be aligned to the first target word. These simplifications are chosen so as to reduce the alignment to a bijective relationship without introducing any extraneous reorderings, i.e. they encode a basic monotone ordering assumption.

We choose permutation distance metrics which are sensitive to the number of words that are out of order, as humans are assumed to be sensitive to the number of words that are out of order in a sentence. The two permutations we refer to, π and σ, are the source-reference permutation and the source-translation permutation. The metrics are normalised so that 0 means that the permutations are completely inverted, and 1 means that they are identical. We report these scores as percentages.

2.1.1 Hamming Distance

The Hamming distance (Hamming, 1950) measures the number of disagreements between two permutations. It is defined as follows:

$$d_h(\pi, \sigma) = 1 - \frac{\sum_{i=1}^{n} x_i}{n}, x_i = \begin{cases} 0 \text{ if } \pi(i) = \sigma(i) \\ 1 \text{ otherwise} \end{cases}$$

Eg.	BLEU	METEOR	TER	d_h	d_k
(a)	100.0	100.0	100.0	100.0	100.0
(b)	61.8	86.9	90.0	80.0	85.1
(c)	81.3	92.6	90.0	0.0	25.5

Table 1. Metric scores for examples in Figure 1 which are calculated by comparing the permutations to the identity. All metrics are adjusted so that 100 is the best score and 0 the worst.

where n is the length of the permutation. The Hamming distance is the simplest permutation distance metric and is useful as a baseline. It has no concept of the relative ordering of words.

2.1.2 Kendall's Tau Distance

Kendall's tau distance is the minimum number of transpositions of two *adjacent* symbols necessary to transform one permutation into another (Kendall, 1938). It represents the percentage of pairs of elements which share the same order between two permutations. It is defined as follows:

$$
\begin{aligned}
d_k(\pi,\sigma) &= 1 - \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} z_{ij}}{Z}} \\
\text{where } z_{ij} &= \begin{cases} 1 \text{ if } \pi(i) < \pi(j) \text{ and } \sigma(i) > \sigma(j) \\ 0 \text{ otherwise} \end{cases} \\
Z &= \frac{(n^2-n)}{2}
\end{aligned}
$$

Kendalls tau seems particularly appropriate for measuring word order differences as the relative ordering words is taken into account. However, most human and machine ordering differences are much closer to monotone than to inverted. The range of values of Kendall's tau is therefore too narrow and close to 1. For this reason we take the square root of the standard metric. This adjusted d_k is also more correlated with human judgements of reordering quality (Birch et al., 2010).

We use the example in Figure 1 to highlight the problem with current MT metrics, and to demonstrate how the permutation distance metrics are calculated. In Table 1 we present the metric results for the example permutations. The metrics are calculated by comparing the permutation string with the monotone permutation. (a) receives the best score for all metrics as it is compared to itself. BLEU and METEOR fail to recognise that (b) represents a small reordering and (c) a large reordering and they assign a lower score to (b). The reason for this is that they are sensitive to breaks in order, but not to the actual word order differences. BLEU matches more n-grams for (c) and consequently assigns it a higher score. METEOR counts the number of blocks that the translation is broken into, in order to align it with the source. (b) is aligned using four blocks, whereas (c) is aligned using only two blocks. TER counts the number of edits, allowing for block shifts, and applies one block shift for each example, resulting in an equal score for (b) and (c). Both the Hamming distance d_h and the Kendall's tau distance d_k correctly assign (c) a worse score than (b). Note that for (c), the Hamming distance was not able to reward the permutation for the correct relative ordering of words within the two large blocks and gave (c) a score of 0, whereas Kendall's tau takes relative ordering into account.

Wong and Kit (2009) also suggest a metric which combines a word choice and a word order component. They propose a type of F-measure which uses a matching function M to calculate precision and recall. M combines the number of matched words, weighted by their *tfidf* importance, with their position difference score, and finally subtracting a score for unmatched words. Including unmatched words in the M function undermines the interpretation of the supposed F-measure. The reordering component is the average difference of absolute and relative word positions which has no clear meaning. This score is not intuitive or easily decomposable and it is more similar to METEOR, with synonym and stem functionality mixed with a reordering penalty, than to our metric.

2.2 Combined Metric

The LRscore consists of a reordering distance metric which is linearly interpolated with a lexical score to form a complete machine translation evaluation metric. The metric is decomposable because the individual lexical and reordering components can be looked at individually. The following formula describes how to calculate the LRscore:

$$LRscore = \alpha R + (1-\alpha)L \qquad (1)$$

The metric contains only one parameter, α, which balances the contribution of the reordering metric, R, and the lexical metric, L. Here we use BLEU as

the lexical metric. R is the average permutation distance metric adjusted by the brevity penalty and it is calculated as follows:

$$R = \frac{\sum_{s \in S} d_s BP_s}{|S|} \quad (2)$$

Where S is a set of test sentences, d_s is the reordering distance for a sentence and BP is the brevity penalty.

The brevity penalty is calculated as:

$$BP = \begin{cases} 1 & \text{if } t > r \\ e^{1-r/t} & \text{if } t \leq r \end{cases} \quad (3)$$

where t is the length of the translation, and r is the closest reference length. If the reference sentence is slightly longer than the translation, then the brevity penalty will be a fraction somewhat smaller than 1. This has the effect of penalising translations that are shorter than the reference. The brevity penalty within the reordering component is necessary as the distance-based metric would provide the same score for a one word translation as it would for a longer monotone translation. R is combined with a system level lexical score.

In this paper we apply the BLEU score as the lexical metric, as it is well known and it measures lexical precision at different n-gram lengths. We experiment with the full BLEU score and the 1-gram BLEU score, BLEU1, which is purely a measure of the precision of the word choice. The 4-gram BLEU score includes some measure of the local reordering success in the precision of the longer n-grams. BLEU is an important baseline, and improving on it by including more reordering information is an interesting result. The lexical component of the system can be any meaningful metric for a particular target language. If a researcher was interested in morphologically rich languages, for example, METEOR could be used. We use the LRscore to return sentence level scores as well system level scores, and when doing so the smoothed BLEU (Lin and Och, 2004) is used.

3 Consistency with Human Judgements

Automatic metrics must be validated by comparing their scores with human judgements. We train the metric parameter to optimise consistency with human preference judgements across different language pairs and then we show that the LRscore is more consistent with humans than other commonly used metrics.

3.1 Experimental Design

Human judgement of rank has been chosen as the official determinant of translation quality for the 2009 Workshop on Machine Translation (Callison-Burch et al., 2009). We used human ranking data from this workshop to evaluate the LRscore. This consisted of German, French, Spanish and Czech translation systems that were run both into and out of English. In total there were 52,265 pairwise rank judgements collected.

Our reordering metric relies upon word alignments that are generated between the source and the reference sentences, and the source and the translated sentences. In an ideal scenario, the translation system outputs the alignments and the reference set can be selected to have gold standard human alignments. However, the data that we use to evaluate metrics does not have any gold standard alignments and we must train automatic alignment models to generate them. We used version two of the Berkeley alignment model (Liang et al., 2006), with the posterior threshold set at 0.5. Our Spanish-, French- and German-English alignment models are trained using Europarl version 5 (Koehn, 2005). The Czech-English alignment model is trained on sections 0-2 of the Czech-English Parallel Corpus, version 0.9 (Bojar and Zabokrtsky, 2009).

The metric scores are calculated for the test set from the 2009 workshop on machine translation. It consists of 2525 sentences in English, French, German, Spanish and Czech. These sentences have been translated by different machine translation systems and the output submitted to the workshop. The system output along with human evaluations can be downloaded from the web[1].

The BLEU score has five parameters, one for each n-gram, and one for the brevity penalty. These parameters are set to a default uniform value of one. METEOR has 3 parameters which have been trained for human judgements of rank (Lavie and Agarwal, 2008). METEOR version 0.7 was used. The other baseline metric used was TER version 0.7.25. We adapt TER by subtracting it from one, so that all

[1] http://www.statmt.org/wmt09/results.html

metric increases mean an improvement in the translation. The TER metric has five parameters which have not been trained.

Using rank judgements, we do not have absolute scores and so we cannot compare translations across different sentences and extract correlation statistics. We therefore use the method adopted in the 2009 workshop on machine translation (Callison-Burch et al., 2009). We ascertained how consistent the automatic metrics were with the human judgements by calculating consistency in the following manner. We take each pairwise comparison of translation output for single sentences by a particular judge, and we recorded whether or not the metrics were consistent with the human rank. I.e. we counted cases where both the metric and the human judge agreed that one system is better than another. We divided this by the total number of pairwise comparisons to get a percentage. We excluded pairs which the human annotators ranked as ties.

	de-en	es-en	fr-en	cz-en
d_k	73.9	80.5	80.4	81.1

Table 2. The average Kendall's tau reordering distance between the test and reference sentences. 100 means monotone thus de-en has the most reordering.

We present a novel method for setting the LRscore parameter. Using multiple language pairs, we train the parameter according to the amount of reordering seen in each test set. The advantage of this approach is that researchers do not need to train the parameter for new language pairs or test domains. They can simply calculate the amount of reordering in the test set and adjust the parameter accordingly. The amount of reordering is calculated as the Kendall's tau distance between the source and the reference sentences as compared to dummy monotone sentences. The amount of reordering for the test sentences is reported in Table 2. German-English shows more reordering than other language pairs as it has a lower d_k score of 73.9. The language independent parameter (θ) is adjusted by applying the reordering amount (d_k) as an exponent. θ is allowed to takes values of between 0 and 1. This works in a similar way to the brevity penalty. With more reordering, the d_k becomes smaller which leads to an increase in the final value of α. α represents the percentage contribution of the reordering component in the LRscore:

$$\alpha = \theta^{d_k} \tag{4}$$

The language independent parameter θ is trained once, over multiple language pairs. This procedure optimises the average of the consistency results across the different language pairs. We use greedy hillclimbing in order to find the optimal setting. As hillclimbing can end up in a local minima, we perform 20 random restarts, and retaining only the parameter value with the best consistency result.

3.2 Results

Table 3 reports the optimal consistency of the LRscore and baseline metrics with human judgements for each language pair. The LRscore variations are named as follows: LR refers to the LRscore, "H" refers to the Hamming distance and "K" to Kendall's tau distance. "B1" and "B4" refer to the smoothed BLEU score with the 1-gram and the complete scores. Table 3 shows that the LRscore is more consistent with human judgement for 5 out of the 8 language pairs. This is an important result which shows that combining lexical and reordering information makes for a stronger metric than the baseline metrics which do not have a strong reordering component.

METEOR is the most consistent for the Czech-English and English-Czech language pairs, which have the least amount of reordering. METEOR lags behind for the language pairs with the most reordering, the German-English and English-German pairs. Here LR-KB4 is the best metric, which shows that metrics which are sensitive to the distance words are out of order are more appropriate for situations with a reasonable amount of reordering.

4 Optimising Translation Models

Automatic metrics are useful for evaluation, but they are essential for training model parameters. In this section we apply the LRscore as the objective function in MERT training (Och, 2003). MERT minimises translation errors according to some automatic evaluation metric while searching for the best parameter settings over the N-best output. A MERT trained model is likely to exhibit the properties that

Metric	de-en	es-en	fr-en	cz-en	en-de	en-es	en-fr	en-cz	ave
METEOR	58.6	58.3	58.3	**59.4**	52.6	55.7	61.2	**55.6**	57.5
TER	53.2	50.1	52.6	47.5	48.6	49.6	58.3	45.8	50.7
BLEU1	56.1	57.0	56.7	52.5	52.1	54.2	62.3	53.3	55.6
BLEU	58.7	55.5	57.7	57.2	54.1	56.7	**63.7**	53.1	57.1
LR-HB1	59.7	**60.0**	58.6	53.2	54.6	55.6	63.7	54.5	57.5
LR-HB4	60.4	57.3	**58.7**	57.2	54.8	**57.3**	63.3	53.8	57.9
LR-KB1	60.4	59.7	58.0	54.0	54.1	54.7	63.4	54.9	57.5
LR-KB4	**61.0**	57.2	58.5	58.6	**54.8**	56.8	63.1	55.0	**58.7**

Table 3. The percentage consistency between human judgements of rank and metrics. The LRscore variations (LR-*) are optimised for average consistency across language pair (shown in right hand column). The bold numbers represent the best consistency score per language pair.

the metric rewards, but will be blind to aspects of translation quality that are not directly captured by the metric. We apply the LRscore in order to improve the reordering performance of a phrase-based translation model.

4.1 Experimental Design

We hypothesise that the LRscore is a good metric for training translation models. We test this by evaluating the output of the models, first with automatic metrics, and then by using human evaluation. We choose to run the experiment with Chinese-English as this language pair has a large amount of medium and long distance reorderings.

4.1.1 Training Setup

The experiments are carried out with Chinese-English data from GALE. We use the official test set of the 2006 NIST evaluation (1994 sentences). For the development test set, we used the evaluation set from the GALE 2008 evaluation (2010 sentences). Both development set and test set have four references. The phrase table was built from 1.727M parallel sentences from the GALE Y2 training data. The phrase-based translation model called MOSES was used, with all the default settings. We extracted phrases as in (Koehn et al., 2003) by running GIZA++ in both directions and merging alignments with the grow-diag-final heuristic. We used the Moses translation toolkit, including a lexicalised reordering model. The SRILM language modelling toolkit (Stolcke, 2002) was used with interpolated Kneser-Ney discounting. There are three separate 3-gram language models trained on the English side of parallel corpus, the AFP part of the Gigaword corpus, and the Xinhua part of the Gigaword corpus. A 4 or 5-gram language model would have led to higher scores for all objective functions, but would not have changed the findings in this paper. We used the MERT code available in the MOSES repository (Bertoldi et al., 2009).

LR-HB1	LR-HB4	LR-KB1	LR-KB4
26.40	07.19	43.33	26.23

Table 4. The parameter setting representing the % impact of the reordering component for the different versions of the LRscore metric.

The reordering metrics require alignments which were created using the Berkeley word alignment package version 1.1 (Liang et al., 2006), with the posterior probability to being 0.5.

We first extracted the LRscore Kendall's tau distance from the monotone for the Chinese-English test set and this value was 66.1%. This is far more reordering than the other language pairs shown in Table 2. We then calculated the optimal parameter setting, using the reordering amount as a power exponent. Table 4 shows the parameter settings we used in the following experiments. The optimal amount of reordering for LR-HB4 is low, but the results show it still makes an important contribution.

4.1.2 Human Evaluation Setup

Human judgements of translation quality are necessary to determine whether humans prefer sentences from models trained with the BLEU score or with the LRscore. There have been some recent studies which have used the online micro-market, Amazons Mechanical Turk, to collect human annotations (Snow et al., 2008; Callison-Burch, 2009). While some of the data generated is very noisy, invalid responses are largely due to a small number of workers (Kittur et al., 2008). We use Mechanical

Turk and we improve annotation quality by collecting multiple judgements, and eliminating workers who do not achieve a certain level of performance on gold standard questions.

We randomly selected a subset of sentences from the test set. We use 60 sentences each for comparing training with BLEU to training with LR-HB4 and with LR-KB4. These sentences were between 15 and 30 words long. Shorter sentences tend to have uninteresting differences, and longer sentences may have many conflicting differences.

Workers were presented with a reference sentence and two translations which were randomly ordered. They were told to compare the translations and select their preferred translation or "Don't Know". Workers were screened to guarantee reasonable judgement quality. 20 sentence pairs were randomly selected from the 120 test units and annotated as gold standard questions. Workers who got less than 60% of these gold questions correct were disqualified and their judgements discarded.

After disagreeing with a gold annotation, a worker is presented with the gold answer and an explanation. This guides the worker on how to perform the task and motivates them to be more accurate. We used the Crowdflower[2] interface to Mechanical Turk, which implements the gold functionality.

Even though experts can disagree on preference judgements, gold standard labels are necessary to weed out the poor standard workers. There were 21 trusted workers who achieved an average accuracy of 91% on the gold. There were 96 untrusted workers who averaged 29% accuracy on the gold. Their judgements were discarded. Three judgements were collected from the trusted workers for each of the 120 test sentences.

4.2 Results

4.2.1 Automatic Evaluation of MERT

In this experiment we demonstrate that the reordering metrics can be used as learning criterion in minimum error rate training to improve parameter estimation for machine translation.

Table 5 reports the average of three runs of MERT training with different objective functions. The lexical metric BLEU is used as an objective function in isolation, and also as part of the LRscore together with the Hamming distance and Kendall's tau distance. We test with these metrics, and we also report the TER and METEOR scores for comparison.

Metrics / Obj.Func.	BLEU	LR-HB4	LR-KB4	TER	MET.
BLEU	**31.1**	32.1	41.0	60.7	55.5
LRHB4	**31.1**	**32.2**	**41.3**	60.6	55.7
LRKB4	31.0	**32.2**	41.2	**61.0**	**55.8**

Table 5. Average results of three different MERT runs for different objective functions.

The first thing we note in Table 5 is that we would expect the highest scores when training with the same metric as that used for evaluation as MERT maximises the objective function on the development data set. Here, however, when testing with BLEU, we see that training with BLEU and with LR-HB4 leads to equally high BLEU scores. The reordering component is more discerning than the BLEU score. It reliably increases as the word order approaches that of the reference, whereas BLEU can reports the same score for a large number of different alternatives. This might make the reordering metric easier to optimise, leading to the joint best scores at test time. This is an important result, as it shows that by training with the LRscore objective function, BLEU scores do not decrease, which is desirable as BLEU scores are usually reported in the field.

The LRscore also results in better scores when evaluated with itself and the other two baseline metrics, TER and METEOR. Reordering and the lexical metrics are orthogonal information sources, and this shows that combining them results in better performing systems. BLEU has shown to be a strong baseline metric to use as an objective function (Cer et al., 2010), and so the LRscore performance in Table 5 is a good result.

Examining the weights that result from the different MERT runs, the only notable difference is that the weight of the distortion cost is considerably lower with the LRscore. This shows more trust in the quality of reorderings. Although it is interesting to look at the model weights, any final conclusion on the impact of the metrics on training must depend on human evaluation of translation quality.

[2] http://www.crowdflower.com

Type	Sentence
Reference	silicon valley is still a rich area in the united states. the average salary in the area was us $62,400 a year, which was 64% higher than the american average.
LR-KB4	silicon valley is still an affluent area of the united states, the regional labor with an average annual salary of 6.24 million us dollars, higher than the average level of 60 per cent.
BLEU	silicon valley is still in the united states in the region in an affluent area of the workforce, the average annual salary of 6.24 million us dollars, higher than the average level of 60 per cent

Table 7. A reference sentence is compared with output from models trained with BLEU and with the LR-KB4 lrscore.

	Prefer LR	Prefer BLEU	Don't Know
LR-KB4	96	79	5
LR-HB4	93	79	8
Total	189 (52.5%)	158 (43.9%)	13

Table 6. The number of the times human judges preferred the output of systems trained either with the LRscore or with the BLEU score, or were unable to choose.

4.2.2 Human Evaluation

We collect human preference judgements for output from systems trained using the BLEU score and the LRscore in order to determine whether training with the LRscore leads to genuine improvements in translation quality. Table 6 shows the number of the times humans preferred the LRscore or the BLEU score output, or when they did not know. We can see that humans have a greater preference for the output for systems trained with the LRscore, which is preferred 52.5% of the time, compared to the BLEU score, which was only preferred 43.9% of the time.

The sign test can be used to determine whether this difference is significant. Our null hypothesis is that the probability of a human preferring the LRscore trained output is the same as that of preferring the BLEU trained output. The one-tailed alternative hypothesis is that humans prefer the LRscore output. If the null hypothesis is true, then there is only a probability of 0.048 that 189 out of 347 $(189 + 158)$ people will select the LRscore output. We therefore discard the null hypothesis and the human preference for the output of the LRscore trained system is significant to the 95% level.

In order to judge how reliable our judgements are we calculate the inter-annotator agreement. This is given by the Kappa coefficient (K), which balances agreement with expected agreement. The Kappa coefficient is 0.464 which is considered to be a moderate level of agreement.

In analysis of the results, we found that output from the system trained with the LRscore tend to produce sentences with better structure. In Table 7 we see a typical example. The word order of the sentence trained with BLEU is mangled, whereas the LR-KB4 model outputs a clear translation which more closely matches the reference. It also garners higher reordering and BLEU scores.

We expect that more substantial gains can be made in the future by using models which have more powerful reordering capabilities. A richer set of reordering features, and a model capable of longer distance reordering would better leverage metrics which reward good word orderings.

5 Conclusion

We introduced the LRscore which combines a lexical and a reordering metric. The main motivation for this metric is the fact that it measures the reordering quality of MT output by using permutation distance metrics. It is a simple, decomposable metric which interpolates the reordering component with a lexical component, the BLEU score. This paper demonstrates that the LRscore metric is more consistent with human preference judgements of machine translation quality than other machine translation metrics. We also show that when training a phrase-based translation model with the LRscore as the objective function, the model retains its performance as measured by the baseline metrics. Crucially, however, optimisation using the LRscore improves subjective evaluation. Ultimately, the availability of a metric which reliably measures reordering performance should accelerate progress towards developing more powerful reordering models.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for MT evaluation with improved correlation with human judgments. In *Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*.

Nicola Bertoldi, Barry Haddow, and Jean-Baptiste Fouet. 2009. Improved Minimum Error Rate Training in Moses. *The Prague Bulletin of Mathematical Linguistics*, 91:7–16.

Alexandra Birch, Phil Blunsom, and Miles Osborne. 2010. Metrics for MT Evaluation: Evaluating Reordering. *Machine Translation*, 24(1):15–26.

Ondrej Bojar and Zdenek Zabokrtsky. 2009. CzEng0.9: Large Parallel Treebank with Rich Annotation. *Prague Bulletin of Mathematical Linguistics*, 92:63–84.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28, Athens, Greece, March. Association for Computational Linguistics.

Chris Callison-Burch. 2009. Fast, cheap, and creative: evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 286–295, Singapore, August. Association for Computational Linguistics.

Daniel Cer, Christopher D. Manning, and Daniel Jurafsky. 2010. The best lexical metric for phrase-based statistical MT system optimization. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 555–563, Los Angeles, California, June.

Richard Hamming. 1950. Error detecting and error correcting codes. *Bell System Technical Journal*, 26(2):147–160.

Maurice Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30:81–89.

A. Kittur, E. H. Chi, and B. Suh. 2008. Crowdsourcing user studies with Mechanical Turk. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 453–456. ACM.

Philipp Koehn, Franz Och, and Daniel Marcu. 2003. Statistical Phrase-Based translation. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference*, pages 127–133, Edmonton, Canada. Association for Computational Linguistics.

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

Alon Lavie and Abhaya Agarwal. 2008. Meteor, m-BLEU and m-TER: Evaluation metrics for high-correlation with human rankings of machine translation output. In *Proceedings of the Workshop on Statistical Machine Translation at the Meeting of the Association for Computational Linguistics (ACL-2008)*, pages 115–118.

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

Chin-Yew Lin and Franz Och. 2004. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of the Conference on Computational Linguistics*, pages 501–507.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.

Sebastian Padó, Daniel Cer, Michel Galley, Dan Jurafsky, and Christopher D. Manning. 2009. Measuring machine translation quality as semantic equivalence: A metric based on entailment features. *Machine Translation*, pages 181–193.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Association for Computational Linguistics*, pages 311–318, Philadelphia, USA.

Matthew Snover, Bonnie Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263. Association for Computational Linguistics.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of Spoken Language Processing*, pages 901–904.

Billy Wong and Chunyu Kit. 2009. ATEC: automatic evaluation of machine translation via word choice and word order. *Machine Translation*, 23(2-3):141–155.

Reordering with Source Language Collocations

Zhanyi Liu[1,2], Haifeng Wang[2], Hua Wu[2], Ting Liu[1], Sheng Li[1]
[1]Harbin Institute of Technology, Harbin, China
[2]Baidu Inc., Beijing, China
{liuzhanyi, wanghaifeng, wu_hua}@baidu.com
{tliu, lisheng}@hit.edu.cn

Abstract

This paper proposes a novel reordering model for statistical machine translation (SMT) by means of modeling the translation orders of the source language collocations. The model is learned from a word-aligned bilingual corpus where the collocated words in source sentences are automatically detected. During decoding, the model is employed to softly constrain the translation orders of the source language collocations, so as to constrain the translation orders of those source phrases containing these collocated words. The experimental results show that the proposed method significantly improves the translation quality, achieving the absolute improvements of 1.1~1.4 BLEU score over the baseline methods.

1 Introduction

Reordering for SMT is first proposed in IBM models (Brown et al., 1993), usually called *IBM constraint* model, where the movement of words during translation is modeled. Soon after, Wu (1997) proposed an ITG (Inversion Transduction Grammar) model for SMT, called *ITG constraint* model, where the reordering of words or phrases is constrained to two kinds: straight and inverted. In order to further improve the reordering performance, many structure-based methods are proposed, including the reordering model in hierarchical phrase-based SMT systems (Chiang, 2005) and syntax-based SMT systems (Zhang et al., 2007; Marton and Resnik, 2008; Ge, 2010; Visweswariah et al., 2010). Although the sentence structure has been taken into consideration, these methods don't explicitly make use of the strong correlations between words, such as collocations, which can effectively indicate reordering in the target language.

In this paper, we propose a novel method to improve the reordering for SMT by estimating the reordering score of the source-language collocations (source collocations for short in this paper). Given a bilingual corpus, the collocations in the source sentence are first detected automatically using a monolingual word alignment (MWA) method without employing additional resources (Liu et al., 2009), and then the reordering model based on the detected collocations is learned from the word-aligned bilingual corpus. The source collocation based reordering model is integrated into SMT systems as an additional feature to softly constrain the translation orders of the source collocations in the sentence to be translated, so as to constrain the translation orders of those source phrases containing these collocated words.

This method has two advantages: (1) it can automatically detect and leverage collocated words in a sentence, including long-distance collocated words; (2) such a reordering model can be integrated into any SMT systems without resorting to any additional resources.

We implemented the proposed reordering model in a phrase-based SMT system, and the evaluation results show that our method significantly improves translation quality. As compared to the baseline systems, an absolute improvement of 1.1~1.4 BLEU score is achieved.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1036–1044,
Portland, Oregon, June 19-24, 2011.

The paper is organized as follows: In section 2, we describe the motivation to use source collocations for reordering, and briefly introduces the collocation extraction method. In section 3, we present our reordering model. And then we describe the experimental results in section 4 and 5. In section 6, we describe the related work. Lastly, we conclude in section 7.

2 Collocation

A collocation is generally composed of a group of words that occur together more often than by chance. Collocations effectively reveal the strong association among words in a sentence and are widely employed in a variety of NLP tasks (Mckeown and Radey, 2000).

Given two words in a collocation, they can be translated in the same order as in the source language, or in the inverted order. We name the first case as *straight*, and the second *inverted*. Based on the observation that some collocations tend to have fixed translation orders such as "金融 jin-rong 'financial' 危机 wei-ji 'crisis'" (financial crisis) whose English translation order is usually straight, and "法律 fa-lv 'law' 范围 fan-wei 'scope'" (scope of law) whose English translation order is generally inverted, some methods have been proposed to improve the reordering model for SMT based on the collocated words crossing the neighboring components (Xiong et al., 2006). We further notice that some words are translated in different orders when they are collocated with different words. For instance, when "潮流 chao-liu 'trend'" is collocated with "时代 shi-dai 'times'", they are often translated into the "trend of times"; when collocated with "历史 li-shi 'history'", the translation usually becomes the "historical trend". Thus, if we can automatically detect the collocations in the sentence to be translated and their orders in the target language, the reordering information of the collocations could be used to constrain the reordering of phrases during decoding. Therefore, in this paper, we propose to improve the reordering model for SMT by estimating the reordering score based on the translation orders of the source collocations.

In general, the collocations can be automatically identified based on syntactic information such as dependency trees (Lin, 1998). However these methods may suffer from parsing errors. Moreover, for many languages, no valid dependency parser exists. Liu et al. (2009) proposed to automatically detect the collocated words in a sentence with the MWA method. The advantage of this method lies in that it can identify the collocated words in a sentence without additional resources. In this paper, we employ MWA Model l~3 described in Liu et al. (2009) to detect collocations in sentences, which are shown in Eq. (1)~(3).

$$p_{\text{MWA Model 1}}(A \mid S) \propto \prod_{j=1}^{l} t(w_j \mid w_{c_j}) \tag{1}$$

$$p_{\text{MWA Model 2}}(A \mid S) \propto \prod_{j=1}^{l} t(w_j \mid w_{c_j}) \cdot d(j \mid c_j, l) \tag{2}$$

$$p_{\text{MWA Model 3}}(A \mid S) \propto \prod_{i=1}^{l} n(\phi_i \mid w_i) \cdot \prod_{j=1}^{l} t(w_j \mid w_{c_j}) \cdot d(j \mid c_j, l) \tag{3}$$

Where $S = w_1^l$ is a monolingual sentence; ϕ_i denotes the number of words collocating with w_i; $A = \{(i, c_i) \mid i \in [1,l] \,\&\, c_i \neq i\}$ denotes the potentially collocated words in S.

The MWA models measure the collocated words under different constraints. MWA Model 1 only models word collocation probabilities $t(w_j \mid w_{c_j})$. MWA Model 2 additionally employs position collocation probabilities $d(j \mid c_j, l)$. Besides the features in MWA Model 2, MWA Model 3 also considers fertility probabilities $n(\phi_i \mid w_i)$.

Given a sentence, the optimal collocated words can be obtained according to Eq. (4).

$$A^* = \arg\max_{A} p_{\text{MWA Model } i}(A \mid S) \tag{4}$$

Given a monolingual word aligned corpus, the collocation probabilities can be estimated as follows.

$$r(w_i, w_j) = \frac{p(w_i \mid w_j) + p(w_j \mid w_i)}{2} \tag{5}$$

Where, $p(w_i \mid w_j) = \dfrac{count(w_i, w_j)}{\sum_{w'} count(w', w_j)}$; (w_i, w_j) denotes the collocated words in the corpus and $count(w_i, w_j)$ denotes the co-occurrence frequency.

3 Reordering Model with Source Language Collocations

In this section, we first describe how to estimate the orientation probabilities for a given collocation, and then describe the estimation of the reordering score during translation. Finally, we describe the integration of the reordering model into the SMT system.

3.1 Reordering probability estimation

Given a source collocation (f_i, f_j) and its corresponding translations (e_{a_i}, e_{a_j}) in a bilingual sentence pair, the reordering orientation of the collocation can be defined as in Eq. (6).

$$o_{i,j,a_i,a_j} = \begin{cases} \text{straight if } i < j \,\&\, a_i < a_j \text{ or } i > j \,\&\, a_i > a_j \\ \text{inverted if } i > j \,\&\, a_i < a_j \text{ or } i < j \,\&\, a_i > a_j \end{cases} \quad (6)$$

In our method, only those collocated words in source language that are aligned to different target words, are taken into consideration, and those being aligned to the same target word are ignored.

Given a word-aligned bilingual corpus where the collocations in source sentences are detected, the probabilities of the translation orientation of collocations in the source language can be estimated, as follows:

$$p(o = \text{straight} \mid f_i, f_j) = \frac{count(o = \text{straight}, f_i, f_j)}{\sum_{o'} count(o', f_i, f_j)} \quad (7)$$

$$p(o = \text{inverted} \mid f_i, f_j) = \frac{count(o = \text{inverted}, f_i, f_j)}{\sum_{o'} count(o', f_i, f_j)} \quad (8)$$

Here, $count(o, f_i, f_j)$ is collected according to the algorithm in Figure 1.

3.2 Reordering model

Given a sentence $F = f_1^l$ to be translated, the collocations are first detected using the algorithm described in Eq. (4). Then the reordering score is estimated according to the reordering probability weighted by the collocation probability of the collocated words. Formally, for a generated translation candidate T, the reordering score is calculated as follows.

$$P_O(F,T) = \sum_{(i,c_i)} r(f_i, f_{c_i}) \log p(o_{i,c_i,a_i,a_{c_i}} \mid f_i, f_{c_i}) \quad (9)$$

Input: A word-aligned bilingual corpus where the source collocations are detected
Initialization: $count(o, f_i, f_j) = 0$
for each sentence pair <*F*, *E*> in the corpus **do**
  for each collocated word pair (f_i, f_{c_i}) in *F* **do**
    if $i < c_i \,\&\, a_i < a_{c_i}$ or $i > c_i \,\&\, a_i > a_{c_i}$ **then**
      $count(o = straight, f_i, f_{c_i})$++
    if $i < c_i \,\&\, a_i > a_{c_i}$ or $i > c_i \,\&\, a_i < a_{c_i}$ **then**
      $count(o = inverted, f_i, f_{c_i})$++
Output: $count(o, f_i, f_j)$

Figure 1. Algorithm of estimating reordering frequency

Here, $r(f_i, f_{c_i})$ denotes the collocation probability of f_i and f_{c_i} as shown in Eq. (5).

In addition to the detected collocated words in the sentence, we also consider other possible word pairs whose collocation probabilities are higher than a given threshold. Thus, the reordering score is further improved according to Eq. (10).

$$P_O(F,T) = \alpha \cdot \sum_{(i,c_i)} r(f_i, f_{c_i}) \log p(o_{i,c_i,a_i,a_{c_i}} \mid f_i, f_{c_i}) + \beta \cdot \sum_{\substack{(i,j) \notin \{(i,c_i)\} \\ \& \, r(f_i,f_j) > \theta}} r(f_i, f_j) \log p(o_{i,j,a_i,a_j} \mid f_i, f_j) \} \quad (10)$$

Where α and β are two interpolation weights. θ is the threshold of collocation probability. The weights and the threshold can be tuned using a development set.

3.3 Integrated into SMT system

The SMT systems generally employ the log-linear model to integrate various features (Chiang, 2005; Koehn et al., 2007). Given an input sentence *F*, the final translation *E** with the highest score is chosen from candidates, as in Eq. (11).

$$E^* = \arg\max_E \{ \sum_{m=1}^{M} \lambda_m h_m(E,F) \} \quad (11)$$

Where $h_m(E, F)$ (m=1,...,M) denotes features. λ_m is a feature weight.

Our reordering model can be integrated into the system as one feature as shown in (10).

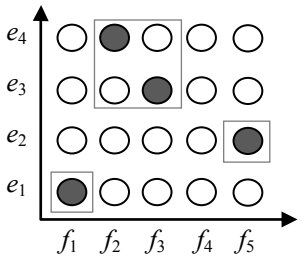


Figure 2. An example for reordering

4 Evaluation of Our Method

4.1 Implementation

We implemented our method in a phrase-based SMT system (Koehn et al., 2007). Based on the GIZA++ package (Och and Ney, 2003), we implemented a MWA tool for collocation detection. Thus, given a sentence to be translated, we first identify the collocations in the sentence, and then estimate the reordering score according to the translation hypothesis. For a translation option to be expanded, the reordering score inside this source phrase is calculated according to their translation orders of the collocations in the corresponding target phrase. The reordering score crossing the current translation option and the covered parts can be calculated according to the relative position of the collocated words. If the source phrase matched by the current translation option is behind the covered parts in the source sentence, then $\log p(o = \text{staight} \mid ...)$ is used, otherwise $\log p(o = \text{inverted} \mid ...)$. For example, in Figure 2, the current translation option is ($f_2\, f_3 \rightarrow e_3\, e_4$). The collocations related to this translation option are (f_1, f_3) , (f_2, f_3) , (f_3, f_5) . The reordering scores can be estimated as follows:

$$r(f_1, f_3) \log p(o = \text{straight} \mid f_1, f_3)$$
$$r(f_2, f_3) \log p(o = \text{inverted} \mid f_2, f_3)$$
$$r(f_3, f_5) \log p(o = \text{inverted} \mid f_3, f_5)$$

In order to improve the performance of the decoder, we design a heuristic function to estimate the future score, as shown in Figure 3. For any uncovered word and its collocates in the input sentence, if the collocate is uncovered, then the higher reordering probability is used. If the collocate has been covered, then the reordering orientation can be determined according to the relative positions of the words and the corresponding reordering probability is employed.

Input: Input sentence $F = f_1^L$
Initialization: *Score* = 0
for each uncovered word f_i **do**
  for each word f_j ($j = c_i$ or $r(f_{i,} f_j) > \theta$) **do**
    if f_j is covered **then**
      if $i > j$ **then**
        $Score \mathrel{+}= r(f_{i,} f_j) \log p(o = \text{straight} \mid f_i, f_j)$
      else
        $Score \mathrel{+}= r(f_{i,} f_j) \log p(o = \text{inverted} \mid f_i, f_j)$
    else
      $Score \mathrel{+}= \arg\max_o r(f_{i,} f_j) \log p(o \mid f_i, f_j)$
Output: *Score*

Figure 3. Heuristic function for estimating future score

4.2 Settings

We use the FBIS corpus (LDC2003E14) to train a Chinese-to-English phrase-based translation model. And the SRI language modeling toolkit (Stolcke, 2002) is used to train a 5-gram language model on the English sentences of FBIS corpus.

We used the NIST evaluation set of 2002 as the development set to tune the feature weights of the SMT system and the interpolation parameters, based on the minimum error rate training method (Och, 2003), and the NIST evaluation sets of 2004 and 2008 (MT04 and MT08) as the test sets.

We use BLEU (Papineni et al., 2002) as evaluation metrics. We also calculate the statistical significance differences between our methods and the baseline method by using the paired bootstrap resample method (Koehn, 2004).

4.3 Translation results

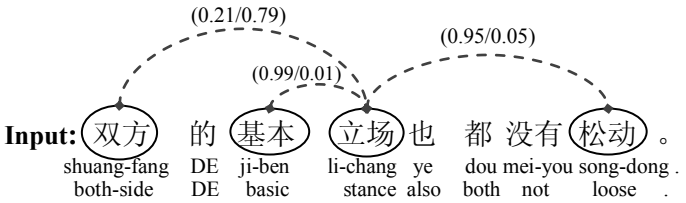
We compare the proposed method with various reordering methods in previous work.

Monotone model: no reordering model is used.

Distortion based reordering (DBR) model: a distortion based reordering method (Al-Onaizan & Papineni, 2006). In this method, the distortion cost is defined in terms of words, rather than phrases. This method considers *outbound*, *inbound*, and *pairwise* distortions that

Reorder models		MT04	MT08
Monotone model		26.99	18.30
DBR model		26.64	17.83
MSDR model (Baseline)		28.77	18.42
MSDR+	DBR model	28.91	18.58
	SCBR Model 1	29.21	19.28
	SCBR Model 2	29.44	19.36
	SCBR Model 3	29.50	19.44
	SCBR models (1+2)	29.65	19.57
	SCBR models (1+2+3)	29.75	19.61

Table 1. Translation results on various reordering models



T1: The two sides are also the basic stand of not relaxed.
T2: The basic stance of the two sides have not relaxed.
Reference: The basic stances of both sides did not move.

Figure 4. Translation example. (*/*) denotes ($p_{straight}$ / $p_{inverted}$)

are directly estimated by simple counting over alignments in the word-aligned bilingual corpus. This method is similar to our proposed method. But our method considers the translation order of the collocated words.

msd-bidirectional-fe reordering (MSDR or Baseline) model: it is one of the reordering models in Moses. It considers three different orientation types (*monotone*, *swap*, and *discontinuous*) on both source phrases and target phrases. And the translation orders of both the next phrase and the previous phrase in respect to the current phrase are modeled.

Source collocation based reordering (SCBR) model: our proposed method. We investigate three reordering models based on the corresponding MWA models and their combinations. In SCBR Model *i* (*i*=1~3), we use MWA Model *i* as described in section 2 to obtain the collocated words and estimate the reordering probabilities according to section 3.

The experiential results are shown in Table 1. The DBR model suffers from serious data sparseness. For example, the reordering cases in the trained pairwise distortion model only covered 32~38% of those in the test sets. So its performance is worse than that of the monotone model. The MSDR model achieves higher BLEU scores than the monotone model and the DBR model. Our models further improve the translation quality, achieving better performance than the combination of MSDR model and DBR model. The results in Table 1 show that "MSDR + SCBR Model 3" performs the best among the SCBR models. This is because, as compared to MWA Model 1 and 2, MWA Model 3 takes more information into consideration, including not only the co-occurrence information of lexical tokens and the position of words, but also the fertility of words in a sentence. And when the three SCBR models are combined, the performance of the SMT system is further improved. As compared to other reordering models, our models achieve an absolute improvement of 0.98~1.19 BLEU score on the test sets, which are statistically significant ($p < 0.05$).

Figure 4 shows an example: T1 is generated by the baseline system and T2 is generated by the system where the SCBR models (1+2+3)[1] are used.

[1] In the remainder of this paper, "SCBR models" means the combination of the SCBR models (1+2+3) unless it is explicitly explained.

Reordering models		MT04	MT08
MSDR model		28.77	18.42
MSDR+	DBR model	28.91	18.58
	CBR model	28.96	18.77
	WCBR model	29.15	19.10
	WCBR+SCBR models	29.87	19.83

Table 2. Translation results of co-occurrence based reordering models

	CBR model	SCBR Model3
Consecutive words	77.9%	73.5%
Interrupted words	74.1%	87.8%
Total	74.3%	84.9%

Table 3. Precisions of the reordering models on the development set

The input sentence contains three collocations. The collocation (基本, 立场) is included in the same phrase and translated together as a whole. Thus its translation is correct in both translations. For the other two long-distance collocations (双方, 立场) and (立场, 松动), their translation orders are not correctly handled by the reordering model in the baseline system. For the collocation (双方, 立场), since the SCBR models indicate $p(o$=straight|双方, 立场$) < p(o$=inverted|双方, 立场$)$, the system finally generates the translation T2 by constraining their translation order with the proposed model.

5 Collocations vs. Co-occurring Words

We compared our method with the method that models the reordering orientations based on co-occurring words in the source sentences, rather than the collocations.

5.1 Co-occurrence based reordering model

We use the similar algorithm described in section 3 to train the co-occurrence based reordering (CBR) model, except that the probability of the reordering orientation is estimated on the co-occurring words and the relative distance. Given an input sentence and a translation candidate, the reordering score is estimated as shown in Eq. (12).

$$P_O(F,T) = \sum_{(i,j)} \log p(o_{i,j,a_i,a_j} \mid f_i, f_j, \Delta_{i-j}) \quad (12)$$

Here, Δ_{i-j} is the relative distance of two words in the source sentence.

We also construct the weighted co-occurrence based reordering (WCBR) model. In this model, the probability of the reordering orientation is additionally weighted by the pointwise mutual information[2] score of the two words (Manning and Schütze, 1999), which is estimated as shown in Eq. (13).

$$P_O(F,T) = \sum_{(i,j)} s_{\mathrm{MI}}(f_i, f_j) \log p(o_{i,j,a_i,a_j} \mid f_i, f_j, \Delta_{i-j}) \quad (13)$$

5.2 Translation results

Table 2 shows the translation results. It can be seen that the performance of the SMT system is improved by integrating the CBR model. The performance of the CBR model is also better than that of the DBR model. It is because the former is trained based on all co-occurring aligned words, while the latter only considers the adjacent aligned words. When the WCBR model is used, the translation quality is further improved. However, its performance is still inferior to that of the SCBR models, indicating that our method (SCBR models) of modeling the translation orders of source collocations is more effective. Furthermore, we combine the weighted co-occurrence based model and our method, which outperform all the other models.

5.3 Result analysis

Precision of prediction

First of all, we investigate the performance of the reordering models by calculating precisions of the translation orders predicted by the reordering models. Based on the source sentences and reference translations of the development set, where the source words and target words are automatically aligned by the bilingual word alignment method, we construct the reference translation orders for two words. Against the references, we calculate three kinds of precisions as follows:

$$P_{\mathrm{CW}} = \frac{|\{|i-j|=1 \;\&\; o_{i,j} = o_{i,j,a_i,a_j}\}|}{|\{o_{i,j} \parallel i-j|=1\}|} \quad (14)$$

[2] For occurring words extraction, the window size is set to [-6, +6].

$$P_{\mathrm{IW}} = \frac{|\{|i-j|>1 \;\&\; o_{i,j} = o_{i,j,a_i,a_j}\}|}{|\{o_{i,j} \parallel i-j|>1\}|} \quad (15)$$

$$P_{\mathrm{total}} = \frac{|\{o_{i,j} = o_{i,j,a_i,a_j}\}|}{|\{o_{i,j}\}|} \quad (16)$$

Here, $o_{i,j}$ denotes the translation order of (f_i, f_j) predicted by the reordering models. If $p(o=\text{straight}\,|\,f_i,f_j) > p(o=\text{inverted}\,|f_i,f_j)$, then $o_{i,j}=\text{straight}$, else if $p(o=\text{straight}\,|\,f_i,f_j) < p(o=\text{inverted}\,|f_i,f_j)$, then $o_{i,j}=\text{inverted}$. o_{i,j,a_i,a_j} denotes the translation order derived from the word alignments. If $o_{i,j}=o_{i,j,a_i,a_j}$, then the predicted translation order is correct, otherwise wrong. P_{CW} and P_{IW} denote the precisions calculated on the consecutive words and the interrupted words in the source sentences, respectively. P_{total} denotes the precision on both cases. Here, the CBR model and SCBR Model 3 are compared. The results are shown in Table 3.

From the results in Table 3, it can be seen that the CBR model has a higher precision on the consecutive words than the SCBR model, but lower precisions on the interrupted words. It is mainly because the CBR model introduces more noise when the relative distance of words is set to a large number, while the MWA method can effectively detect the long-distance collocations in sentences (Liu et al., 2009). This explains why the combination of the two models can obtain the highest BLEU score as shown in Table 2. On the whole, the SCBR Model 3 achieves higher precision than the CBR model.

Effect of the reordering model

Then we evaluate the reordering results of the generated translations in the test sets. Using the above method, we construct the reference translation orders of collocations in the test sets. For a given word pair in a source sentence, if the translation order in the generated translation is the same as that in the reference translations, then it is correct, otherwise wrong.

We compare the translations of the baseline method, the co-occurrence based method, and our method (SCBR models). The precisions calculated on both kinds of words are shown in Table 4. From the results, it can be seen that our method achieves higher precisions than both the baseline and the method modeling the translation orders of the co-occurring words. It indicates that the proposed method effectively constrains the reordering of source words during decoding and improves the translation quality.

Test sets	Baseline (MSDR)	MSDR+ WCBR	MSDR+ SCBR
MT04	78.9%	80.8%	82.5%
MT08	80.7%	83.8%	85.0%

Table 4. Precisions (total) of the reordering models on the test sets

6 Related Work

Reordering was first proposed in the IBM models (Brown et al., 1993), later was named *IBM constraint* by Berger et al. (1996). This model treats the source word sequence as a coverage set that is processed sequentially and a source token is covered when it is translated into a new target token. In 1997, another model called *ITG constraint* was presented, in which the reordering order can be hierarchically modeled as *straight* or *inverted* for two nodes in a binary branching structure (Wu, 1997). Although the *ITG constraint* allows more flexible reordering during decoding, Zens and Ney (2003) showed that the *IBM constraint* results in higher BLEU scores. Our method models the reordering of collocated words in sentences instead of all words in IBM models or two neighboring blocks in ITG models.

For phrase-based SMT models, Koehn et al. (2003) linearly modeled the distance of phrase movements, which results in poor global reordering. More methods are proposed to explicitly model the movements of phrases (Tillmann, 2004; Koehn et al., 2005) or to directly predict the orientations of phrases (Tillmann and Zhang, 2005; Zens and Ney, 2006), conditioned on current source phrase or target phrase. Hierarchical phrase-based SMT methods employ SCFG bilingual translation model and allow flexible reordering (Chiang, 2005). However, these methods ignored the correlations among words in the source language or in the target language. In our method, we automatically detect the collocated words in sentences and

their translation orders in the target languages, which are used to constrain the ordering models with the estimated reordering (straight or inverted) score. Moreover, our method allows flexible reordering by considering both consecutive words and interrupted words.

In order to further improve translation results, many researchers employed syntax-based reordering methods (Zhang et al., 2007; Marton and Resnik, 2008; Ge, 2010; Visweswariah et al., 2010). However these methods are subject to parsing errors to a large extent. Our method directly obtains collocation information without resorting to any linguistic knowledge or tools, therefore is suitable for any language pairs.

In addition, a few models employed the collocation information to improve the performance of the ITG constraints (Xiong et al., 2006). Xiong et al. used the consecutive co-occurring words as collocation information to constrain the reordering, which did not lead to higher translation quality in their experiments. In our method, we first detect both consecutive and interrupted collocated words in the source sentence, and then estimated the reordering score of these collocated words, which are used to softly constrain the reordering of source phrases.

7 Conclusions

We presented a novel model to improve SMT by means of modeling the translation orders of source collocations. The model was learned from a word-aligned bilingual corpus where the potentially collocated words in source sentences were automatically detected by the MWA method. During decoding, the model is employed to softly constrain the translation orders of the source language collocations. Since we only model the reordering of collocated words, our methods can partially alleviate the data sparseness encountered by other methods directly modeling the reordering based on source phrases or target phrases. In addition, this kind of reordering information can be integrated into any SMT systems without resorting to any additional resources.

The experimental results show that the proposed method significantly improves the translation quality of a phrase based SMT system, achieving an absolute improvement of 1.1~1.4 BLEU score over the baseline methods.

References

Yaser Al-Onaizan and Kishore Papineni. 2006. Distortion Models for Statistical Machine Translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pp. 529-536.

Adam L. Berger, Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Andrew S. Kehler, and Robert L. Mercer. 1996. Language Translation Apparatus and Method of Using Context-Based Translation Models. *United States Patent, Patent Number 5510981*, April.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert. L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter estimation. *Computational Linguistics*, 19(2): 263-311.

David Chiang. 2005. A Hierarchical Phrase-based Model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pp. 263-270.

Niyu Ge. 2010. A Direct Syntax-Driven Reordering Model for Phrase-Based Machine Translation. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL*, pp. 849-857.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pp. 388-395.

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics*, pp. 127-133.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL, Poster and Demonstration Sessions*, pp. 177-180.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In *Proceedings of International Workshop on Spoken Language Translation*.

Dekang Lin. 1998. Extracting Collocations from Text Corpora. In *Proceedings of the 1st Workshop on Computational Terminology*, pp. 57-63.

Zhanyi Liu, Haifeng Wang, Hua Wu, and Sheng Li. 2009. Collocation Extraction Using Monolingual Word Alignment Method. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pp. 487-495.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*, Cambridge, MA; London, U.K.: Bradford Book & MIT Press.

Yuval Marton and Philip Resnik. 2008. Soft Syntactic Constraints for Hierarchical Phrased-based Translation. In *Proceedings of the 46st Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 1003-1011.

Kathleen R. McKeown and Dragomir R. Radev. 2000. Collocations. In Robert Dale, Hermann Moisl, and Harold Somers (Ed.), *A Handbook of Natural Language Processing*, pp. 507-523.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 160-167.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1) : 19-51.

Kishore Papineni, Salim Roukos, Todd Ward, and Weijing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311-318.

Andreas Stolcke. 2002. SRILM - An Extensible Language Modeling Toolkit. In *Proceedings for the International Conference on Spoken Language Processing*, pp. 901-904.

Christoph Tillmann. 2004. A Unigram Orientation Model for Statistical Machine Translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics*, pp. 101-104.

Christoph Tillmann and Tong Zhang. 2005. A Localized Prediction Model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pp. 557-564.

Karthik Visweswariah, Jiri Navratil, Jeffrey Sorensen, Vijil Chenthamarakshan, and Nanda Kambhatla. 2010. Syntax Based Reordering with Automatically Derived Rules for Improved Statistical Machine Translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 1119-1127.

Dekai Wu. 1997. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3):377-403.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 521-528.

Richard Zens and Herman Ney. 2003. A Comparative Study on Reordering Constraints in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 192-202.

Richard Zens and Herman Ney. 2006. Discriminative Reordering Models for Statistical Machine Translation. In *Proceedings of the Workshop on Statistical Machine Translation*, pp. 55-63.

Dongdong Zhang, Mu Li, Chi-Ho Li, and Ming Zhou. 2007. Phrase Reordering Model Integrating Syntactic Knowledge for SMT. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 533-540.

A Joint Sequence Translation Model with Integrated Reordering

Nadir Durrani **Helmut Schmid** **Alexander Fraser**
Institute for Natural Language Processing
University of Stuttgart
{durrani,schmid,fraser}@ims.uni-stuttgart.de

Abstract

We present a novel machine translation model which models translation by a linear sequence of *operations*. In contrast to the "N-gram" model, this sequence includes not only translation but also reordering operations. Key ideas of our model are (i) a new reordering approach which better restricts the position to which a word or phrase can be moved, and is able to handle short and long distance reorderings in a unified way, and (ii) a joint sequence model for the translation and reordering probabilities which is more flexible than standard phrase-based MT. We observe statistically significant improvements in BLEU over Moses for German-to-English and Spanish-to-English tasks, and comparable results for a French-to-English task.

1 Introduction

We present a novel generative model that explains the translation process as a linear sequence of operations which generate a source and target sentence in parallel. Possible operations are (i) generation of a sequence of source and target words (ii) insertion of *gaps* as explicit target positions for reordering operations, and (iii) forward and backward jump operations which do the actual reordering. The probability of a sequence of operations is defined according to an N-gram model, i.e., the probability of an operation depends on the $n-1$ preceding operations. Since the translation (generation) and reordering operations are coupled in a single generative story, the reordering decisions may depend on preceding translation decisions and translation decisions may depend on preceding reordering decisions. This provides a natural reordering mechanism which is able to deal with local and long-distance reorderings in a consistent way. Our approach can be viewed as an extension of the N-gram SMT approach (Mariño et al., 2006) but our model does reordering as an integral part of a generative model.

The paper is organized as follows. Section 2 discusses the relation of our work to phrase-based and the N-gram SMT. Section 3 describes our generative story. Section 4 defines the probability model, which is first presented as a generative model, and then shifted to a discriminative framework. Section 5 provides details on the search strategy. Section 6 explains the training process. Section 7 describes the experimental setup and results. Section 8 gives a few examples illustrating different aspects of our model and Section 9 concludes the paper.

2 Motivation and Previous Work

2.1 Relation of our work to PBSMT

Phrase-based SMT provides a powerful translation mechanism which learns local reorderings, translation of short idioms, and the insertion and deletion of words sensitive to local context. However, PBSMT also has some drawbacks. (i) Dependencies across phrases are not directly represented in the translation model. (ii) Discontinuous phrases cannot be used. (iii) The presence of many different equivalent segmentations increases the search space.

Phrase-based SMT models dependencies between words and their translations inside of a phrase well. However, dependencies across phrase boundaries are largely ignored due to the strong phrasal inde-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1045–1054,
Portland, Oregon, June 19-24, 2011.

German	English
hat er ein buch gelesen	he read a book
hat eine pizza gegessen	has eaten a pizza
er	he
hat	has
ein	a
eine	a
menge	lot of
butterkekse	butter cookies
gegessen	eaten
buch	book
zeitung	newspaper
dann	then

Table 1: Sample Phrase Table

pendence assumption. A phrase-based system using the phrase table[1] shown in Table 1, for example, correctly translates the German sentence "er hat eine pizza gegessen" to "he has eaten a pizza", but fails while translating "er hat eine menge butterkekse gegessen" (see Table 1 for a gloss) which is translated as "he has a lot of butter cookies eaten" unless the language model provides strong enough evidence for a different ordering. The generation of this sentence in our model starts with generating "er – he", "hat – has". Then a gap is inserted on the German side, followed by the generation of "gegessen – eaten". At this point, the (partial) German and English sentences look as follows:

er hat ☐ gegessen
he has eaten

We jump back to the gap on the German side and fill it by generating "eine – a" and "pizza – pizza", for the first example and generating "eine – a", "menge – lot of", "butterkekse – butter cookies" for the second example, thus handling both short and long distance reordering in a unified manner. Learning the pattern "hat ☐ gegessen – has eaten" helps us to generalize to the second example with unseen context. Notice how the reordering decision is triggered by the translation decision in our model. The probability of a gap insertion operation after the generation of the auxiliaries "hat – has" will be high because reordering is necessary in order to move the second part of the German verb complex ("gegessen") to its correct position at the end of the clause. This mechanism better restricts reordering

[1]The examples given in this section are not taken from the real data/system, but made-up for the sake of argument.

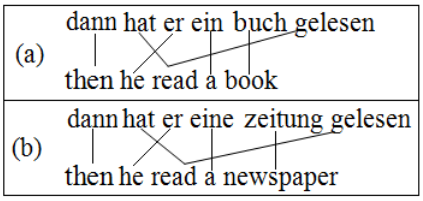


Figure 1: (a) Known Context (b) Unknown Context

than traditional PBSMT and is able to deal with local and long-distance reorderings in a consistent way.

Another weakness of the traditional phrase-based system is that it can only capitalize on continuous phrases. Given the phrase inventory in Table 1, phrasal MT is able to generate example in Figure 1(a). The information "hat...gelesen – read" is internal to the phrase pair "hat er ein buch gelesen – he read a book", and is therefore handled conveniently. On the other hand, the phrase table does not have the entry "hat er eine zeitung gelesen – he read a newspaper" (Figure 1(b)). Hence, there is no option but to translate "hat...gelesen" separately, translating "hat" to "has" which is a common translation for "hat" but wrong in the given context. Context-free hierarchical models (Chiang, 2007; Melamed, 2004) have rules like "hat er X gelesen – he read X" to handle such cases. Galley and Manning (2010) recently solved this problem for phrasal MT by extracting phrase pairs with source and target-side gaps. Our model can also use tuples with source-side discontinuities. The above sentence would be generated by the following sequence of operations: (i) generate "dann – then" (ii) insert a gap (iii) generate "er – he" (iv) backward jump to the gap (v) generate "hat...[gelesen] – read" (only "hat" and "read" are added to the sentences yet) (vi) jump forward to the right-most source word so far generated (vii) insert a gap (viii) continue the source cept ("gelesen" is inserted now) (ix) backward jump to the gap (x) generate "ein – a" (xi) generate "buch – book".

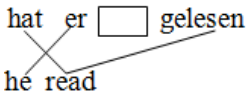


Figure 2: Pattern

From this operation sequence, the model learns a pattern (Figure 2) which allows it to generalize to the example in Figure 1(b). The open gap represented by ☐ serves a similar purpose as the non-terminal categories in a hierarchical phrase-based system such as Hiero. Thus it generalizes to translate "eine zeitung" in exactly the same way as "ein buch".

Another problem of phrasal MT is spurious phrasal segmentation. Given a sentence pair and a corresponding word alignment, phrasal MT can learn an arbitrary number of source segmentations. This is problematic during decoding because different compositions of the same minimal phrasal units are allowed to compete with each other.

2.2 Relation of our work to N-gram SMT

N-gram based SMT is an alternative to hierarchical and non-hierarchical phrase-based systems. The main difference between phrase-based and N-gram SMT is the extraction procedure of translation units and the statistical modeling of translation context (Crego et al., 2005a). The tuples used in N-gram systems are much smaller translation units than phrases and are extracted in such a way that a unique segmentation of each bilingual sentence pair is produced. This helps N-gram systems to avoid the spurious phrasal segmentation problem. Reordering works by linearization of the source side and tuple unfolding (Crego et al., 2005b). The decoder uses word lattices which are built with linguistically motivated re-write rules. This mechanism is further enhanced with an N-gram model of bilingual units built using POS tags (Crego and Yvon, 2010). A drawback of their reordering approach is that search is only performed on a small number of reorderings that are pre-calculated on the source side independently of the target side. Often, the evidence for the correct ordering is provided by the target-side language model (LM). In the N-gram approach, the LM only plays a role in selecting between the pre-calculated orderings.

Our model is based on the N-gram SMT model, but differs from previous N-gram systems in some important aspects. It uses operation n-grams rather than tuple n-grams. The reordering approach is entirely different and considers all possible orderings instead of a small set of pre-calculated orderings. The standard N-gram model heavily relies on POS tags for reordering and is unable to use lexical triggers whereas our model exclusively uses lexical triggers and no POS information. Linearization and unfolding of the source sentence according to the target sentence enables N-gram systems to handle source-side gaps. We deal with this phenomenon more directly by means of tuples with source-side discontinuities. The most notable feature of our work is that it has a complete generative story of translation which combines translation and reordering operations into a single operation sequence model.

Like the N-gram model[2], our model cannot deal with target-side discontinuities. These are eliminated from the training data by a post-editing process on the alignments (see Section 6). Galley and Manning (2010) found that target-side gaps were not useful in their system and not useful in the hierarchical phrase-based system Joshua (Li et al., 2009).

3 Generative Story

Our generative story is motivated by the complex reorderings in the German-to-English translation task. The German and English sentences are jointly generated through a sequence of operations. The English words are generated in linear order[3] while the German words are generated in parallel with their English translations. Occasionally the translator jumps back on the German side to insert some material at an earlier position. After this is done, it jumps forward again and continues the translation. The backward jumps always end at designated landing sites (gaps) which were explicitly inserted before. We use 4 translation and 3 reordering operations. Each is briefly discussed below.

Generate (X,Y): X and Y are German and English cepts[4] respectively, each with one or more words. Words in X (German) may be consecutive or discontinuous, but the words in Y (English) must be consecutive. This operation causes the words in Y and the first word in X to be added to the English and German strings respectively, that were generated so far. Subsequent words in X are added to a queue to be generated later. All the English words in Y are generated immediately because English is generated in linear order. The generation of the second (and subsequent) German word in a multi-word cept can be delayed by gaps, jumps and the Generate Source Only operation defined below.

Continue Source Cept: The German words added

[2]However, Crego and Yvon (2009), in their N-gram system, use split rules to handle target-side gaps and show a slight improvement on a Chinese-English translation task.

[3]Generating the English words in order is also what the decoder does when translating from German to English.

[4]A cept is a group of words in one language translated as a minimal unit in one specific context (Brown et al., 1993).

to the queue by the Generate (X,Y) operation are generated by the Continue Source Cept operation. Each Continue Source Cept operation removes one German word from the queue and copies it to the German string. If X contains more than one German word, say n many, then it requires n translation operations, an initial Generate $(X_1...X_n, Y)$ operation and $n-1$ Continue Source Cept operations. For example "hat...gelesen – read" is generated by the operation Generate (hat gelesen, read), which adds "hat" and "read" to the German and English strings and "gelesen" to a queue. A Continue Source Cept operation later removes "gelesen" from the queue and adds it to the German string.

Generate Source Only (X): The string X is added at the current position in the German string. This operation is used to generate a German word X with no corresponding English word. It is performed immediately after its preceding German word is covered. This is because there is no evidence on the English-side which indicates when to generate X. Generate Source Only (X) helps us learn a source word deletion model. It is used during decoding, where a German word (X) is either translated to some English word(s) by a Generate (X,Y) operation or deleted with a Generate Source Only (X) operation.

Generate Identical: The same word is added at the current position in both the German and English strings. The Generate Identical operation is used during decoding for the translation of unknown words. The probability of this operation is estimated from singleton German words that are translated to an identical string. For example, for a tuple "Portland – Portland", where German "Portland" was observed exactly once during training, we use a Generate Identical operation rather than Generate (Portland, Portland).

We now discuss the set of reordering operations used by the generative story. Reordering has to be performed whenever the German word to be generated next does not immediately follow the previously generated German word. During the generation process, the translator maintains an index which specifies the position after the previously covered German word (j), an index (Z) which specifies the index after the right-most German word covered so far, and an index of the next German word to be covered (j'). The set of reordering operations used in generation depends upon these indexes.

Operations	Generation	States
Generate (dann , then)	dann ↓ then	i=1 j=1 j'=2 k=0 Z=1
Insert Gap – Generate (er , he)	dann □ er ↓ then he	i=2 j=3 j'=1 k=0 Z=3 S=1
Jump Back(1) – Generate (hat gelesen, read)	dann hat ↓ er then he read	i=3 j=2 j'=5 k=1 Z=3
Jump Forward – Insert Gap – Continue Source Cept	dann hat er □ gelesen ↓ then he read	i=3 j=6 j'=3 k=0 Z=6 S=3
Jump Back(1) – Generate (ein , a)	dann hat er ein ↓ gelesen then he read a	i=4 j =4 j'=4 k=0 Z=6
Generate (buch , book)	dann hat er ein buch ↓ gelesen then he read a book	i=5 j=5 j'=6 k=0 Z=6

Table 2: Step-wise Generation of Example 1(a). The arrow indicates position j.

Insert Gap: This operation inserts a gap which acts as a place-holder for the skipped words. There can be more than one open gap at a time.

Jump Back (W): This operation lets the translator jump back to an open gap. It takes a parameter W specifying which gap to jump to. Jump Back (1) jumps to the closest gap to Z, Jump Back (2) jumps to the second closest gap to Z, etc. After the backward jump the target gap is closed.

Jump Forward: This operation makes the translator jump to Z. It is performed if some already generated German word is between the previously generated word and the word to be generated next. A Jump Back (W) operation is only allowed at position Z. Therefore, if $j \neq Z$, a Jump Forward operation has to be performed prior to a Jump Back operation.

Table 2 shows step by step the generation of a German/English sentence pair, the corresponding translation operations, and the respective values of the index variables. A formal algorithm for converting a word-aligned bilingual corpus into an operation sequence is presented in Algorithm 1.

4 Model

Our translation model $p(F, E)$ is based on operation N-gram model which integrates translation and reordering operations. Given a source string F, a sequence of tuples $T = (t_1, \ldots, t_n)$ as hypothesized by the decoder to generate a target string E, the translation model estimates the probability of a

Algorithm 1 Corpus Conversion Algorithm

i	Position of current English cept
j	Position of current German word
j'	Position of next German word
N	Total number of English cepts
f_j	German word at position j
E_i	English cept at position i
F_i	Sequence of German words linked to E_i
L_i	Number of German words linked with E_i
k	Number of already generated German words for E_i
a_{ik}	Position of k^{th} German translation of E_i
Z	Position after right-most generated German word
S	Position of the first word of a target gap

$i := 0; j := 0; k := 0$

while f_j is an unaligned word **do**
  Generate Source Only (f_j)
  $j := j + 1$
$Z := j$

while $i < N$ **do**
  $j' := a_{ik}$
  if $j < j'$ **then**
    if f_j was not generated yet **then**
      Insert Gap
    if $j = Z$ **then**
      $j := j'$
    else
      Jump Forward
  if $j' < j$ **then**
    if $j < Z$ and f_j was not generated yet **then**
      Insert Gap
    $W :=$ relative position of target gap
    Jump Back (W)
    $j := S$
  if $j < j'$ **then**
    Insert Gap
    $j := j'$
  if k = 0 **then**
    Generate (F_i, E_i) {or **Generate Identical**}
  else
    Continue Source Cept
  $j := j + 1; k := k + 1$
  while f_j is an unaligned word **do**
    Generate Source Only (f_j)
    $j := j + 1$
  if $Z < j$ **then**
    $Z := j$
  if $k = L_i$ **then**
    $i := i + 1; k := 0$

Remarks:
We use cept positions for English (not word positions) because English cepts are composed of consecutive words. German positions are word-based.
The relative position of the target gap is 1 if it is closest to Z, 2 if it is the second closest gap etc.
The operation **Generate Identical** is chosen if $F_i = E_i$ and the overall frequency of the German cept F_i is 1.

generated operation sequence $O = (o_1, \ldots, o_J)$ as:

$$p(F, E) \approx \prod_{j=1}^{J} p(o_j | o_{j-m+1} ... o_{j-1})$$

where m indicates the amount of context used. Our translation model is implemented as an N-gram model of operations using SRILM-Toolkit (Stolcke, 2002) with Kneser-Ney smoothing. We use a 9-gram model ($m = 8$).

Integrating the language model the search is defined as:

$$\hat{E} = \arg\max_{E} p_{LM}(E) p(F, E)$$

where $p_{LM}(E)$ is the monolingual language model and $p(F, E)$ is the translation model. But our translation model is a joint probability model, because of which E is generated twice in the numerator. We add a factor, prior probability $p_{pr}(E)$, in the denominator, to negate this effect. It is used to marginalize the joint-probability model $p(F, E)$. The search is then redefined as:

$$\hat{E} = \arg\max_{E} p_{LM}(E) \frac{p(F, E)}{p_{pr}(E)}$$

Both, the monolingual language and the prior probability model are implemented as standard word-based n-gram models:

$$p_x(E) \approx \prod_{j=1}^{J} p(w_j | w_{j-m+1}, \ldots, w_{j-1})$$

where $m = 4$ (5-gram model) for the standard monolingual model ($x = LM$) and $m = 8$ (same as the operation model[5]) for the prior probability model ($x = pr$).

In order to improve end-to-end accuracy, we introduce new features for our model and shift from the generative[6] model to the standard log-linear approach (Och and Ney, 2004) to tune[7] them. We search for a target string E which maximizes a linear combination of feature functions:

[5] In decoding, the amount of context used for the prior probability is synchronized with the position of back-off in the operation model.

[6] Our generative model is about 3 BLEU points worse than the best discriminative results.

[7] We tune the operation, monolingual and prior probability models as separate features. We expect the prior probability model to get a negative weight but we do not force MERT to assign a negative weight to this feature.

$$\hat{E} = \arg\max_{E} \left\{ \sum_{j=1}^{J} \lambda_j h_j(F, E) \right\}$$

where λ_j is the weight associated with the feature $h_j(F, E)$. Other than the 3 features discussed above (log probabilities of the operation model, monolingual language model and prior probability model), we train 8 additional features discussed below:

Length Bonus The length bonus feature counts the length of the target sentence in words.

Deletion Penalty Another feature for avoiding too short translations is the deletion penalty. Deleting a source word (Generate Source Only (X)) is a common operation in the generative story. Because there is no corresponding target-side word, the monolingual language model score tends to favor this operation. The deletion penalty counts the number of deleted source words.

Gap Bonus and Open Gap Penalty These features are introduced to guide the reordering decisions. We observe a large amount of reordering in the automatically word aligned training text. However, given only the source sentence (and little world knowledge), it is not realistic to try to model the reasons for all of this reordering. Therefore we can use a more robust model that reorders less than humans. The gap bonus feature sums to the total number of gaps inserted to produce a target sentence. The open gap penalty feature is a penalty (paid once for each translation operation performed) whose value is the number of open gaps. This penalty controls how quickly gaps are closed.

Distortion and Gap Distance Penalty We have two additional features to control the reordering decisions. One of them is similar[8] to the distance-based reordering model used by phrasal MT. The other feature is the gap distance penalty which calculates the distance between the first word of a source cept X and the start of the left-most gap. This cost is paid once for each Generate, Generate Identical and Generate Source Only. For a source cept coverd by indexes $X_1, \ldots, X_n$, we get the feature value $g_j = X_1 - S$, where S is the index of the left-most source word where a gap starts.

[8]Let $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$ represent indexes of the source words covered by the tuples t_j and t_{j-1} respectively. The distance between t_j and t_{j-1} is given as $d_j = min(|X_k - Y_l| - 1) \; \forall \; X_k \in \{X_1, \ldots, X_n\}$ and $\forall \; Y_l \in \{Y_1, \ldots, Y_m\}$

Lexical Features We also use source-to-target $p(e|f)$ and target-to-source $p(f|e)$ lexical translation probabilities. Our lexical features are standard (Koehn et al., 2003). The estimation is motivated by IBM Model-1. Given a tuple t_i with source words $f = f_1, f_2, \ldots, f_n$, target words $e = e_1, e_2, \ldots, e_m$ and an alignment a between the source word positions $x = 1, \ldots, n$ and the target word positions $y = 1, \ldots, m$, the lexical feature $p_w(f|e)$ is computed as follows:

$$p_w(f|e, a) = \prod_{x=1}^{n} \frac{1}{|\{y : (x, y) \in a\}|} \sum_{\forall (x,y) \in a} w(f_x|e_y)$$

$p_w(e|f, a)$ is computed in the same way.

5 Decoding

Our decoder for the new model performs a stack-based search with a beam-search algorithm similar to that used in Pharoah (Koehn, 2004a). Given an input sentence F, it first extracts a set of matching source-side cepts along with their n-best translations to form a tuple inventory. During hypothesis expansion, the decoder picks a tuple from the inventory and generates the sequence of operations required for the translation with this tuple in light of the previous hypothesis.[9] The sequence of operations may include translation (generate, continue source cept etc.) and reordering (gap insertions, jumps) operations. The decoder also calculates the overall cost of the new hypothesis. Recombination is performed on hypotheses having the same coverage vector, monolingual language model context, and operation model context. We do histogram-based pruning, maintaining the 500 best hypotheses for each stack.[10]

[9]A hypothesis maintains the index of the last source word covered (j), the position of the right-most source word covered so far (Z), the number of open gaps, the number of gaps so far inserted, the previously generated operations, the generated target string, and the accumulated values of all the features discussed in Section 4.

[10]We need a higher beam size to produce translation units similar to the phrase-based systems. For example, the phrase-based system can learn the phrase pair "zum Beispiel – for example" and generate it in a single step placing it directly into the stack two words to the right. Our system generates this example with two separate tuple translations "zum – for" and "Beispiel – example" in two adjacent stacks. Because "zum – for" is not a frequent translation unit, it will be ranked quite low in the first stack until the tuple "Beispiel – example" appears in the second stack. Koehn and his colleagues have repeatedly shown that in-

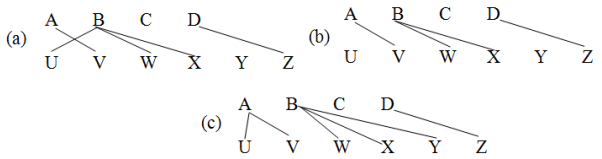


Figure 3: Post-editing of Alignments (a) Initial (b) No Target-Discontinuities (c) Final Alignments

6 Training

Training includes: (i) post-editing of the alignments, (ii) generation of the operation sequence (iii) estimation of the n-gram language models.

Our generative story does not handle target-side discontinuities and unaligned target words. Therefore we eliminate them from the training corpus in a 3-step process: If a source word is aligned with multiple target words which are not consecutive, first the link to the least frequent target word is identified, and the group of links containing this word is retained while the others are deleted. The intuition here is to keep the alignments containing content words (which are less frequent than functional words). The new alignment has no target-side discontinuities anymore, but might still contain unaligned target words. For each unaligned target word, we determine the (left or right) neighbour that it appears more frequently with and align it with the same source word as the neighbour. The result is an alignment without target-side discontinuities and unaligned target words. Figure 3 shows an illustrative example of the process. The tuples in Figure 3c are "A – U V", "B – W X Y", "C – NULL", "D – Z".

We apply Algorithm 1 to convert the preprocessed aligned corpus into a sequence of translation operations. The resulting operation corpus contains one sequence of operations per sentence pair.

In the final training step, the three language models are trained using the SRILM Toolkit. The operation model is estimated from the operation corpus. The prior probability model is estimated from the target side part of the bilingual corpus. The monolingual language model is estimated from the target side of the bilingual corpus and additional monolingual data.

creasing the Moses stack size from 200 to 1000 does not have a significant effect on translation into English, see (Koehn and Haddow, 2009) and other shared task papers.

7 Experimental Setup

7.1 Data

We evaluated the system on three data sets with German-to-English, Spanish-to-English and French-to-English news translations, respectively. We used data from the 4^{th} version of the *Europarl Corpus* and the *News Commentary* which was made available for the translation task of the *Fourth Workshop on Statistical Machine Translation*.[11] We use 200K bilingual sentences, composed by concatenating the entire news commentary ($\approx$ 74K sentences) and Europarl ($\approx$ 126K sentence), for the estimation of the translation model. Word alignments were generated with GIZA++ (Och and Ney, 2003), using the grow-diag-final-and heuristic (Koehn et al., 2005). In order to obtain the best alignment quality, the alignment task is performed on the entire parallel data and not just on the training data we use. All data is lowercased, and we use the Moses tokenizer and recapitalizer. Our monolingual language model is trained on 500K sentences. These comprise 300K sentences from the monolingual corpus (news commentary) and 200K sentences from the target-side part of the bilingual corpus. The latter part is also used to train the prior probability model. The dev and test sets are news-dev2009a and news-dev2009b which contain 1025 and 1026 parallel sentences. The feature weights are tuned with Z-MERT (Zaidan, 2009).

7.2 Results

Baseline: We compare our model to a recent version of Moses (Koehn et al., 2007) using Koehn's training scripts and evaluate with BLEU (Papineni et al., 2002). We provide Moses with the same initial alignments as we are using to train our system.[12] We use the default parameters for Moses, and a 5-gram English language model (the same as in our system).

We compare two variants of our system. The first system (Tw_{no-rl}) applies no hard reordering limit and uses the distortion and gap distance penalty features as soft constraints, allowing all possible reorderings. The second system (Tw_{rl-6}) uses no distortion and gap distance features, but applies a hard constraint which limits reordering to no more than 6

[11] http://www.statmt.org/wmt09/translation-task.html

[12] We tried applying our post-processing to the alignments provided to Moses and found that this made little difference.

Source	German	Spanish	French
$\mathbf{Bl_{no-rl}}$	17.41	19.85	19.39
$\mathbf{Bl_{rl-6}}$	18.57	21.67	20.84
$\mathbf{Tw_{no-rl}}$	**18.97**	**22.17**	20.94
$\mathbf{Tw_{rl-6}}$	**19.03**	21.88	20.72

Table 3: This Work(Tw) vs Moses (Bl), no-rl = No Reordering Limit, rl-6 = Reordering limit 6

Source	German	Spanish	French
$\mathbf{Tw_{no-rl-asg}}$	18.61	21.60	20.59
$\mathbf{Tw_{rl-6-asg}}$	18.65	21.40	20.47
$\mathbf{Tw_{no-rl-hsg}}$	18.91	21.93	20.87
$\mathbf{Tw_{rl-6-hsg}}$	**19.23**	21.79	20.85

Table 4: Our Systems with Gappy Units, asg = All Gappy Units, hsg = Heuristic for pruning Gappy Units

Source	German	Spanish	French
Gaps	965515	1705156	1473798
No-Gaps	256992	313690	343220
Heuristic (hsg)	281618	346993	385869

Table 5: 10-best Translation Options With & Without Gaps and using our Heuristic

positions. Specifically, we do not extend hypotheses that are more than 6 words apart from the first word of the left-most gap during decoding. In this experiment, we disallowed tuples which were discontinuous on the source side. We compare our systems with two Moses systems as baseline, one using no reordering limit (Bl_{no-rl}) and one using the default distortion limit of 6 (Bl_{rl-6}).

Both of our systems (see Table 3) outperform Moses on the German-to-English and Spanish-to-English tasks and get comparable results for French-to-English. Our best system (Tw_{no-rl}), which uses no hard reordering limit, gives statistically significant ($p < 0.05$)[13] improvements over Moses (both baselines) for the German-to-English and Spanish-to-English translation task. The results for Moses drop by more than a BLEU point without the reordering limit (see Bl_{no-rl} in Table 3). All our results are statistically significant over the baseline Bl_{no-rl} for all the language pairs.

In another experiment, we tested our system also with tuples which were discontinuous on the source side. These gappy translation units neither improved the performance of the system with hard reordering limit ($Tw_{rl-6-asg}$) nor that of the system without reordering limit ($Tw_{no-rl-asg}$) as Table 4 shows. In an analysis of the output we found two reasons for this result: (i) Using tuples with source gaps increases the list of extracted n-best translation tuples exponentially which makes the search problem even more difficult. Table 5 shows the number of tuples (with and without gaps) extracted when decoding the test file with 10-best translations. (ii) The future cost[14] is poorly estimated in case of tuples with gappy source cepts, causing search errors.

In an experiment, we deleted gappy tuples with a score (future cost estimate) lower than the sum of the best scores of the parts. This heuristic removes many useless discontinuous tuples. We found that results improved ($Tw_{no-rl-hsg}$ and $Tw_{rl-6-hsg}$ in Table 4) compared to the version using all gaps ($Tw_{no-rl-asg}$, $Tw_{rl-6-asg}$), and are closer to the results without discontinuous tuples (Tw_{no-rl} and Tw_{rl-6} in Table 3).

8 Sample Output

In this section we compare the output of our systems and Moses. Example 1 in Figure 4 shows the powerful reordering mechanism of our model which moves the English verb phrase "do not want to negotiate" to its correct position between the subject "they" and the prepositional phrase "about concrete figures". Moses failed to produce the correct word order in this example. Notice that although our model is using smaller translation units "nicht – do not", "verhandlen – negotiate" and "wollen – want to", it is able to memorize the phrase translation "nicht verhandlen wollen – do not want to negotiate" as a sequence of translation and reordering operations. It learns the reordering of "verhandlen – negotiate" and "wollen – want to" and also captures dependencies across phrase boundaries.

Example 2 shows how our system without a reordering limit moves the English translation "vote" of the German clause-final verb "stimmen" across about 20 English tokens to its correct position behind the auxiliary "would".

Example 3 shows how the system with gappy tuples translates a German sentence with the particle verb "kehrten...zurück" using a single tuple (dashed lines). Handling phenomena like particle verbs

[13] We used Kevin Gimpel's implementation of pairwise bootstrap resampling (Koehn, 2004b), 1000 samples.

[14] The dynamic programming approach of calculating future cost for bigger spans gives erroneous results when gappy cepts can interleave. Details omitted due to space limitations.

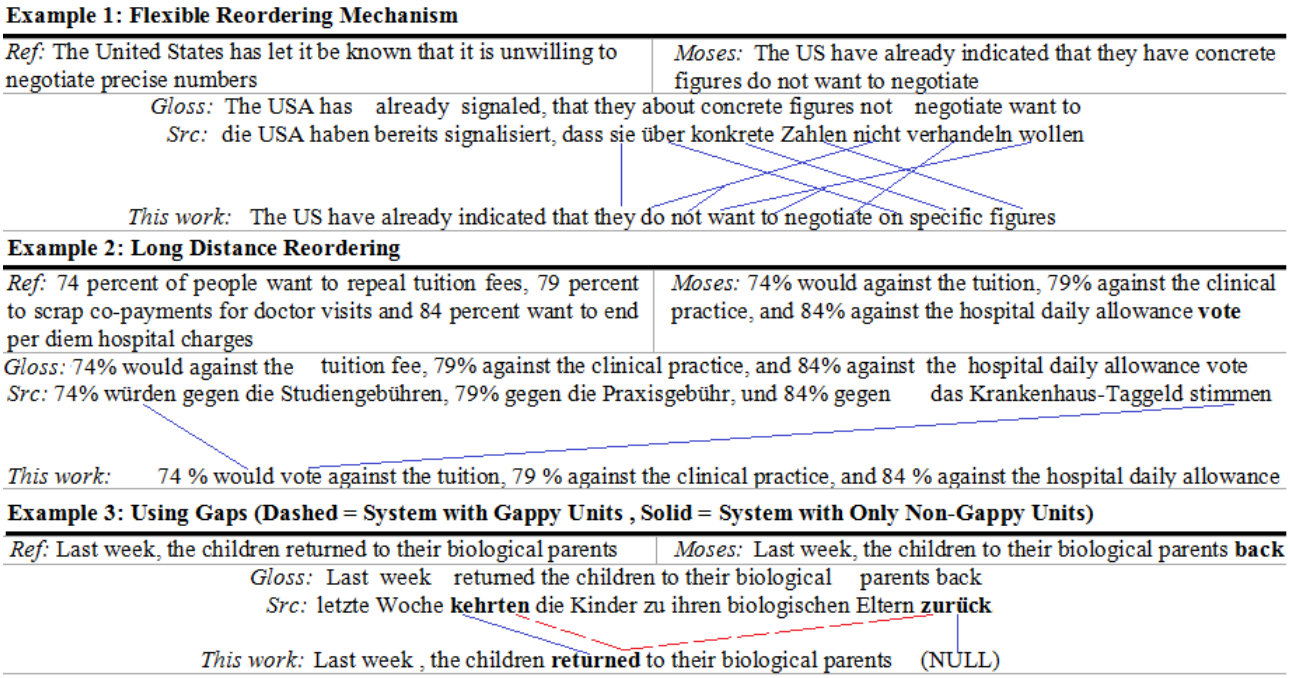


Figure 4: Sample Output Sentences

strongly motivates our treatment of source side gaps. The system without gappy units happens to produce the same translation by translating "kehrten" to "returned" and deleting the particle "zurück" (solid lines). This is surprising because the operation for translating "kehrten" to "returned" and for deleting the particle are too far apart to influence each other in an n-gram model. Moses run on the same example deletes the main verb ("kehrten"), an error that we frequently observed in the output of Moses.

Our last example (Figure 5) shows that our model learns idioms like "meiner Meinung nach – In my opinion ," and short phrases like "gibt es – there are" showing its ability to memorize these "phrasal" translations, just like Moses.

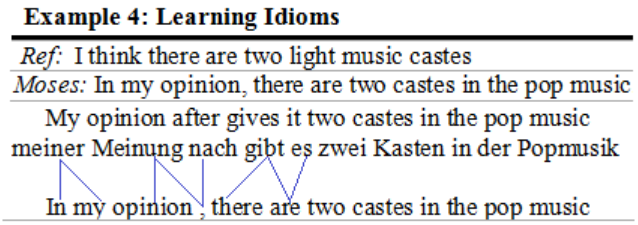


Figure 5: Learning Idioms

9 Conclusion

We have presented a new model for statistical MT which can be used as an alternative to phrase-based translation. Similar to N-gram based MT, it addresses three drawbacks of traditional phrasal MT by better handling dependencies across phrase boundaries, using source-side gaps, and solving the phrasal segmentation problem. In contrast to N-gram based MT, our model has a generative story which tightly couples translation and reordering. Furthermore it considers all possible reorderings unlike N-gram systems that perform search only on a limited number of pre-calculated orderings. Our model is able to correctly reorder words across large distances, and it memorizes frequent phrasal translations including their reordering as probable operations sequences. Our system outperformed Moses on standard Spanish-to-English and German-to-English tasks and achieved comparable results for French-to-English. A binary version of the corpus conversion algorithm and the decoder is available.[15]

Acknowledgments

The authors thank Fabienne Braune and the reviewers for their comments. Nadir Durrani was funded by the Higher Education Commission (HEC) of Pakistan. Alexander Fraser was funded by Deutsche Forschungsgemeinschaft grant Models of Morphosyntax for Statistical Machine Translation. Helmut Schmid was supported by Deutsche Forschungsgemeinschaft grant SFB 732.

[15] http://www.ims.uni-stuttgart.de/∼durrani/resources.html

References

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

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

Josep Maria Crego and Franois Yvon. 2009. Gappy translation units under left-to-right smt decoding. In *Proceedings of the meeting of the European Association for Machine Translation (EAMT)*, pages 66–73, Barcelona, Spain.

Josep Maria Crego and François Yvon. 2010. Improving reordering with linguistically informed bilingual n-grams. In *Coling 2010: Posters*, pages 197–205, Beijing, China, August. Coling 2010 Organizing Committee.

Josep M. Crego, Marta R. Costa-juss, Jos B. Mario, and Jos A. R. Fonollosa. 2005a. Ngram-based versus phrasebased statistical machine translation. In *In Proceedings of the International Workshop on Spoken Language Technology (IWSLT05*, pages 177–184.

Josep M. Crego, José B. Mariño, and Adrià de Gispert. 2005b. Reordered search and unfolding tuples for ngram-based SMT. In *Proceedings of the 10th Machine Translation Summit (MT Summit X)*, pages 283–289, Phuket, Thailand.

Michel Galley and Christopher D. Manning. 2010. Accurate non-hierarchical phrase-based translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 966–974, Los Angeles, California, June. Association for Computational Linguistics.

Philipp Koehn and Barry Haddow. 2009. Edinburgh's submission to all tracks of the WMT 2009 shared task with reordering and speed improvements to Moses. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 160–164, Athens, Greece, March. Association for Computational Linguistics.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference*, pages 127–133, Edmonton, Canada.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 iwslt speech translation evaluation. In *International Workshop on Spoken Language Translation 2005*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Demonstration Program*, Prague, Czech Republic.

Philipp Koehn. 2004a. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *AMTA*, pages 115–124.

Philipp Koehn. 2004b. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Zhifei Li, Chris Callison-burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren N. G. Thornton, Jonathan Weese, and Omar F. Zaidan. 2009. Joshua: An open source toolkit for parsing-based machine translation.

J.B. Mariño, R.E. Banchs, J.M. Crego, A. de Gispert, P. Lambert, J.A.R. Fonollosa, and M.R. Costa-jussà. 2006. N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549.

I. Dan Melamed. 2004. Statistical machine translation by parsing. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz J. Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(1):417–449.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Morristown, NJ, USA. Association for Computational Linguistics.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Intl. Conf. Spoken Language Processing*, Denver, Colorado.

Omar F. Zaidan. 2009. Z-MERT: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.

Integrating surprisal and uncertain-input models in online sentence comprehension: formal techniques and empirical results

Roger Levy
Department of Linguistics
University of California at San Diego
9500 Gilman Drive # 0108
La Jolla, CA 92093-0108
rlevy@ucsd.edu

Abstract

A system making optimal use of available information in incremental language comprehension might be expected to use linguistic knowledge together with current input to revise beliefs about previous input. Under some circumstances, such an error-correction capability might induce comprehenders to adopt grammatical analyses that are inconsistent with the true input. Here we present a formal model of how such input-unfaithful garden paths may be adopted and the difficulty incurred by their subsequent disconfirmation, combining a rational noisy-channel model of syntactic comprehension under uncertain input with the surprisal theory of incremental processing difficulty. We also present a behavioral experiment confirming the key empirical predictions of the theory.

1 Introduction

In most formal theories of human sentence comprehension, input recognition and syntactic analysis are taken to be distinct processes, with the only feedback from syntax to recognition being prospective prediction of likely upcoming input (Jurafsky, 1996; Narayanan and Jurafsky, 1998, 2002; Hale, 2001, 2006; Levy, 2008a). Yet a system making optimal use of all available information might be expected to perform fully joint inference on sentence identity and structure given perceptual input, using linguistic knowledge both prospectively *and* retrospectively in drawing inferences as to how raw input should be segmented and recognized as a sequence of linguistic tokens, and about the degree to which each input token should be trusted during grammatical analysis. Formal models of such joint inference over uncertain input have been proposed (Levy, 2008b), and corroborative empirical evidence exists that strong coherence of current input with a perceptual neighbor of previous input may induce confusion in comprehenders as to the identity of that previous input (Connine et al., 1991; Levy et al., 2009).

In this paper we explore a more dramatic prediction of such an uncertain-input theory: that, when faced with sufficiently biasing input, comprehenders might under some circumstances adopt a grammatical analysis inconsistent with the true raw input comprising a sentence they are presented with, but consistent with a slightly perturbed version of the input that has higher prior probability. If this is the case, then subsequent input strongly disconfirming this "hallucinated" garden-path analysis might be expected to induce the same effects as seen in classic cases of garden-path disambiguation traditionally studied in the psycholinguistic literature. We explore this prediction by extending the rational uncertain-input model of Levy (2008b), integrating it with SURPRISAL THEORY (Hale, 2001; Levy, 2008a), which successfully accounts for and quantifies traditional garden-path disambiguation effects; and by testing predictions of the extended model in a self-paced reading study. Section 2 reviews surprisal theory and how it accounts for traditional garden-path effects. Section 3 provides background information on garden-path effects relevant to the current study, describes how we might hope to reveal comprehenders' use of grammatical knowledge to revise beliefs about the identity of previous linguistic sur-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1055–1065,
Portland, Oregon, June 19-24, 2011.

face input and adopt grammatical analyses inconsistent with true input through a controlled experiment, and informally outlines how such belief revisions might arise as a side effect in a general theory of rational comprehension under uncertain input. Section 4 defines and estimates parameters for a model instantiating the general theory, and describes the predictions of the model for the experiment described in Section 3 (along with the inference procedures required to determine those predictions). Section 5 reports the results of the experiment. Section 6 concludes.

2 Garden-path disambiguation under surprisal

The SURPRISAL THEORY of incremental sentence-processing difficulty (Hale, 2001; Levy, 2008a) posits that the cognitive effort required to process a given word w_i of a sentence in its context is given by the simple information-theoretic measure of the log of the inverse of the word's conditional probability (also called its "surprisal" or "Shannon information content") in its intra-sentential context $w_{1,\ldots,i-1}$ and extra-sentential context *Ctxt*:

$$\text{Effort}(w_i) \propto \log \frac{1}{P(w_i|w_{1\ldots i-1}, \textit{Ctxt})}$$

(In the rest of this paper, we consider isolated-sentence comprehension and ignore *Ctxt*.) The theory derives empirical support not only from controlled experiments manipulating grammatical context but also from broad-coverage studies of reading times for naturalistic text (Demberg and Keller, 2008; Boston et al., 2008; Frank, 2009; Roark et al., 2009), including demonstration that the shape of the relationship between word probability and reading time is indeed log-linear (Smith and Levy, 2008).

Surprisal has had considerable success in accounting for one of the best-known phenomena in psycholinguistics, the GARDEN-PATH SENTENCE (Frazier, 1979), in which a local ambiguity biases the comprehender's incremental syntactic interpretation so strongly that upon encountering disambiguating input the correct interpretation can only be recovered with great effort, if at all. The most famous example is (1) below (Bever, 1970):

(1) The horse raced past the barn fell.

where the context before the final word is strongly biased toward an interpretation where *raced* is the main verb of the sentence (**MV**; Figure 1a), the intended interpretation, where *raced* begins a reduced relative clause (**RR**; Figure 1b) and *fell* is the main verb, is extremely difficult to recover. Letting T_j range over the possible incremental syntactic analyses of words $w_{1\ldots6}$ preceding *fell*, under surprisal the conditional probability of the disambiguating continuation *fell* can be approximated as

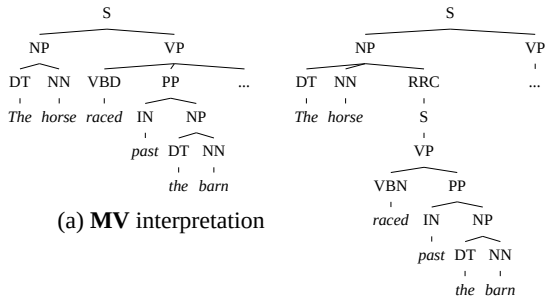
$$P(\textit{fell}|w_{1\ldots6}) = \sum_j P(\textit{fell}|T_j, w_{1\ldots6}) P(T_j|w_{1\ldots6}) \quad \text{(I)}$$

For all possible predisambiguation analyses T_j, either the analysis is disfavored by the context ($P(T_j|w_{1\ldots6})$ is low) or the analysis makes the disambiguating word unlikely ($P(\textit{fell}|T_j, w_{1\ldots6})$ is low). Since every summand in the marginalization of Equation (I) has a very small term in it, the total marginal probability is thus small and the surprisal is high. Hale (2001) demonstrated that surprisal thus predicts strong garden-pathing effects in the classic sentence *The horse raced past the barn fell* on basis of the overall rarity of reduced relative clauses alone. More generally, Jurafsky (1996) used a combination of syntactic probabilities (reduced RCs are rare) and argument-structure probabilities (*raced* is usually intransitive) to estimate the probability ratio of the two analyses of pre-disambiguation context in Figure 1 as roughly 82:1, putting a lower bound on the additional surprisal incurred at *fell* for the reduced-RC variant over the unreduced variant (*The horse that was raced past the barn fell*) of 6.4 bits.[1]

3 Garden-pathing and input uncertainty

We now move on to cases where garden-pathing can apparently be blocked by only small changes to the surface input, which we will take as a starting point for developing an integrated theory of uncertain-input inference and surprisal. The backdrop is what is known in the psycholinguistic literature as the **NP/Z** ambiguity, exemplified in (2) below:

[1] We say that this is a "lower bound" because incorporating even finer-grained information—such as the fact that *horse* is a canonical subject for intransitive *raced*—into the estimate would almost certainly push the probability ratio even farther in favor of the main-clause analysis.



(a) **MV** interpretation

(b) **RR** interpretation

Figure 1: Classic garden pathing

(2) While Mary was mending the socks fell off her lap.

In incremental comprehension, the phrase *the socks* is ambiguous between being the **NP** object of the preceding subordinate-clause verb *mending* versus being the subject of the main clause (in which case *mending* has a **Z**ero object); in sentences like (2) the initial bias is toward the **NP** interpretation. The main-clause verb *fell* disambiguates, ruling out the initially favored **NP** analysis. It has been known since Frazier and Rayner (1982) that this effect of garden-path disambiguation can be measured in reading times on the main-clause verb (see also Mitchell, 1987; Ferreira and Henderson, 1993; Adams et al., 1998; Sturt et al., 1999; Hill and Murray, 2000; Christianson et al., 2001; van Gompel and Pickering, 2001; Tabor and Hutchins, 2004; Staub, 2007). Small changes to the context can have huge effects on comprehenders' initial interpretations, however. It is unusual for sentence-initial subordinate clauses not to end with a comma or some other type of punctuation (searches in the parsed Brown corpus put the rate at about 18%); empirically it has consistently been found that a comma eliminates the garden-path effect in NP/Z sentences:

(3) While Mary was mending, the socks fell off her lap.

Understanding sentences like (3) is intuitively much easier, and reading times at the disambiguating verb are reliably lower when compared with (2). Fodor (2002) summarized the power of this effect succinctly:

> [w]ith a comma after *mending*, there would be no syntactic garden path left to be studied. (Fodor, 2002)

In a surprisal model with clean, veridical input, Fodor's conclusion is exactly what is predicted: separating a verb from its direct object with a comma effectively never happens in edited, published written English, so the conditional probability of the **NP** analysis should be close to zero.[2] When uncertainty about surface input is introduced, however—due to visual noise, imperfect memory representations, and/or beliefs about possible speaker error—analyses come into play in which some parts of the true string are treated as if they were absent. In particular, because the two sentences are perceptual neighbors, the pre-disambiguation garden-path analysis of (2) may be entertained in (3).

We can get a tighter handle on the effect of input uncertainty by extending Levy (2008b)'s analysis of the *expected* beliefs of a comprehender about the sequence of words constituting an input sentence to joint inference over both sentence identity and sentence structure. For a true sentence $\mathbf{w}^*$ which yields perceptual input I, joint inference on sentence identity $\mathbf{w}$ and structure T marginalizing over I yields:

$$P_C(T, \mathbf{w}|\mathbf{w}^*) = \int_I P_C(T, \mathbf{w}|I, \mathbf{w}^*) P_T(I|\mathbf{w}^*)\, dI$$

where $P_T(I|\mathbf{w}^*)$ is the true model of noise (perceptual inputs derived from the true sentence) and $P_C(\cdot)$ terms reflect the comprehender's linguistic knowledge and beliefs about the noise processes intervening between intended sentences and perceptual input. $\mathbf{w}^*$ and $\mathbf{w}$ must be conditionally independent given I since $\mathbf{w}^*$ is not observed by the comprehender, giving us (through Bayes' Rule):

$$P(T, \mathbf{w}|\mathbf{w}^*) = \int_I \frac{P_C(I|T, \mathbf{w}) P_C(T, \mathbf{w})}{P_C(I)} P_T(I|\mathbf{w}^*)\, dI$$

For present purposes we constrain the comprehender's model of noise so that T and I are conditionally independent given $\mathbf{w}$, an assumption that can be relaxed in future work.[3] This allows us the further

[2] A handful of `VP -> V , NP ...` rules can be found in the Penn Treebank, but they all involve appositives (*It [$_{VP}$ ran, this apocalyptic beast . . .]*), vocatives (*You should [$_{VP}$ understand, Jack, . . .]*), cognate objects (*She [$_{VP}$ smiled, a smile without humor]*), or indirect speech (*I [$_{VP}$ thought, you nasty brute. . .]*); none involve true direct objects of the type in (3).

[3] This assumption is effectively saying that noise processes are syntax-insensitive, which is clearly sensible for environmental noise but would need to be relaxed for some types of speaker error.

simplification to

$$P(T, \mathbf{w}|\mathbf{w}^*) = \overbrace{P_C(T, \mathbf{w})}^{\text{(i)}} \overbrace{\int_I \frac{P_C(I|\mathbf{w}) P_T(I|\mathbf{w}^*)}{P_C(I)} \, dI}^{\text{(ii)}} \quad \text{(II)}$$

That is, a comprehender's average inferences about sentence identity and structure involve a tradeoff between (i) the prior probability of a grammatical derivation given a speaker's linguistic knowledge and (ii) the fidelity of the derivation's yield to the true sentence, as measured by a combination of true noise processes and the comprehender's beliefs about those processes.

3.1 Inducing hallucinated garden paths through manipulating prior grammatical probabilities

Returning to our discussion of the NP/Z ambiguity, the relative ease of comprehending (3) entails an interpretation in the uncertain-input model that the cost of infidelity to surface input is sufficient to prevent comprehenders from deriving strong belief in a *hallucinated* garden-path analysis of (3) pre-disambiguation in which the comma is ignored. At the same time, the uncertain-input theory predicts that if we manipulate the balance of prior grammatical probabilities $P_C(T, \mathbf{w})$ strongly enough (term (i) in Equation (II)), it may shift the comprehender's beliefs toward a garden-path interpretation. This observation sets the stage for our experimental manipulation, illustrated below:

(4) As the soldiers marched, toward the tank lurched an injured enemy combatant.

Example (4) is qualitatively similar to (3), but with two crucial differences. First, there has been LOCATIVE INVERSION (Bolinger, 1971; Bresnan, 1994) in the main clause: a locative PP has been fronted before the verb, and the subject NP is realized postverbally. Locative inversion is a low-frequency construction, hence it is crucially disfavored by the comprehender's prior over possible grammatical structures. Second, the subordinate-clause verb is no longer transitive, as in (3); instead it is intransitive but could itself take the main-clause fronted PP as a dependent. Taken together, these properties should shift comprehenders' posterior inferences given prior grammatical knowledge and pre-disambiguation input more sharply than in (3) toward the input-unfaithful interpretation in which the immediately preverbal main-clause constituent (*toward the tank* in (4)) is interpreted as a dependent of the subordinate-clause verb, as if the comma were absent.

If comprehenders do indeed seriously entertain such interpretations, then we should be able to find the empirical hallmarks (e.g., elevated reading times) of garden-path disambiguation at the main-clause verb *lurched*, which is incompatible with the "hallucinated" garden-path interpretation. Empirically, however, it is important to disentangle these empirical hallmarks of garden-path disambiguation from more general disruption that may be induced by encountering locative inversion itself. We address this issue by introducing a control condition in which a postverbal PP is placed within the subordinate clause:

(5) As the soldiers marched *into the bunker*, toward the tank lurched an injured enemy combatant. **[+PP]**

Crucially, this PP fills a similar thematic role for the subordinate-clause verb *marched* as the main-clause fronted PP would, reducing the extent to which the comprehender's prior favors the input-unfaithful interpretation (that is, the prior ratio $\frac{P(\textit{marched into the bunker toward the tank}|\text{VP})}{P(\textit{marched into the bunker}|\text{VP})}$ for (5) is much lower than the corresponding prior ratio $\frac{P(\textit{marched toward the tank}|\text{VP})}{P(\textit{marched}|\text{VP})}$ for (4)), while leaving locative inversion present. Finally, to ensure that sentence length itself does not create a confound driving any observed processing-time difference, we cross presence/absence of the subordinate-clause PP with inversion in the main clause:

(6)

a. As the soldiers marched, the tank lurched toward an injured enemy combatant. **[Uninverted,−PP]**
b. As the soldiers marched into the bunker, the tank lurched toward an injured enemy combatant. **[Uninverted,+PP]**

4 Model instantiation and predictions

To determine the predictions of our uncertain-input/surprisal model for the above sentence types, we extracted a small grammar from the parsed

TOP	→ S .	1.000000
S	→ INVERTED NP	0.003257
S	→ SBAR S	0.012289
S	→ SBAR , S	0.041753
S	→ NP VP	0.942701
INVERTED	→ PP VBD	1.000000
SBAR	→ INSBAR S	1.000000
VP	→ VBD RB	0.002149
VP	→ VBD PP	0.202024
VP	→ VBD NP	0.393660
VP	→ VBD PP PP	0.028029
VP	→ VBD RP	0.005731
VP	→ VBD	0.222441
VP	→ VBD JJ	0.145966
PP	→ IN NP	1.000000
NP	→ DT NN	0.274566
NP	→ NNS	0.047505
NP	→ NNP	0.101198
NP	→ DT NNS	0.045082
NP	→ PRP	0.412192
NP	→ NN	0.119456

Table 1: A small PCFG (lexical rewrite rules omitted) covering the constructions used in (4)–(6), with probabilities estimated from the parsed Brown corpus.

Brown corpus (Kučera and Francis, 1967; Marcus et al., 1994), covering sentence-initial subordinate clause and locative-inversion constructions.[4,5] The non-terminal rewrite rules are shown in Table 1, along with their probabilities; of terminal rewrite rules for all words which either appear in the sentences to be parsed or appeared at least five times in the corpus, with probabilities estimated by relative frequency.

As we describe in the following two sections, uncertain input is represented as a weighted finite-state automaton (WFSA), allowing us to represent the incremental inferences of the comprehender through intersection of the input WFSA with the PCFG above (Bar-Hillel et al., 1964; Nederhof and Satta, 2003, 2008).

[4]Rule counts were obtained using `tgrep2/Tregex` patterns (Rohde, 2005; Levy and Andrew, 2006); the probabilities given are relative frequency estimates. The patterns used can be found at `http://idiom.ucsd.edu/~rlevy/papers/acl2011/tregex_patterns.txt`.

[5]Similar to the case noted in Footnote 2, a small number of `VP -> V , PP ...` rules can be found in the parsed Brown corpus. However, the PPs involved are overwhelmingly (i) set expressions, such as *for example*, *in essence*, and *of course*, or (ii) manner or temporal adjuncts. The handful of true locative PPs (5 in total) are all parentheticals intervening between the verb and a complement strongly selected by the verb (e.g., *[$_{VP}$ means, in my country, homosexual]*); none fulfill one of the verb's thematic requirements.

4.1 Uncertain-input representations

Levy (2008a) introduced the LEVENSHTEIN-DISTANCE KERNEL as a model of the average effect of noise in uncertain-input probabilistic sentence comprehension; this corresponds to term (ii) in our Equation (II). This kernel had a single noise parameter governing scaling of the cost of considering word *substitutions*, *insertions*, and *deletions* are considered, with the cost of a word substitution falling off exponentially with Levenshtein distance between the true word and the substituted word, and the cost of word insertion or deletion falling off exponentially with word length. The distribution over the infinite set of strings $\mathbf{w}$ can be encoded in a weighted finite-state automaton, facilitating efficient inference.

We use the Levenshtein-distance kernel here to capture the effects of perceptual noise, but make two modifications necessary for incremental inference and for the correct computation of surprisal values for new input: the distribution over already-seen input must be *proper*, and possible future inputs must be *costless*. The resulting weighted finite-state representation of noisy input for a true sentence prefix $\mathbf{w}^* = w_{1\ldots j}$ is a $j+1$-state automaton with arcs as follows:

- For each $i \in 1, \ldots, j$:
  - A *substitution* arc from $i-1$ to i with cost proportional to $\exp[-\text{LD}(w', w_i)\ \gamma]$ for each word w' in the lexicon, where $\gamma > 0$ is a noise parameter and $\text{LD}(w', w_i)$ is the Levenshtein distance between w' and w_i (when $w' = w_i$ there is no change to the word);
  - A *deletion* arc from $i-1$ to i labeled ϵ with cost proportional to $\exp[-\text{len}(w_i)/\gamma]$;
  - An *insertion* loop arc from $i-1$ to $i-1$ with cost proportional to $\exp[-\text{len}(w')/\gamma]$ for every word w' in the lexicon;
- A loop arc from j to j for each word w' in

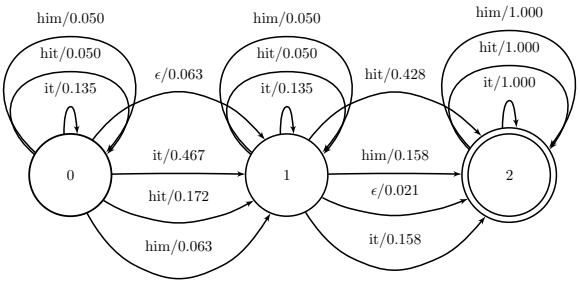


Figure 2: Noisy WFSA for partial input *it hit...* with lexicon {*it*,*hit*,*him*}, noise parameter γ=1

the lexicon, with zero cost (value 1 in the real semiring);

- State j is a zero-cost final state; no other states are final.

The addition of loop arcs at state n allows modeling of *incremental* comprehension through the automaton/grammar intersection (see also Hale, 2006); and the fact that these arcs are costless ensures that the partition function of the intersection reflects only the grammatical prior plus the costs of input already seen. In order to ensure that the distribution over already-seen input is proper, we normalize the costs on outgoing arcs from all states but j.[6] Figure 2 gives an example of a simple WFSA representation for a short partial input with a small lexicon.

4.2 Inference

Computing the surprisal incurred by the disambiguating element given an uncertain-input representation of the sentence involves a standard application of the definition of conditional probability (Hale, 2001):

$$\log \frac{1}{P(I_{1...i}|I_{1...i-1})} = \log \frac{P(I_{1...i-1})}{P(I_{1...i})} \qquad \text{(III)}$$

Since our uncertain inputs $I_{1...k}$ are encoded by a WFSA, the probability $P(I_{1...k})$ is equal to the partition function of the intersection of this WFSA with the PCFG given in Table 1.[7] PCFGs are a special class of weighted context-free grammars (WCFGs), which are closed under intersection with WFSAs; a constructive procedure exists for finding the intersection (Bar-Hillel et al., 1964; Nederhof and Satta, 2003). Hence we are left with finding the partition function of a WCFG, which cannot be computed exactly, but a number of approximation methods are known (Stolcke, 1995; Smith and Johnson, 2007; Nederhof and Satta, 2008). In practice, the computation required to compute the partition function under any of these methods increases with the size of the WCFG resulting from the intersection, which for a binarized PCFG with R rules and an n-state WFSA is Rn^2. To increase efficiency we implemented what is to our knowledge a novel method for finding the minimal grammar including all rules that will have non-zero probability in the intersection. We first parse the WFSA bottom-up with the item-based method of Goodman (1999) in the Boolean semiring, storing partial results in a chart. After completion of this bottom-up parse, every rule that will have non-zero probability in the intersection PCFG will be identifiable with a set of entries in the chart, but not all entries in this chart will have non-zero probability, since some are not connected to the root. Hence we perform a second, top-down Boolean-semiring parsing pass on the bottom-up chart, throwing out entries that cannot be derived from the root. We can then include in the intersection grammar only those rules from the classic construction that can be identified with a set of surviving entries in the final parse chart.[8] The partition functions for each category in this intersection grammar can then be computed; we used a fixed-point method preceded by a topological sort on the grammar's ruleset, as described by Nederhof and Satta (2008). To obtain the surprisal of the input deriving from a word w_i in its context, we can thus com-

[6] If a state's total unnormalized cost of insertion arcs is α and that of deletion and insertion arcs is β, its normalizing constant is $\frac{\beta}{1-\alpha}$. Note that we must have $\alpha < 1$, placing a constraint on the value that γ can take (above which the normalizing constant diverges).

[7] Using the WFSA representation of average noise effects here actually involves one simplifying assumption, that the average surprisal of I_i, or $E_{P_T}\left[\log \frac{1}{P_C(I_i|I_{1...i-1})}\right]$, is well approximated by the log of the ratio of the expected probabilities of the noisy inputs $I_{1...i-1}$ and $I_{1...i}$, since as discussed in Section 3 the quantities $P(I_{1...i-1})$ and $P(I_{1...i})$ are expectations under the true noise distribution. This simplifying assumption has the advantage of bypassing commitment to a specific representation of perceptual input and should be justifiable for reasonable noise functions, but the issue is worth further scrutiny.

[8] Note that a standard top-down algorithm such as Earley parsing cannot be used to avoid the need for both bottom-up and top-down passes, since the presence of loops in the WFSA breaks the ability to operate strictly left-to-right.

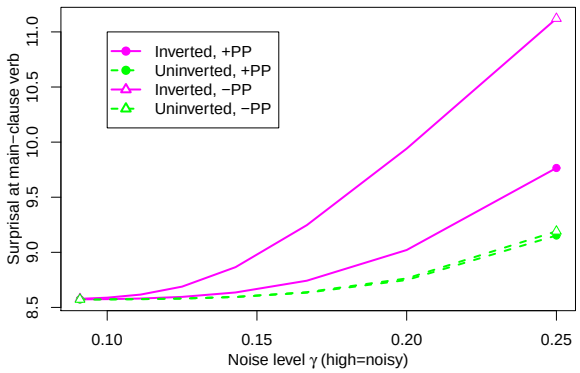


Figure 3: Model predictions for (4)–(6)

pute the partition functions for noisy inputs $I_{1...i-1}$ and $I_{1...i}$ corresponding to words $w_{1...i-1}$ and words $w_{1...i}$ respectively, and take the log of their ratio as in Equation (III).

4.3 Predictions

The noise level γ is a free parameter in this model, so we plot model predictions—the expected surprisal of input from the main-clause verb for each variant of the target sentence in (4)–(6)—over a wide range of its possible values (Figure 3). The far left of the graph asymptotes toward the predictions of *clean surprisal*, or noise-free input. With little to no input uncertainty, the presence of the comma rules out the garden-path analysis of the fronted PP *toward the tank*, and the surprisal at the main-clause verb is the same across condition (here reflecting only the uncertainty of verb identity for this small grammar). As input uncertainty increases, however, surprisal in the **[Inverted, −PP]** condition increases, reflecting the stronger belief given preceding context in an input-unfaithful interpretation.

5 Empirical results

To test these predictions we conducted a word-by-word self-paced reading study, in which participants read by pressing a button to reveal each successive word in a sentence; times between button presses are recorded and analyzed as an index of incremental processing difficulty (Mitchell, 1984). Forty monolingual native-English speaker participants read twenty-four sentence quadruplets ("items") on the pattern of (4)–(6), with a Latin-square design so that each participant saw an equal

	Inverted	Uninverted
-PP	0.76	0.93
+PP	0.85	0.92

Table 2: Question-answering accuracy

number of sentences in each condition and saw each item only once. Experimental items were pseudo-randomly interspersed with 62 filler sentences; no two experimental items were ever adjacent. Punctuation was presented with the word to its left, so that for (4) the four and fifth button presses would yield

```
-------------- marched, --------------
```

and

```
----------------------- toward --------
```

respectively (right-truncated here for reasons of space). Every sentence was followed by a yes/no comprehension question (e.g., *Did the tank lurch toward an injured enemy combatant?*); participants received feedback whenever they answered a question incorrectly.

Reading-time results are shown in Figure 4. As can be seen, the model's predictions are matched at the main-clause verb: reading times are highest in the **[Inverted, −PP]** condition, and there is an interaction between main-clause inversion and presence of a subordinate-clause PP such that presence of the latter reduces reading times more for inverted than for uninverted main clauses. This interaction is significant in both by-participants and by-items ANOVAs (both $p < 0.05$) and in a linear mixed-effects analysis with participants- and item-specific random interactions ($t > 2$; see Baayen et al., 2008). The same pattern persists and remains significant through to the end of the sentence, indicating considerable processing disruption, and is also observed in question-answering accuracies for experimental sentences, which are superadditively lowest in the **[Inverted, −PP]** condition (Table 2).

The inflated reading times for the **[Inverted, −PP]** condition beginning at the main-clause verb confirm the predictions of the uncertain-input/surprisal theory. Crucially, the input that would on our theory induce the comprehender to question the comma (the fronted main-clause PP)

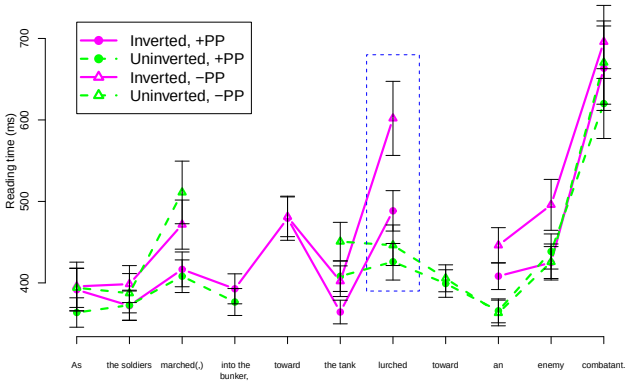


Figure 4: Average reading times for each part of the sentence, broken down by experimental condition

is not seen until after the comma is no longer visible (and presumably has been integrated into beliefs about syntactic analysis on veridical-input theories). This empirical result is hence difficult to accommodate in accounts which do not share our theory's crucial property that comprehenders can revise their belief in previous input on the basis of current input.

6 Conclusion

Language is redundant: the content of one part of a sentence carries predictive value both for what will precede and what will follow it. For this reason, and because the path from a speaker's intended utterance to a comprehender's perceived input is noisy and error-prone, a comprehension system making optimal use of available information would use current input not only for forward prediction but also to assess the veracity of previously encountered input. Here we have developed a theory of how such an adaptive error-correcting capacity is a consequence of noisy-channel inference, with a comprehender's beliefs regarding sentence form and structure at any moment in incremental comprehension reflecting a balance between fidelity to perceptual input and a preference for structures with higher prior probability. As a consequence of this theory, certain types of sentence contexts will cause the drive toward higher prior-probability analyses to overcome the drive to maintain fidelity to input, undermining the comprehender's belief in an earlier part of the input actually perceived in favor of an analysis unfaithful to part of the true input. If subsequent input strongly disconfirms this incorrect interpretation, we should see behavioral signatures of classic garden-path disambiguation. Within the theory, the size of this "hallucinated" garden-path effect is indexed by the surprisal value under uncertain input, marginalizing over the actual sentence observed. Based on a model implementing theory we designed a controlled psycholinguistic experiment making specific predictions regarding the role of fine-grained grammatical context in modulating comprehenders' strength of belief in a highly specific bit of linguistic input—a comma marking the end of a sentence-initial subordinate clause—and tested those predictions in a self-paced reading experiment. As predicted by the theory, reading times at the word disambiguating the "hallucinated" garden-path were inflated relative to control conditions. These results contribute to the theory of uncertain-input effects in online sentence processing by suggesting that comprehenders may be induced not only to entertain but to adopt relatively strong beliefs in grammatical analyses that require modification of the surface input itself. Our results also bring a new degree of nuance to surprisal theory, demonstrating that perceptual neighbors of true preceding input may need to be taken into account in order to estimate how surprising a comprehender will find subsequent input to be.

Beyond the domain of psycholinguistics, the methods employed here might also be usefully applied to practical problems such as parsing of degraded or fragmentary sentence input, allowing joint constraint derived from grammar and available input to fill in gaps (Lang, 1988). Of course, practical applications of this sort would raise challenges of their own, such as extending the grammar to broader coverage, which is delicate here since the surface input places a weaker check on overgeneration from the grammar than in traditional probabilistic parsing. Larger grammars also impose a technical burden since parsing uncertain input is in practice more computationally intensive than parsing clean input, raising the question of what approximate-inference algorithms might be well-suited to processing uncertain input with grammatical knowledge. Answers to this question might in turn be of interest for sentence processing, since the exhaustive-parsing idealization employed here is not psychologically plausible. It seems likely that human comprehension in-

volves approximate inference with severely limited memory that is nonetheless highly optimized to recover something close to the intended meaning of an utterance, even when the recovered meaning is not completely faithful to the input itself. Arriving at models that closely approximate this capacity would be of both theoretical and practical value.

Acknowledgments

Parts of this work have benefited from presentation at the 2009 Annual Meeting of the Linguistic Society of America and the 2009 CUNY Sentence Processing Conference. I am grateful to Natalie Katz and Henry Lu for assistance in preparing materials and collecting data for the self-paced reading experiment described here. This work was supported by a UCSD Academic Senate grant, NSF CAREER grant 0953870, and NIH grant 1R01HD065829-01.

References

Adams, B. C., Clifton, Jr., C., and Mitchell, D. C. (1998). Lexical guidance in sentence processing? *Psychonomic Bulletin & Review*, 5(2):265–270.

Baayen, R. H., Davidson, D. J., and Bates, D. M. (2008). Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, 59(4):390–412.

Bar-Hillel, Y., Perles, M., and Shamir, E. (1964). On formal properties of simple phrase structure grammars. In *Language and Information: Selected Essays on their Theory and Application*. Addison-Wesley.

Bever, T. (1970). The cognitive basis for linguistic structures. In Hayes, J., editor, *Cognition and the Development of Language*, pages 279–362. John Wiley & Sons.

Bolinger, D. (1971). A further note on the nominal in the progressive. *Linguistic Inquiry*, 2(4):584–586.

Boston, M. F., Hale, J. T., Kliegl, R., Patil, U., and Vasishth, S. (2008). Parsing costs as predictors of reading difficulty: An evaluation using the Potsdam sentence corpus. *Journal of Eye Movement Research*, 2(1):1–12.

Bresnan, J. (1994). Locative inversion and the architecture of universal grammar. *Language*, 70(1):72–131.

Christianson, K., Hollingworth, A., Halliwell, J. F., and Ferreira, F. (2001). Thematic roles assigned along the garden path linger. *Cognitive Psychology*, 42:368–407.

Connine, C. M., Blasko, D. G., and Hall, M. (1991). Effects of subsequent sentence context in auditory word recognition: Temporal and linguistic constraints. *Journal of Memory and Language*, 30(2):234–250.

Demberg, V. and Keller, F. (2008). Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, 109(2):193–210.

Ferreira, F. and Henderson, J. M. (1993). Reading processes during syntactic analysis and reanalysis. *Canadian Journal of Experimental Psychology*, 16:555–568.

Fodor, J. D. (2002). Psycholinguistics cannot escape prosody. In *Proceedings of the Speech Prosody Conference*.

Frank, S. L. (2009). Surprisal-based comparison between a symbolic and a connectionist model of sentence processing. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, pages 1139–1144.

Frazier, L. (1979). *On Comprehending Sentences: Syntactic Parsing Strategies*. PhD thesis, University of Massachusetts.

Frazier, L. and Rayner, K. (1982). Making and correcting errors during sentence comprehension: Eye movements in the analysis of structurally ambiguous sentences. *Cognitive Psychology*, 14:178–210.

Goodman, J. (1999). Semiring parsing. *Computational Linguistics*, 25(4):573–605.

Hale, J. (2001). A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 159–166.

Hale, J. (2006). Uncertainty about the rest of the sentence. *Cognitive Science*, 30(4):609–642.

Hill, R. L. and Murray, W. S. (2000). Commas and spaces: Effects of punctuation on eye movements and sentence parsing. In Kennedy, A., Radach, R., Heller, D., and Pynte, J., editors, *Reading as a Perceptual Process*. Elsevier.

Jurafsky, D. (1996). A probabilistic model of lexical and syntactic access and disambiguation. *Cognitive Science*, 20(2):137–194.

Kučera, H. and Francis, W. N. (1967). *Computational Analysis of Present-day American English*. Providence, RI: Brown University Press.

Lang, B. (1988). Parsing incomplete sentences. In *Proceedings of COLING*.

Levy, R. (2008a). Expectation-based syntactic comprehension. *Cognition*, 106:1126–1177.

Levy, R. (2008b). A noisy-channel model of rational human sentence comprehension under uncertain input. In *Proceedings of the 13th Conference on Empirical Methods in Natural Language Processing*, pages 234–243.

Levy, R. and Andrew, G. (2006). Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of the 2006 conference on Language Resources and Evaluation*.

Levy, R., Bicknell, K., Slattery, T., and Rayner, K. (2009). Eye movement evidence that readers maintain and act on uncertainty about past linguistic input. *Proceedings of the National Academy of Sciences*, 106(50):21086–21090.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1994). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Mitchell, D. C. (1984). An evaluation of subject-paced reading tasks and other methods for investigating immediate processes in reading. In Kieras, D. and Just, M. A., editors, *New methods in reading comprehension*. Hillsdale, NJ: Earlbaum.

Mitchell, D. C. (1987). Lexical guidance in human parsing: Locus and processing characteristics. In Coltheart, M., editor, *Attention and Performance XII: The psychology of reading*. London: Erlbaum.

Narayanan, S. and Jurafsky, D. (1998). Bayesian models of human sentence processing. In *Proceedings of the Twelfth Annual Meeting of the Cognitive Science Society*.

Narayanan, S. and Jurafsky, D. (2002). A Bayesian model predicts human parse preference and reading time in sentence processing. In *Advances in Neural Information Processing Systems*, volume 14, pages 59–65.

Nederhof, M.-J. and Satta, G. (2003). Probabilistic parsing as intersection. In *Proceedings of the International Workshop on Parsing Technologies*.

Nederhof, M.-J. and Satta, G. (2008). Computing partition functions of PCFGs. *Research on Logic and Computation*, 6:139–162.

Roark, B., Bachrach, A., Cardenas, C., and Pallier, C. (2009). Deriving lexical and syntactic expectation-based measures for psycholinguistic modeling via incremental top-down parsing. In *Proceedings of EMNLP*.

Rohde, D. (2005). *TGrep2 User Manual*, version 1.15 edition.

Smith, N. A. and Johnson, M. (2007). Weighted and probabilistic context-free grammars are equally expressive. *Computational Linguistics*, 33(4):477–491.

Smith, N. J. and Levy, R. (2008). Optimal processing times in reading: a formal model and empirical investigation. In *Proceedings of the 30th Annual Meeting of the Cognitive Science Society*.

Staub, A. (2007). The parser doesn't ignore intransitivity, after all. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, 33(3):550–569.

Stolcke, A. (1995). An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201.

Sturt, P., Pickering, M. J., and Crocker, M. W. (1999). Structural change and reanalysis difficulty in language comprehension. *Journal of Memory and Language*, 40:136–150.

Tabor, W. and Hutchins, S. (2004). Evidence for self-organized sentence processing: Digging in effects. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, 30(2):431–450.

van Gompel, R. P. G. and Pickering, M. J. (2001). Lexical guidance in sentence processing: A note on Adams, Clifton, and Mitchell (1998). *Psychonomic Bulletin & Review*, 8(4):851–857.

Metagrammar Engineering: Towards systematic exploration of implemented grammars

Antske Fokkens
Department of Computational Linguistics, Saarland University &
German Research Center for Artificial Intelligence (DFKI) Project Office Berlin
Alt-Moabit 91c, 10559 Berlin, Germany
afokkens@coli.uni-saarland.de

Abstract

When designing grammars of natural language, typically, more than one formal analysis can account for a given phenomenon. Moreover, because analyses interact, the choices made by the engineer influence the possibilities available in further grammar development. The order in which phenomena are treated may therefore have a major impact on the resulting grammar. This paper proposes to tackle this problem by using metagrammar development as a methodology for grammar engineering. I argue that metagrammar engineering as an approach facilitates the systematic exploration of grammars through comparison of competing analyses. The idea is illustrated through a comparative study of auxiliary structures in HPSG-based grammars for German and Dutch. Auxiliaries form a central phenomenon of German and Dutch and are likely to influence many components of the grammar. This study shows that a special auxiliary+verb construction significantly improves efficiency compared to the standard argument-composition analysis for both parsing and generation.

1 Introduction

One of the challenges in designing grammars of natural language is that, typically, more than one formal analysis can account for a given phenomenon. The criteria for choosing between competing analyses are fairly clear (observational adequacy, analytical clarity, efficiency), but given that analyses of different phenomena interact, actually evaluating analyses on those criteria in a systematic manner is far from straightforward. The standard methodology involves either picking one analysis, and seeing how it goes, then backing out if it does not work out, or laboriously adapting a grammar to two versions supporting different analyses (Bender, 2010). The former approach is not in any way systematic, increasing the risk that the grammar is far from optimal in terms of efficiency. The latter approach potentially causes the grammar engineer an amount of work that will not scale for considering many different phenomena.

This paper proposes a more systematic and tractable alternative to grammar development: metagrammar engineering. I use "metagrammar" as a generic term to refer to a system that can generate implemented grammars. The key idea is that the grammar engineer adds alternative plausable analyses for linguistic phenomena to a metagrammar. This metagrammar can generate all possible combinations of these analyses automatically, creating different versions of a grammar that cover the same phenomena. The engineer can test directly how competing analyses for different phenomena interact, and determine which combinations are possible (after minor adaptations) and which analyses are incompatible.

The idea of metagrammar engineering is illustrated here through a case study of word order and auxiliaries in Germanic languages, which forms the second goal of this paper. Auxiliaries form a central phenomenon of German and Dutch and are likely to influence many components of the grammar. The results show that the analysis of auxiliary+verb structures presented in Bender (2010) significantly im-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1066–1076,
Portland, Oregon, June 19-24, 2011.

proves efficiency of the grammar compared to the standard argument-composition analysis within the range of phenomena studied. Because future research is needed to determine whether the auxiliary+verb alternative can interact properly with additional phenomena and still lead to more efficient results than argument-composition, it is particularly useful to have a grammar generator that can automatically create grammars with either of the two analyses.

The remainder of this paper starts with the case study. Section 2 provides a description of the context of the study. The relevant linguistic properties and alternative analyses are described in Sections 3 and 4. After evaluating and discussing the case study's results, I return to the general approach of metagrammar engineering. Section 6 presents related work on metagrammars. It is followed by a conclusion and discussion on using metagrammars as a methodology for grammar engineering.

2 A metagrammar for Germanic Languages

2.1 The LinGO Grammar Matrix

The LinGO Grammar Matrix (Bender et al., 2002; Bender et al., 2010) provides the main context for the experiments described in this paper. To begin with, its further development plays a significant role for the motivation of the present study. More importantly, the Germanic metagrammar is implemented as a special branch of the LinGO Grammar Matrix and uses a significant amount of its code.

The Grammar Matrix customization system allows users to derive a starter grammar for a particular language from a common multi-lingual resource by specifying linguistic properties through a web-based questionnaire. The grammars are intended for parsing and generation with the LKB (Copestake, 2002) using Minimal Recursion Semantics (Copestake et al., 2005, MRS) as parsing output and generation input. After the starter grammar has been created, its development continues independently: engineers can thus make modifications to their grammar without affecting the multi-lingual resource.

Internally, the customization system works as follows: The web-based questionnaire registers linguistic properties in a file called "choices" (hence-forth choices file). The customization system takes this choices file as input to create grammar fragments, using so-called "libraries" that contain implementations of cross-linguistically variable phenomena. Depending on the definitions provided in the choices file, different analyses are retrieved from the customization system's libraries. The language specific implementations inherit from a core grammar which handles basic phrase types, semantic compositionality and general infrastructure, such as feature geometry (Bender et al., 2002).

The present study is part of a larger effort to improve the customization library for auxiliary structures in free word order and verb second languages. It examines whether Bender's observations concerning an improved analysis for auxiliaries in Wambaya (Bender, 2010) also hold for Germanic languages. A more elaborate study of German and Dutch (including both Flemish and (Northern) Dutch, which have slightly different word order constraints) is informative, because these languages are well-described and known to have distinctly challenging word order behavior.

2.2 Germanic branch

In order to create grammars for Germanic languages, a specialized branch of the Grammar Matrix customization system was developed. This Germanic grammars generator uses the Grammar Matrix's facilities to generate types in type description language (tdl). At present, the generator uses the Grammar Matrix analyses for agreement and case marking as well as basics from its morphotactics, coordination and lexicon implementations.

In the first stage, the word order library and auxiliary implementation were extended to cover two alternative analyses for Germanic word order (see Section 4). The coordination library was adapted to ensure correct interactions with the new word order analyses and agreement. The morphotactics library was extended to cover Dutch and Flemish interactions between word order and morphology. Finally, the lexicon and verbal case pattern implementations were extended to cover ditransitive verbs.

Both versions of word order analyses can be tweaked to include or exclude a rarely occurring variant of partial VP fronting (see Section 4.3) resulting in four distinct grammars for each of the

Vorfeld	**LB**	Mittelfeld	**RB**	Nachfeld
Der Mann	**hat**	den Jungen	**gesehen**	nach der Party
The man.nom	*has*	*the boy.acc*	*seen*	*after the party*
Der Mann	**hat**	den Jungen nach der Party	**gesehen**	
Den Jungen	**hat**	der Mann	**gesehen**	nach der Party
Nach der Party	**hat**	der Mann den Jungen	**gesehen**	
Den Jungen **gesehen**	**hat**	der Mann nach der Party		
Gesehen	**hat**	der Mann den Jungen nach der Party		
The man saw the boy after the party				

Table 1: Basic structure of German word order (not exhaustive)

languages under investigation. These 12 grammars cover Dutch, Flemish and German main clauses with up to three core arguments.[1]

3 Germanic word order

3.1 German word order

Topological fields (Erdmann, 1886; Drach, 1937) form the easiest way to describe German word order. The sentence structure for declarative main clauses, consists of five topological fields: Vorfeld ("pre-field"), Left Bracket (LB), Mittelfeld ("middle field"), Right Bracket (RB) and the Nachfeld ("after field"). A subset of permissible alternations in German are provided in Table 1. The last two sentences present an example of partial VP fronting.

The fields are defined with regard to verbal forms, which are placed in the Left and Right Brackets. Each topological field has word order restrictions of its own. The Vorfeld must contain exactly one constituent in an affirmative main clause. The Left Bracket contains the finite verb and no other elements. Other verbal forms (if not fronted to the Vorfeld) must be placed in the Right Bracket. Most non-verbal elements are placed in the Mittelfeld. When main verbs are placed in the Vorfeld, their object(s) may stay in the Mittelfeld. This kind of partial VP fronting is illustrated by the last example in Table 1. The Nachfeld typically contains subordinate clauses and sometimes adverbial phrases.

In German, the respective order between the verbs in the Right Bracket is head-final, i.e. auxiliaries follow their complements. The only exception is the auxiliary flip: under certain conditions in subordinate clauses, the finite verb precedes all other verbal forms.

3.2 Dutch word order

Dutch word order reveals the same topological fields as German. There are two main differences between the languages where word order is concerned. First, whereas the order of arguments in the German Mittelfeld allows some flexibility depending on information structure, Dutch argument order is fixed, except for the possibility of placing any argument in the Vorfeld. A related aspect is that Dutch is less flexible as to what partial VPs can be placed in the Vorfeld.

The second difference is the word order in the Right Bracket. The order of auxiliaries and their complements is less rigid in Dutch and typically auxiliary-complement, the inverse of German order. Most Dutch auxiliaries can occur in both orders, but this may be restricted according to their verb form. Four groups of auxiliary verbs can be distinguished that have different syntactic restrictions.

1. Verbs selecting for participles which may appear on either side of their complement (e.g. *hebben* ("have"), *zijn* ("be")).

2. Verbs selecting for participles which prefer to follow their complement and must do so if they are in participle form themselves (e.g. *blijven* ("remain"), *krijgen* ("get")).

3. Modals selecting for infinitives which prefer to precede their complement and must do so if they appear in infinitive form themselves.

[1]The grammar generation system also creates Danish grammars. Danish results are not presented, because the language does not pose the challenges explained in Section 4.

VF	LB	MF	RB
De man	zou	haar	kunnen hebben gezien
the man	*would*	*her.acc*	*can have seen*
De man	zou	haar	gezien kunnen hebben
%De man	zou	haar	kunnen gezien hebben
The man should have been able to see her			

Table 2: Variations of Dutch auxiliary order

4. Verbs selecting for "to infinitives" which must precede their complement.

While there is some variation among speakers, the generalizations above are robust. The permitted variations assuming a verb of the 3rd and 1st category in the right bracket are presented in Table 2.[2]

The variant %*De man zou haar kunnen gezien hebben* is typical of speakers from Belgium (Haeseryn, 1997); speakers from the Netherlands tend to regard such structures as ungrammatical. Our system can both generate a Flemish grammar accepting all of the above and a (Northern) Dutch grammar, rejecting the third variant.

4 Alternative auxiliary approaches

This section presents the alternative analyses for auxiliary-verb structures in Germanic languages compared in this study. For reasons of space, I limit my description to an explanation of the differences and relevance of the compared analyses.[3]

4.1 Argument-composition

The standard analysis for German and Dutch auxiliaries in HPSG is a so-called "argument-composition" analysis (Hinrichs and Nakazawa, 1994), which I will explain through the following Dutch example:[4]

(1) Ik zou het boek willen lezen.
I would the book want read.
"I would like to read the book."

In the sentence above, the auxiliary *willen* "want" separates the verb *lezen* "read" from its object *het boek* "the book". A parser respecting surface order can thus not combine *lezen* and *het boek* before combining *willen* and *lezen*.

Figure 1: Standard Auxiliary Subcategorization

The argument-composition analysis was introduced to make sure that *het boek* can be picked up as the object of the embedded verb *lezen*. The subcategorization of an auxiliary under this analysis is presented in Figure 1. The subject of the auxiliary is identical to the subject of the auxiliary's complement. Its complement list consists of the concatenation of the verbal complement and any complement this verbal complement may select for. In the sentence above, *willen* will add the subject and the object of *lezen* to its own subcatorization lists.[5] This standard solution for auxiliary-verb structures is (with minor differences) also what is provided by the Matrix customization system.

Argument-composition can capture the grammatical behavior of auxiliaries in German and Dutch. However, grammaticality and coverage is not all that matters for grammars of natural language. Efficiency remains an important factor, and argument-composition has some undesirable properties on this level. The problem lies in the fact that lexical entries of auxiliaries have underspecified elements on their subcategorization lists. With the current chart parsing and chart generation algorithms (Carroll and Oepen, 2005), an auxiliary in a language with flexible word order will speculatively add edges to the chart for potential analyses with the adjacent constituent as subject or complement. Because the length of the lists are underspecified as well, it can continue wrongly combining with all elements in the string. In the worse case scenario, the number of edges created by an auxiliary grows exponentially in the number of words and constituents in the string. The efficiency problem is even worse for generation: while the parser is restricted by the surface order of

[2] Note that the same orders as in the Right Brackets may also occur in the Vorfeld (with or without the object).

[3] Details of the implementations can be found by using the metagrammar, which can be found on my homepage.

[4] Hinrichs and Nakazawa (1994) present an analysis for the German auxiliary flip. The relevant observations are the same.

[5] In the semantic representation, both arguments will be directly related to the main verb exclusively.

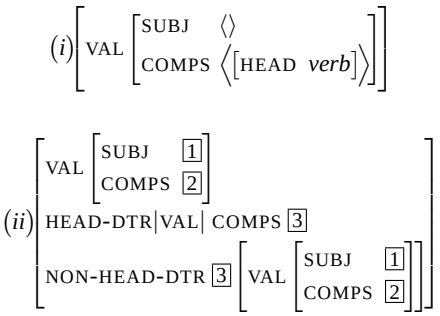


Figure 2: Auxiliary lexical type (*i*) and Auxiliary+verb construction (*ii*) under alternative analysis

the string, the generator will attempt to combine all lexical items suggested by the input semantics, as well as lexical items with empty semantics, in random order.

4.2 Aux+verb construction

Bender (Bender, 2010)[6] presents an alternative approach to auxiliary-verb structures for the Australian language Wambaya. The analysis introduces auxiliaries that only subcategorize for one verbal complement, not raising any of the complement's arguments or its subject. Auxiliaries combine with their complement using a special auxiliary+verb rule. Figure 2 presents this alternative solution. In principle, the new analysis uses the same technique as argument composition. The difference is that the auxiliary now starts out with only one element in its subcategorization lists and can only combine with potential verbal complements that are appropriately constrained. The structure that combines the auxiliary with its complement places the remaining elements on the complement's SUBJ and COMPS lists on the respective lists of the newly formed phrase, as can be seen in Figure 2 (*ii*). The constraints on raised arguments are known when the construction applies. The efficiency problem sketched above is thus avoided.

4.3 A small wrinkle: partial VP fronting

In its basic form, the auxiliary+verb structure cannot handle partial VP fronting where the main verb is placed in first position leaving one or more verbal forms in the verbal cluster, as illustrated in (2) for Dutch:

(2) Gezien zou de man haar kunnen hebben.
Seen should the man her can have
"The man should have been able to **see** her."

The problem is that *hebben* "have" cannot combine with *gezien* "seen", because they are separated by the head of the clause. Because the verb *hebben* cannot combine with its complement, it cannot raise its complement's arguments either: the auxiliary+verb analysis only permits raising when auxiliary and complement combine.

This shortcoming is no reason to immediately dismiss the proposal. Structures such as (2) are extremely rare. The difference in coverage of a parser that can and a parser that cannot handle such structures is likely to be tiny, if present at all, nor is it vital for a sentence generator to be able to produce them. However, a correct grammar should be able to analyze and produce all grammatical structures.

I implemented an additional version of the auxiliary+verb construction using two rather complex rules that capture examples such as (2). Because the structure in (2) also presented difficulties for the argument-composition analysis in Dutch, I tested both of the analyses with and without the inclusion of these structures. In the ideal case, the full coverage version will remain efficient enough as the grammar grows. But if this turns out not to be the case, the decision can be made to exclude the additional rule from the grammar or to use it as a robustness rule that is only called when regular rules fail. Given the metagrammar engineering approach, it will be straightforward to decide at a later point to exclude the special rule, if corpus studies reveal this is favourable.

5 Grammars and evaluation

5.1 Experimental set-up

As described above, the Germanic metagrammar is a branch of the customization system. As such, it takes a choices file as input to create a grammar. The basic choices files for Dutch and German were created through the LinGO Grammar Matrix web inter-

[6] Bender credits the key idea behind this analysis to Dan Flickinger (Bender, 2010).

	Complete Set		Reduced Set		
	Positive	Total	Positive	Total	Av.
	s	s	s	s	w/s
Du	177	14654	138	14591	6.61
Fl	195	14654	156	14606	6.61
Ge	116	6926	84	6914	6.65

Table 3: Number of test examples (s) used in evaluation and average words per sentence (w/s)

face.[7] The choices files defined artificial grammars with a dummy vocabulary. The system can produce real fragments of the languages, but strings representing syntactic properties through dummy vocabulary were used to give better control over ambiguity facilitating the evaluation of coverage and overgeneration of the grammars. The grammars have a lexicon of 9-10 unambiguous dummy words.

The created choices files were extended offline to define those properties that the Germanic metagrammar captures, but are not incorporated in the Matrix customization system. This included word order of the auxiliary and complement, fixed or free argument order, influence of inflection on word order, a more elaborate case hierarchy, ditransitive verbs, and the choice of auxiliary/verb analysis. Four choices files with different combinations of analyses were created for each language, resulting in 12 choices files in total.

A basic test suite was developed that covers intransitive, transitive and ditransitive main clauses with up to three auxiliaries. The German set was based on a description provided by Kathol (2000), Dutch and Flemish were based on Haeseryn (1997). For each verb and auxiliary combination, all permissible word orders were defined based on descriptive resources. In order to make sure the grammars do not reveal unexpected forms of overgeneration, all possible ungrammatical orders were automatically generated. Table 3 provides the sizes of the test suites. Each language has both a complete set for the 6 grammars that provide full coverage, and a reduced set for the 6 grammars that can not handle split verbal clusters (see Section 4.3 for the motivation to test grammars that do not have full coverage).

[7] `http://www.delph-in.net/matrix/customize/`

Each grammar was created using the metagrammar, ensuring that all components except the competing analyses were held constant among compared grammars. The [incr tsdb()] competence and performance profiling environment (Oepen, 2001) was used in combination with the LKB to evaluate parsing performance of the individual grammars on the test suites. For each grammar, the number of required parsing tasks, memory (space) and CPU time per sentence, as well as the number of passive edges created during an average parse were compared. Performance on language generation was evaluated using the LKB.

5.2 Parsing results

Table 4 presents the results from the parsing experiment. Note that all directly compared grammars have the same empirical coverage (100% coverage and 0% overgeneration on the phenomena included in the test suites). The comparison therefore addresses the effect on efficiency of the alternative analyses. Three tests per grammar were carried out: one on positive data, one on negative data and one on the complete dataset. Results were similar for all three sets, with slightly larger differences in efficiency for negative examples. For reasons of space, only the results on positive examples are presented, which are more relevant for most applications involving parsing. The results show that the auxiliary+verb (aux+v) leads to a more efficient grammar according to all measures used. There is an average reduction of 73.2% in performed tasks, 56.3% in produced passive edges and 32.9% in memory when parsing grammatical examples using the auxiliary+verb structure compared to argument-composition. CPU-time per sentence also improved significantly, but, due to the short average sentence length (5-10 words) the value is too small for exact comparison with [incr tsdb()].

5.3 Sentence generation evaluation

The complete coverage versions of Dutch and German were used to create the exhaustive set of sentences with an intransitive, transitive and ditransitive verb combined with none, one or two auxiliaries but rapidly loses ground when one or more auxiliaries[8]

[8] All auxiliaries in the grammars contribute an ep.

Average Performed Tasks				
	Compl. Cov. Gram.		No Split Cl. Gram.	
	arg-comp	**aux+v**	**arg-comp**	**aux+v**
Du	524	149	480	134
Fl	529	150	483	137
Ge	684	148	486	136
Average Created Edges				
	Compl. Cov. Gram.		No Split Cl. Gram.	
	arg-comp	**aux+v**	**arg-comp**	**aux+v**
Du	58	25	52	25
Fl	58	26	52	25
Ge	67	23	52	24
Average Memory Use (kb)				
	Compl. Cov. Gram.		No Split Cl. Gram.	
	arg-comp	**aux+v**	**arg-comp**	**aux+v**
Du	9691	6692	8944	6455
Fl	9716	6717	8989	6504
Ge	10289	5675	8315	5468
Average CPU Time (s)				
	Compl. Cov. Gram.		No Split Cl. Gram.	
	arg-comp	**aux+v**	**arg-comp**	**aux+v**
Du	0.04	0.02	0.03	0.01
Fl	0.04	0.02	0.03	0.01
Ge	0.06	0.01	0.04	0.01

Table 4: Parsing results positive examples

from a total of 18 MRSs. The input MRSs were obtained by parsing a sentence with canonical word order. Both versions provide the same set of sentences as output, confirming their identical empirical coverage. Table 5 presents the number of edges required by the generator to produce the full set of generated sentences from a given MRS. The cells with no number represent conditions under which the LKB generator reaches the maximum limit of edges, set at 40,000, without completing its exhaustive search.

The grammar using argument-composition is slightly more efficient when there are no auxiliaries, are added, in particular when sentence length increases: For ditransitive verbs (dv), the Dutch argument-composition grammar maxes out the 40,000 edge limit with two auxiliaries, whereas the auxiliary+verb grammar creates 910 edges, a manageable number. Due to the more liberal order of arguments, results are even worse for German: the argument-composition grammar reaches its limit with the first auxiliary for ditransitive verbs. These results indicate that the auxiliary+verb analysis is

	Required edges					
Du	No Aux		1 Aux		2 Aux	
	arg-c	**aux+v**	**arg-c**	**aux+v**	**arg-c**	**aux+v**
iv	54	57	221	99	792	248
tv	124	141	1311	211	7455	500
dv	212	230	14968	378	–	910
Ge	No Aux		1 Aux		2 Aux	
	arg-c	**aux+v**	**arg-c**	**aux+v**	**arg-c**	**aux+v**
iv	54	57	295	84	1082	165
tv	130	142	4001	212	18473	422
dv	306	351	–	608	–	1379

Table 5: Performance on Sentence Generation

strongly preferable where natural language generation is concerned.

5.4 In summary

The results of the experiment presented above show that avoiding underspecified subcategorization lists, as found in the standard argument-composition analysis, significantly increases the efficiency of the grammar for both parsing and generation. On average, they show a reduction of 73.2% in performed tasks, 56.3% in produced passive edges and 32.9% in memory for parsing. In generation experiments, results are even more impressive: the reduction of edges for German sentences with one auxiliary and a ditransitve verb is at least 98.5%. These results show that the auxiliary+verb alternative should be considered seriously as an alternative to the HPSG standard analysis of argument-composition, though further investigation in a larger context is needed before final conclusions can be drawn.

Future work will focus on increasing the coverage of the grammars, as well as the number of alternative options explored. In particular, both approaches for auxiliaries should be compared using alternative analyses for verb-second word order found in other HPSG-based grammars, such as the GG (Müller and Kasper, 2000; Crysmann, 2005), Grammix (Müller, 2009; Müller, 2008) and Cheetah (Cramer and Zhang, 2009) for German, and Alpino (Bouma et al., 2001) for Dutch. These grammars may use approaches that somewhat reduce the problem of argument-composition, leading to less significant differences between the auxiliary+verb and argument-composition analyses. On the other hand, planned extensions that cover modification and sub-

ordinate clauses will increase local ambiguities. The advantage of the auxiliary+verb analysis is likely to become more important as a result.

In addition to providing a clearer picture of auxiliary structures, these extensions will also lead to a better insight into efforts involved in using grammar generation to explore alternative versions of a grammar over time. In particular, it should provide an indication of the feasibility of maintaining a higher number of competing analyses as the grammar grows. After providing background on related metagrammar projects and their goals, I will elaborate on the importance of systematic exploration of grammars in the discussion.

6 Related work

Metagrammars (or grammar generators) have been established in the field for over a decade. This section provides an overview of the goals and set-up of some of the most notable projects.

The MetaGrammar project (Candito, 1998; de la Clergerie, 2005; Kinyon et al., 2006) started as an effort to encode syntactic knowledge in an abstract class hierarchy. The hierarchy can contain cross-linguistically invariable properties and syntactic properties that hold across frameworks (Kinyon et al., 2006). The factorized descriptions of MetaGrammar support Tree-Adjoining Grammars (Joshi et al., 1975, TAG) as well as Lexical Functional Grammars (Bresnan, 2001, LFG). The eXtensible MetaGrammar (Crabbé, 2005, XMG) defines its MetaGrammar as classes that are part of a multiple inheritance hierarchy. Kinyon et al. (Kinyon et al., 2006) use XMG to perform a cross-linguistic comparison of verb-second structures. Their study focuses on code-sharing between the languages, but does not address the problem of competing analyses investigated in this paper.

The GF Resource Grammar Library (Ranta, 2009) is a multi-lingual linguistic resource that contains a set of syntactic analyses implemented in GF (Grammatical Framework). The purpose of the library is to allow engineers working on NLP applications to write simple grammar rules that can call more complex syntactic implementations from the grammar library. The grammar library is written by researchers with linguistic expertise. It makes extensive use of code sharing: general categories and constructions that are used by all languages are implemented in a core syntax grammar. Each language[9] has its own lexicon and morphology, as well as a set of language specific syntactic structures. Code sharing also takes place between the subset of languages explored, in particular by means of common modules for Romance languages and for Scandanavian languages.

PAWS creates PC-PATR (McConnel, 1995) grammars based on field linguists' input. The main purpose of PAWS lies in descriptive grammar writing and "computer-assisted related language adaptation", where the grammar is used to map words from a text in a source language to a target language. PAWS differs from the other projects discussed here, because grammar engineering or syntactic research are not the main focus of the project.

The LinGO Grammar Matrix, described in Section 2.1, is most closely related to the work presented in this paper. Like the other projects reviewed here, the Grammar Matrix does not offer alternative analyses for the same phenomenon. Moreover, starter grammars created by the Grammar Matrix are developed manually and individually after their creation. The approach taken in this paper differs from the original goal of the Grammar Matrix in that it continues the development of new grammars within the system, introducing a novel application for metagrammars. By using a metagrammar to store alternative analyses, grammars can be explored systematically over time. As such, the paper introduces a novel methodology for grammar engineering. The discussion and conclusion will elaborate on the advantages of the approach.

7 Discussion and conclusion

7.1 The challenge of choosing the right analysis

As mentioned in the introduction, most phenomena in natural languages can be accounted for by more than one formal analysis. An engineer may implement alternative solutions and test the impact on the grammar concerning interaction with other phenomena (Bierwisch, 1963; Müller, 1999; Bender, 2008; Bender et al., 2011) and efficiency to decide between analyses.

[9] Ranta (Ranta, 2009) reports that GF is developed for fourteen languages, and more are under development.

However, it is not feasible to carry out comparative tests by manually creating different versions of a grammar every time a decision about an implementation is made. Moreover, even if such a study were carried out at each stage, only the interaction with the current state of the grammar would be tested. This has two undesirable consequences. First, options may be rejected that would have worked perfectly well if different decisions had been made in the past. Second, because each decision is only based on the current state of the grammar, the resulting grammar is partially (or even largely) a product of the order in which phenomena are treated.[10]

For grammar engineers with practical applications in mind, this is undesirable because the resulting grammar may end up far from optimal. For grammar writers that use engineering to find valid linguistic analyses, the problem is even more serious: if there is a truth in a declarative grammar, surely, this should not depend on the order in which phenomena are treated.

7.2 Metagrammar engineering

This paper proposes to systematically explore analyses throughout the development of a grammar by writing a metagrammar (or grammar generator), rather than directly implementing the grammar. A metagrammar can contain several different analyses for the same phenomenon. After adding a new phenomenon to the metagrammar, the engineer can automatically generate versions of the grammar containing different combinations of previous analyses. As a result, the engineer can not only systematically explore how alternative analyses interact with the current grammar, but also continue to explore interactions with phenomena added in the future. Especially for alternative approaches to basic properties of the language, such as the auxiliary-verb structures examined in this study, parallel analyses may prevent the cumbersome scenario of changing a deeply embedded property of a large grammar.

An additional advantage is that the engineer can use the methodology to make different versions of the grammar depending on its intended application. For instance, it is possible to develop a highly restricted version for grammar checking that provides detailed feedback on detected errors (Bender et al., 2004), next to a version with fewer constraints to parse open text.

As far as finding optimal solutions is concerned, it must be noted that this approach does not guarantee a perfect result, partially because there is no guarantee the grammar engineer will think of the perfect solution for each phenomenon, but mainly because it is not maintainable to implement all possible alternatives for each phenomenon and make them interact correctly with all other variations in the grammar. The grammar engineer still needs to decide which alternatives are the most promising and therefore the most important to implement and maintain. The resulting grammar therefore partially remains a result of the order in which phenomena are implemented. Nevertheless, the grammar engineer can keep and try out solutions in parallel for a longer time, increasing the possibility of exploring more alternative versions of the grammar. These additional investigations allow for better informed decisions to stop exploring certain analyses. In addition, by breaking up analyses into possible alternatives, chances are that the resulting metagrammar will be more modular than a directly written grammar would have been, which facilitates exploring alternatives further.

In sum, even though metagrammar engineering does not completely solve the challenge of complete explorations of a grammar's possibilities, it does facilitate this process so that finding optimal solutions becomes more likely, leading to better supported choices among alternatives and a more scientific approach to grammar development.

Acknowledgments.

The work described in this paper has been supported by the project TAKE (Technologies for Advanced Knowledge Extraction), funded under contract 01IW08003 by the German Federal Ministry of Education and Research. Emily M. Bender, Laurie Poulson, Christoph Zwirello, Bart Cramer, Kim Gerdes and three anonymous reviewers provided valuable feedback that resulted in significant improvement of the paper. Naturally, all remaining errors are my own.

[10] It is, of course, possible to go back and change old analyses based on new evidence. In practice, the large effort involved will only be undertaken if the advantages are apparent beforehand.

References

Emily M. Bender, Dan Flickinger, and Stephan Oepen. 2002. The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In John Carroll, Nelleke Oostdijk, and Richard Sutcliffe, editors, *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pages 8–14, Taipei, Taiwan.

Emily M. Bender, Dan Flickinger, Stephan Oepen, Annemarie Walsh, and Tim Baldwin. 2004. Arboretum: Using a precision grammar for grammar checking in call. In *Proceedings of the InSTIL/ICAL Symposium: NLP and Speech Technologies in Advance Language Learning Systems*, Venice, Italy.

Emily M. Bender, Scott Drellishak, Antske Fokkens, Laurie Poulson, and Safiyyah Saleem. 2010. Grammar customization. *Research on Language & Computation*, 8(1):23–72.

Emily M. Bender, Dan Flickinger, and Stephan Oepen. 2011. Grammar engineering and linguistic hypothesis testing. In Emily M. Bender and Jennifer E. Arnold, editors, *Language from a Cognitive Perspective: Grammar, Usage and Processing*, pages 5–29. Stanford: CSLI Publications, Palo Alto, USA.

Emily M. Bender. 2008. Grammar engineering for linguistic hypothesis testing. In Nicholas Gaylord, Alexis Palmer, and Elias Ponvert, editors, *Proceedings of the Texas Linguistics Society X Conference: Computational Linguistics for Less-Studied Languages*, pages 16–36, Stanford. CSLI Publications.

Emily M. Bender. 2010. Reweaving a grammar for Wambaya: A case study in grammar engineering for linguistic hypothesis testing. *Linguistic Issues in Language Technology*, 3(3):1–34.

Manfred Bierwisch. 1963. *Grammatik des deutschen Verbs*, volume II of *Studia Grammatica*. Akademie Verlag.

Gosse Bouma, Gertjan van Noord, and Robert Malouf. 2001. Alpino: Wide coverage computational analysis of Dutch. In *Computational Linguistics in the Netherlands CLIN 2000*.

Joan Bresnan. 2001. *Lexical Functional Syntax*. Blackwell Publishers, Oxford.

Marie-Helene Candito. 1998. Building parallel LTAG for French and Italian. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 211–217, Montreal, Quebec, Canada. Association for Computational Linguistics.

John Carroll and Stephan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In *IJCNLP*, Jeju Island. Springer-Verlag LNCS.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan Sag. 2005. Minimal recursion semantics. an introduction. *Journal of Research on Language and Computation*, 3(2–3):281 – 332.

Ann Copestake. 2002. *Implementing Typed Feature Structure Grammars*. CSLI Publications, Stanford, CA.

Benoît Crabbé. 2005. *Représentation modulaire et paramétrable de grammaires électroniques lexicalisées*. Ph.D. thesis, Université de Paris 7.

Bart Cramer and Yi Zhang. 2009. Constructon of a German HPSG grammar from a detailed treebank. In *Proceedings of the ACL 2009 Grammar Engineering across Frameworks workshop*, pages 37–45, Singapore, Singapore.

Berthold Crysmann. 2005. Relative clause extraposition in German: An efficient and portable implementation. *Research on Language and Computation*, 3(1):61–82.

Éric Villemonte de la Clergerie. 2005. From metagrammars to factorized TAG/TIG parsers. In *Proceedings of IWPT'05*, pages 190–191.

Erich Drach. 1937. *Grundgedanken der Deutschen Satzlehre*. Diesterweg, Frankfurt am Main, Germany.

Oskar Erdmann. 1886. *Grundzüge der deutschen Syntax nach ihrer geschichtlichen Entwicklung dargestellt. Erste Abteilung*. Verlag der Cotta'schen Buchhandlung, Stuttgart, Germany.

Walter Haeseryn. 1997. De gebruikswaarde van de ans voor tekstschrijvers, taaltrainers en taaladviseurs. *Tekst[blad]*, 3.

Erhard Hinrichs and Tsuneko Nakazawa. 1994. Linearizing auxs in German verbal complexes. In John Nerbonne, Klaus Netter, and Carl Pollard, editors, *German in HPSG*. CSLI, Stanford, USA.

Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1):136–163.

Andreas Kathol. 2000. *Linear Syntax*. Oxford Press.

Alexandra Kinyon, Owen Rambow, Tatjana Scheffler, SinWon Yoon, and Aravind K. Joshi. 2006. The metagrammar goes multilingual: A cross-linguistic look at the V2-phenomenon. In *Proceedings of the Eighth International Workshop on Tree Adjoining Grammar and Related Formalisms*, pages 17–24, Sydney, Australia. Association for Computational Linguistics.

Stephen McConnel. 1995. PC-PATR reference manual.

Stefan Müller and Walter Kasper. 2000. HPSG analysis for German. In Wolfgang Wahlster, editor, *Verbmobil: Foundations of Speech-to-Speech translation*, pages 238 – 253, Berlin, Germany. Springer.

Stefan Müller. 1999. *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche*. Max Niemeyer Verlag, Tübingen.

Stefan Müller. 2008. Depictive secondary predicates in german and english. In Christoph Schroeder, Gerd Hentschel, and Winfried Boeder, editors, *Secondary Predicates in Eastern European Languages and Beyond*, number 16 in Studia Slavica Oldenburgensia, pages 255–273, Oldenburg, Germany. BIS-Verlag.

Stefan Müller. 2009. On predication. In Stefan Müller, editor, *Proceedings of the 16th International Conference on Head-Driven Phrase Structure Grammar*, Stanford, USA. CSLI Publications.

Stephan Oepen. 2001. [incr tsdb()] — competence and performance laboratory. Technical report, DFKI, Saarbrücken, Germany.

Aarne Ranta. 2009. The GF resource grammar library. *Linguistic Issues in Language Technology*, 2(2).

Simple Unsupervised Grammar Induction from Raw Text with Cascaded Finite State Models

Elias Ponvert, Jason Baldridge and Katrin Erk
Department of Linguistics
The University of Texas at Austin
Austin, TX 78712
{ponvert,jbaldrid,katrin.erk}@mail.utexas.edu

Abstract

We consider a new subproblem of unsupervised parsing from raw text, unsupervised partial parsing—the unsupervised version of text chunking. We show that addressing this task directly, using probabilistic finite-state methods, produces better results than relying on the local predictions of a current best unsupervised parser, Seginer's (2007) CCL. These finite-state models are combined in a cascade to produce more general (full-sentence) constituent structures; doing so outperforms CCL by a wide margin in unlabeled PARSEVAL scores for English, German and Chinese. Finally, we address the use of phrasal punctuation as a heuristic indicator of phrasal boundaries, both in our system and in CCL.

1 Introduction

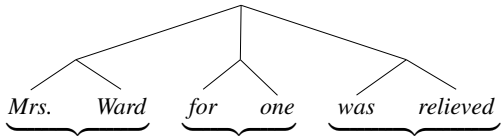
Unsupervised grammar induction has been an active area of research in computational linguistics for over twenty years (Lari and Young, 1990; Pereira and Schabes, 1992; Charniak, 1993). Recent work (Headden III et al., 2009; Cohen and Smith, 2009; Hänig, 2010; Spitkovsky et al., 2010) has largely built on the dependency model with valence of Klein and Manning (2004), and is characterized by its reliance on gold-standard part-of-speech (POS) annotations: the models are trained on and evaluated using sequences of POS tags rather than raw tokens. This is also true for models which are not successors of Klein and Manning (Bod, 2006; Hänig, 2010).

An exception which learns from raw text and makes no use of POS tags is the *common cover links* parser (CCL, Seginer 2007). CCL established state-of-the-art results for unsupervised *constituency* parsing from raw text, and it is also incremental and extremely fast for both learning and parsing. Unfortunately, CCL is a non-probabilistic algorithm based on a complex set of inter-relating heuristics and a non-standard (though interesting) representation of constituent trees. This makes it hard to extend. Note that although Reichart and Rappoport (2010) improve on Seginer's results, they do so by selecting training sets to best match the particular test sentences—CCL itself is used without modification.

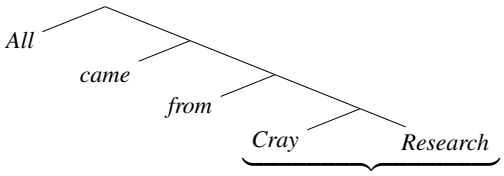
Ponvert et al. (2010) explore an alternative strategy of *unsupervised partial parsing*: directly predicting low-level constituents based solely on word co-occurrence frequencies. Essentially, this means segmenting raw text into multiword constituents. In that paper, we show—somewhat surprisingly—that CCL's performance is mostly dependent on its effectiveness at identifying low-level constituents. In fact, simply extracting non-hierarchical multiword constituents from CCL's output and putting a right-branching structure over them actually works *better* than CCL's own higher level predictions. This result suggests that improvements to low-level constituent prediction will ultimately lead to further gains in overall constituent parsing.

Here, we present such an improvement by using probabilistic finite-state models for phrasal segmentation from raw text. The task for these models is chunking, so we evaluate performance on identification of multiword chunks of all constituent types as well as only noun phrases. Our unsupervised chunkers extend straightforwardly to a cascade that predicts higher levels of constituent structure, similar to the supervised approach of Brants (1999). This forms an overall unsupervised parsing system that outperforms CCL by a wide margin.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1077–1086,
Portland, Oregon, June 19-24, 2011.



(a) Chunks: (*Mrs. Ward*), (*for one*), and (*was relieved*)



(b) Only one chunk extracted: (*Cray Research*)

Fig. 1: Examples of constituent chunks extracted from syntactic trees

2 Data

We use the standard data sets for unsupervised constituency parsing research: for **English**, the Wall Street Journal subset of the Penn Treebank-3 (WSJ, Marcus et al. 1999); for **German**, the Negra corpus v2 (Krenn et al., 1998); for **Chinese**, the Penn Chinese Treebank v5.0 (CTB, Palmer et al., 2006). We lower-case text but otherwise do not alter the raw text of the corpus. Sentence segmentation and tokenization from the treebank is used. As in previous work, punctuation is not used for evaluation.

In much unsupervised parsing work the test sentences are included in the training material. Like Cohen and Smith, Headden III et al., Spitkovsky et al., we depart from this experimental setup and keep the evaluation sets blind to the models during training. For English (WSJ) we use sections 00-22 for training, section 23 for test and we develop using section 24; for German (Negra) we use the first 18602 sentences for training, the last 1000 sentences for development and the penultimate 1000 sentences for testing; for Chinese (CTB) we adopt the data-split of Duan et al. (2007).

3 Tasks and Benchmark

Evaluation. By **unsupervised partial parsing**, or simply **unsupervised chunking**, we mean the segmentation of raw text into (non-overlapping) multiword constituents. The models are intended to capture local constituent structure – the lower branches of a constituent tree. For this reason we evaluate

WSJ	Chunks	203K	
	NPs	172K	
	Chnk ∩ NPs	161K	
Negra	Chunks	59K	
	NPs	33K	
	Chnk ∩ NPs	23K	
CTB	Chunks	92K	
	NPs	56K	
	Chnk ∩ NPs	43K	

Table 1: Constituent chunks and base NPs in the datasets.

		% constituents	% words
WSJ	Chunks	32.9	57.7
	NPs	27.9	53.1
Negra	Chunks	45.4	53.6
	NPs	25.5	42.4
CTB	Chunks	32.5	55.4
	NPs	19.8	42.9

Table 2: Percentage of gold standard constituents and words under constituent chunks and base NPs.

using what we call *constituent chunks*, the subset of gold standard constituents which are i) branching (multiword) but ii) non-hierarchical (do not contain subconstituents). We also evaluate our models based on their performance at identifying base noun phrases, NPs that do not contain nested NPs.

Examples of constituent chunks extracted from treebank constituent trees are in Fig. 1. In English newspaper text, constituent chunks largely correspond with base NPs, but this is less the case with Chinese and German. Moreover, the relationship between NPs and constituent chunks is not a subset relation: some base NPs do have internal constituent structure. The numbers of constituent chunks and NPs for the training datasets are in Table 1. The percentage of constituents in these datasets which fall under these definitions, and the percentage of words under these constituents, are in Table 2.

For parsing, the standard unsupervised parsing metric is unlabeled PARSEVAL. It measures precision and recall on constituents produced by a parser as compared to gold standard constituents.

CCL benchmark. We use Seginer's CCL as a benchmark for several reasons. First, there is a free/open-source implementation facilitating exper-

imental replication and comparison.[1] More importantly, until recently it was the only unsupervised raw text constituent parser to produce results competitive with systems which use gold POS tags (Klein and Manning, 2002; Klein and Manning, 2004; Bod, 2006) – and the recent improved raw-text parsing results of Reichart and Rappoport (2010) make direct use of CCL without modification. There are other raw-text parsing systems of note, EMILE (Adriaans et al., 2000), ABL (van Zaanen, 2000) and ADIOS (Solan et al., 2005); however, there is little consistent treebank-based evaluation of these models. One study by Cramer (2007) found that none of the three performs particularly well under treebank evaluation. Finally, CCL outperforms most published POS-based models when those models are trained on unsupervised word classes rather than gold POS tags. The only exception we are aware of is Hänig's (2010) unsuParse+, which outperforms CCL on Negra, though this is shown only for sentences with ten or fewer words.

Phrasal punctuation. Though punctuation is usually entirely ignored in unsupervised parsing research, Seginer (2007) departs from this in one key aspect: the use of *phrasal punctuation* – punctuation symbols that often mark phrasal boundaries within a sentence. These are used in two ways: i) they impose a hard constraint on constituent spans, in that no constituent (other than sentence root) may extend over a punctuation symbol, and ii) they contribute to the model, specifically in terms of the statistics of words seen adjacent to a phrasal boundary. We follow this convention and use the following set:

. ? ! ; , -- 。 、

The last two are ideographic full-stop and comma.[2]

4 Unsupervised partial parsing

We learn partial parsers as constrained sequence models over tags encoding local constituent structure (Ramshaw and Marcus, 1995). A simple tagset is unlabeled BIO, which is familiar from supervised chunking and named-entity recognition: the tag *B* denotes the beginning of a chunk, *I* denotes membership in a chunk and *O* denotes exclusion from any chunk. In addition we use the tag *STOP* for sentence boundaries and phrasal punctuation.

HMMs and PRLGs. The models we use for unsupervised partial parsing are hidden Markov models, and a generalization we refer to as probabilistic right linear grammars (PRLGs). An HMM models a sequence of observed states (words) $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ and a corresponding set of hidden states $\mathbf{y} = \{y_1, y_2, \ldots, y_N\}$. HMMs may be thought of as a special case of probabilistic context-free grammars, where the non-terminal symbols are the hidden state space, terminals are the observed states and rules are of the form NONTERM $\rightarrow$ TERM NONTERM (assuming y_1 and y_N are fixed and given). So, the emission and transition emanating from y_n would be characterized as a PCFG rule $y_n \rightarrow x_n\ y_{n+1}$. HMMs factor rule probabilities into emission and transition probabilities:

$$\begin{aligned} P(y_n \rightarrow x_n\ y_{n+1}) &= P(x_n, y_{n+1}|y_n) \\ &\approx P(x_n|y_n)\ P(y_{n+1}|y_n). \end{aligned}$$

However, without making this independence assumption, we can model right linear rules directly:

$$P(x_n, y_{n+1}|y_n) = P(x_n|y_n, y_{n+1})\ P(y_{n+1}|y_n).$$

So, when we condition emission probabilities on both the current state y_n and the next state y_{n+1}, we have an exact model. This direct modeling of the right linear grammar rule $y_n \rightarrow x_n\ y_{n+1}$ is what we call a probabilistic right-linear grammar. To be clear, a PRLG is just an HMM without the independence of emissions and transitions. See Smith and Johnson (2007) for a discussion, where they refer to PRLGs as Mealy HMMs.

We use expectation maximization to estimate model parameters. For the E step, the forward-backward algorithm (Rabiner, 1989) works identically for the HMM and PRLG. For the M step, we use maximum likelihood estimation with additive smoothing on the emissions probabilities. So, for the HMM and PRLG models respectively, for words

[1] http://www.seggu.net/ccl

[2] This set is essentially that of Seginer (2007). While it is clear from our analysis of CCL that it does make use of phrasal punctuation in Chinese, we are not certain whether ideographic comma is included.

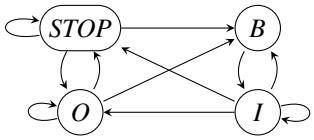


Fig. 2: Possible tag transitions as a state diagram.

	STOP	*B*	*I*	*O*
STOP	.33	.33		.33
B			1	
I	.25	.25	.25	.25
O	.33	.33		.33

Fig. 3: Uniform initialization of transition probabilities subject to the constraints in Fig. 2: rows correspond to antecedent state, columns to following state.

w and tags s, t:

$$\hat{P}(w|t) = \frac{C(t,w) + \lambda}{C(t) + \lambda V}$$
$$\hat{P}(w|s,t) = \frac{C(t,w,s) + \lambda}{C(t,s) + \lambda V}$$

where C are the soft counts of emissions $C(t,w)$, rules $C(t,w,s) = C(t \rightarrow w\ s)$, tags $C(t)$ and transitions $C(t,s)$ calculated during the E step; V is the number of terms w, and λ is a smoothing parameter. We fix $\lambda = .1$ for all experiments; more sophisticated smoothing could avoid dependence on λ.

We do not smooth transition probabilities (so $\hat{P}(s|t) = C(t,s)/C(t)$) for two reasons. First, with four tags, there is no data-sparsity concern with respect to transitions. Second, the nature of the task imposes certain constraints on transition probabilities: because we are only interested in multiword chunks, we expressly do not want to generate a *B* following a *B* – in other words $P(B|B) = 0$.

These constraints boil down to the observation that the *B* and *I* states will only be seen in *BII** sequences. This may be expressed via the state transition diagram in Fig. 2. The constraints of also dictate the initial model input to the EM process. We use uniform probability distributions subject to the constraints of Fig. 2. So, initial model transition probabilities are given in Fig. 3. In EM, if a parameter is equal to zero, subsequent iterations of the EM process will not "unset" this parameter; thus, this form of initialization is a simple way of encoding constraints on model parameters. We also experimented with random initial models (subject to the constraints in Fig. 2). Uniform initialization usually works slightly better; also, uniform initialization does not require multiple runs of each experiment, as random initialization does.

Motivating the HMM and PRLG. This approach – encoding a chunking problem as a tagging problem and learning to tag with HMMs – goes back to Ramshaw and Marcus (1995). For unsupervised learning, the expectation is that the model will learn to generalize on phrasal boundaries. That is, the models will learn to associate terms like *the* and *a*, which often occur at the beginnings of sentences and rarely at the end, with the tag *B*, which cannot occur at the end of a sentence. Likewise common nouns like *company* or *asset*, which frequently occur at the ends of sentences, but rarely at the beginning, will come to be associated with the *I* tag, which cannot occur at the beginning.

The basic motivation for the PRLG is the assumption that information is lost due to the independence assumption characteristic of the HMM. With so few states, it is feasible to experiment with the more fine-grained PRLG model.

Evaluation. Using the low-level predictions of CCL as as benchmark, we evaluate the HMM and PRLG chunkers on the tasks of constituent chunk and base NP identification. Models were initialized uniformly as illustrated in Fig. 3. Sequence models learn via EM. We report accuracy only after convergence, that is after the change in full dataset perplexity (log inverse probability) is less than %.01 between iterations. Precision, recall and *F*-score are reported for full constituent identification – brackets which do not match the gold standard exactly are false positives.

Model performance results on held-out test datasets are reported in Table 3. 'CCL' refers to the lowest-level constituents extracted from full CCL output, as a benchmark chunker. The sequence models outperform the CCL benchmark at both tasks and on all three datasets. In most cases, the PRLG sequence model performs better than the HMM; the exception is CTB, where the PRLG model is behind the HMM in evaluation, as well as behind CCL.

As the lowest-level constituents of CCL were not specifically designed to describe chunks, we also

Task	Model	English / WSJ			German / Negra			Chinese / CTB		
		Prec	Rec	*F*	Prec	Rec	*F*	Prec	Rec	*F*
Chunking	CCL	57.5	53.5	55.4	28.4	29.6	29.0	23.5	23.9	23.7
	HMM	53.8	62.2	57.7	35.0	37.7	36.3	**37.4**	**41.3**	**39.3**
	PRLG	**76.2**	**63.9**	**69.5**	**39.6**	**47.8**	**43.3**	23.0	18.3	20.3
NP	CCL	46.2	51.1	48.5	15.6	29.2	20.3	10.4	17.3	13.0
	HMM	47.7	65.6	55.2	23.8	46.2	31.4	17.0	**30.8**	21.9
	PRLG	**76.8**	**76.7**	**76.7**	**24.6**	**53.4**	**33.6**	**21.9**	28.5	**24.8**

Table 3: Unsupervised chunking results for local constituent structure identification and NP chunking on held-out test sets. CCL refers to the lowest constituents extracted from CCL output.

	WSJ	Negra	CTB
Chunking	57.8	36.0	25.5
NPs	57.8	38.8	23.2

Table 4: Recall of CCL on the chunking tasks.

POS Sequence	# of errors
TO VB	673
NNP NNP	450
MD VB	407
DT JJ	368
DT NN	280

Table 5: Top 5 POS sequences of the false positives predicted by the HMM.

checked the recall of *all* brackets generated by CCL against gold-standard constituent chunks. The results are given in Table 4. Even compared to this, the sequence models' recall is almost always higher.

The sequence models, as well as the CCL benchmark, show relatively low precision on the Negra corpus. One possible reason for this lies in the design decision of Negra to use relatively flat tree structures. As a result, many structures that in other treebanks would be prepositional phrases with embedded noun phrases – and thus non-local constituents – are flat prepositional phrases here. Examples include "auf die Wiesbadener Staatsanwaelte" (on Wiesbaden's district attorneys) and "in Hannovers Nachbarstadt" (in Hannover's neighbor city).

In fact, in Negra, the sequence model chunkers often find NPs embedded in PPs, which are not annotated as such. For instance, in the PP "hinter den Kulissen" (behind the scenes), both the PRLG and HMM chunkers identify the internal NP, though this is not identified in Negra and thus considered a false positive. The fact that the HMM and PRLG have higher recall on NP identification on Negra than precision is further evidence towards this.

Comparing the HMM and PRLG. To outline some of the factors differentiating the HMM and PRLG, we focus on NP identification in WSJ.

The PRLG has higher precision than the HMM, while the two models are closer in recall. Comparing the predictions directly, the two models often have the same correct predictions and often miss the same gold standard constituents. The improved results of the PRLG are based mostly on the fewer overall brackets predicted, and thus fewer false positives: for WSJ the PRLG incorrectly predicts 2241 NP constituents compared to 6949 for the HMM. Table 5 illustrates the top 5 POS sequences of the false positives predicted by the HMM.[3] (Recall that we use gold standard POS *only* for post-experiment results analysis—the model itself does not have access to them.) By contrast, the sequence representing the largest class of errors of the PRLG is DT NN, with 165 errors – this sequence represents the largest class of predictions for both models.

Two of the top classes of errors, MD VB and TO VB, represent verb phrase constituents, which are often predicted by the HMM chunker, but not by the PRLG. The class represented by NNP NNP corresponds with the tendency of the HMM chunker to split long proper names: for example, it systematically splits *new york stock exchange* into two chunks, (*new york*) (*stock exchange*), whereas the PRLG chunker predicts a single four-word chunk.

The most interesting class is DT JJ, which represents the difficulty the HMM chunker has at dis-

[3] For the Penn Treebank tagset, see Marcus et al. (1993).

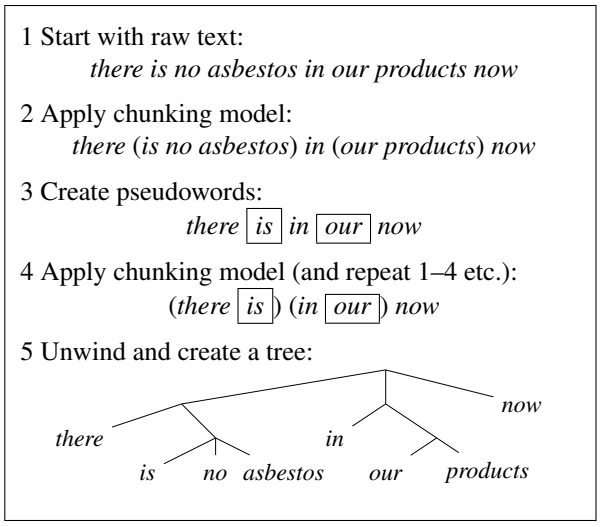


Fig. 4: Cascaded chunking illustrated. Pseudowords are indicated with boxes.

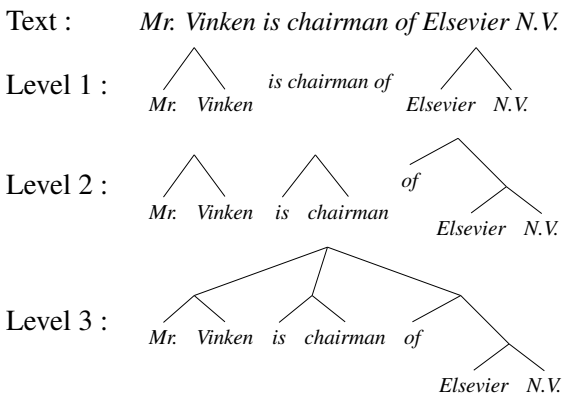


Fig. 5: PRLG cascaded chunker output.

		NPs		PPs	
		Lev 1	Lev 2	Lev 1	Lev 2
WSJ	HMM	66.5	68.1	20.6	70.2
	PRLG	77.5	78.3	9.1	77.6
Negra	HMM	54.7	62.3	24.8	48.1
	PRLG	61.6	65.2	40.3	44.0
CTB	HMM	33.3	35.4	34.6	38.4
	PRLG	30.9	33.6	31.6	47.1

Table 7: NP and PP recall at cascade levels 1 and 2. The level 1 NP numbers differ from the NP chunking numbers from Table 3 since they include root-level constituents which are often NPs.

tinguishing determiner-adjective from determiner-noun pairs. The PRLG chunker systematically gets DT JJ NN trigrams as chunks. The greater context provided by right branching rules allows the model to explicitly estimate separate probabilities for $P(I \rightarrow \textit{recent I})$ versus $P(I \rightarrow \textit{recent O})$. That is, *recent* within a chunk versus ending a chunk. Bigrams like *the acquisition* allow the model to learn rules $P(B \rightarrow \textit{the I})$ and $P(I \rightarrow \textit{acquisition O})$. So, the PRLG is better able to correctly pick out the trigram chunk (*the recent acquisition*).

5 Constituent parsing with a cascade of chunkers

We use cascades of chunkers for full constituent parsing, building hierarchical constituents bottom-up. After chunking is performed, all multiword constituents are collapsed and represented by a single pseudoword. We use an extremely simple, but effective, way to create pseudoword for a chunk: pick the term in the chunk with the highest corpus frequency, and mark it as a pseudoword. The sentence is now a string of symbols (normal words and pseudowords), to which a subsequent unsupervised chunking model is applied. This process is illustrated in Fig. 4.

Each chunker in the cascade chunks the raw text, then regenerates the dataset replacing chunks with pseudowords; this process is iterated until no new chunks are found. The separate chunkers in the cascade are referred to as *levels*. In our experiments the cascade process took a minimum of 5 levels, and a maximum of 7. All chunkers in the cascade have the same settings in terms of smoothing, the tagset and initialization.

Evaluation. Table 6 gives the unlabeled PARSEVAL scores for CCL and the two finite-state models. PRLG achieves the highest F-score for all datasets, and does so by a wide margin for German and Chinese. CCL does achieve higher recall for English.

While the first level of constituent analysis has high precision and recall on NPs, the second level often does well finding prepositional phrases (PPs), especially in WSJ; see Table 7. This is illustrated in Fig. 5. This example also illustrates a PP attachment error, which are a common problem for these models.

We also evaluate using short – 10-word or less – sentences. That said, we maintain the training/test split from before. Also, making use of the open

Parsing Model	English / WSJ			German / Negra			Chinese / CTB		
	Prec	Rec	*F*	Prec	Rec	*F*	Prec	Rec	*F*
CCL	53.6	**50.0**	51.7	33.4	32.6	33.0	37.0	21.6	27.3
HMM	48.2	43.6	45.8	30.8	**50.3**	38.2	43.0	29.8	35.2
PRLG	**60.0**	49.4	**54.2**	**38.8**	47.4	**42.7**	**50.4**	**32.8**	**39.8**

Table 6: Unlabeled PARSEVAL scores for cascaded models.

source implementation by F. Luque,[4] we compare on WSJ and Negra to the *constituent context model* (CCM) of Klein and Manning (2002). CCM learns to predict a set of brackets over a string (in practice, a string of POS tags) by jointly estimating constituent and distituent strings and contexts using an iterative EM-like procedure (though, as noted by Smith and Eisner (2004), CCM is deficient as a generative model). Note that this comparison is methodologically problematic in two respects. On the one hand, CCM is evaluated using gold standard POS sequences as input, so it receives a major source of supervision not available to the other models. On the other hand, the other models use punctuation as an indicator of constituent boundaries, but all punctuation is dropped from the input to CCM. Also, note that CCM performs better when trained on short sentences, so here CCM is trained only on the 10-word-or-less subsets of the training datasets.[5]

The results from the cascaded PRLG chunker are near or better than the best performance by CCL or CCM in these experiments. These and the full-length parsing results suggest that the cascaded chunker strategy generalizes better to longer sentences than does CCL. CCM does very poorly on longer sentences, but does not have the benefit of using punctuation, as do the raw text models; unfortunately, further exploration of this trade-off is beyond the scope of this paper. Finally, note that CCM has higher recall, and lower precision, generally, than the raw text models. This is due, in part, to the chart structure used by CCM in the calculation of constituent and distituent probabilities: as in CKY parsing, the chart structure entails the trees predicted will be binary-branching. CCL and the cascaded models can predict higher-branching constituent structures, so fewer constituents are predicted overall.

		Prec	Rec	*F*
WSJ	CCM	*62.4*	***81.4***	*70.7*
	CCL	71.2	73.1	**72.1**
	HMM	64.4	64.7	64.6
	PRLG	**74.6**	66.7	70.5
Negra	CCM	*52.4*	***83.4***	***64.4***
	CCL	52.9	54.0	53.0
	HMM	47.7	72.0	57.4
	PRLG	**56.3**	72.1	63.2
CTB	CCL	54.4	44.3	48.8
	HMM	55.8	53.1	54.4
	PRLG	**62.7**	**56.9**	**59.6**

Table 8: Evaluation on 10-word-or-less sentences. CCM scores are italicized as a reminder that CCM uses gold-standard POS sequences as input, so its results are not strictly comparable to the others.

6 Phrasal punctuation revisited

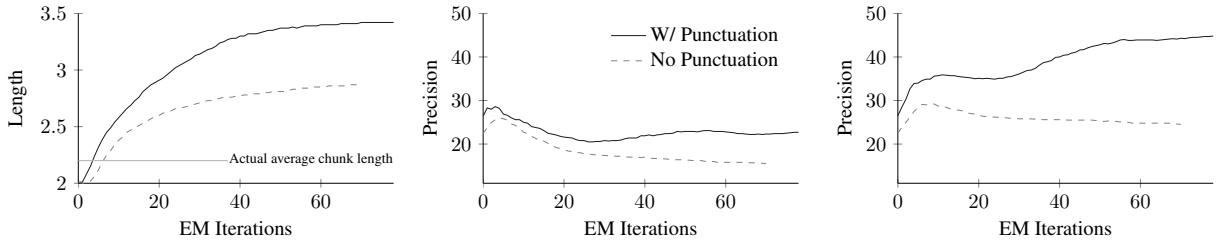
Up to this point, the proposed models for chunking and parsing use phrasal punctuation as a phrasal separator, like CCL. We now consider how well these models perform in absence of this constraint.

Table 9a provides comparison of the sequence models' performance on the constituent chunking task without using phrasal punctuation in training and evaluation. The table shows absolute improvement (+) or decline (−) in precision and recall when phrasal punctuation is removed from the data. The punctuation constraint seems to help the chunkers some, but not very much; ignoring punctuation seems to *improve* chunker results for the HMM on Chinese. Overall, the effect of phrasal punctuation on the chunker models' performance is not clear.

The results for cascaded parsing differ strongly from those for chunking, as Table 9b indicates. Using phrasal punctuation to constrain bracket prediction has a larger impact on cascaded parsing re-

[4] `http://www.cs.famaf.unc.edu.ar/~francolq/en/proyectos/dmvccm/`

[5] This setup is the same as Seginer's (2007), except the train/test split.



a) Average Predicted Constituent Length b) Chunking Precision c) Precision at All Brackets

Fig. 6: Behavior of the PRLG model on CTB over the course of EM.

	WSJ		Negra		CTB	
	Prec	Rec	Prec	Rec	Prec	Rec
HMM	−5.8	−9.8	−0.1	−0.4	+0.7	+4.9
PRLG	−2.5	−2.1	−2.1	−2.1	−7.0	+1.2

a) Constituent Chunking

	WSJ		Negra		CTB	
	Prec	Rec	Prec	Rec	Prec	Rec
CCL	−14.1	−13.5	−10.7	−4.6	−11.6	−6.0
HMM	−7.8	−8.6	−2.8	+1.7	−13.4	−1.2
PRLG	−10.1	−7.2	−4.0	−4.5	−22.0	−11.8

b) (Cascade) Parsing

Table 9: Effects of dropping phrasal punctuation in unsupervised chunking and parsing evaluations relative to Tables 3 and 6.

sults almost across the board. This is not surprising: while performing unsupervised partial parsing from raw text, the sequence models learn two general patterns: i) they learn to chunk rare sequences, such as named entities, and ii) they learn to chunk high-frequency function words next to lower frequency content words, which often correlate with NPs headed by determiners, PPs headed by prepositions and VPs headed by auxiliaries. When these patterns are themselves replaced with pseudowords (see Fig. 4), the models have fewer natural cues to identify constituents. However, within the degrees of freedom allowed by punctuation constraints as described, the chunking models continue to find relatively good constituents.

While CCL makes use of phrasal punctuation in previously reported results, the open source implementation allows it to be evaluated without this constraint. We did so, and report results in Table 9b. CCL is, in fact, very sensitive to phrasal punctuation. Comparing CCL to the cascaded chunkers when none of them use punctuation constraints, the cascaded chunkers (both HMMs and PRLGs) outperform CCL for each evaluation and dataset.

For the CTB dataset, best chunking performance and cascaded parsing performance flips from the HMM to the PRLG. More to the point, the PRLG is actually with worst performing model at the constituent chunking task, but the best performing cascade parser; also, this model has the most serious degrade in performance when phrasal punctuation is dropped from input. To investigate, we track the performance of the chunkers on the development dataset over iterations of EM. This is illustrated in Fig. 6 with the PRLG model. First of all, Fig. 6a reveals the average length of the constituents predicted by the PRLG model increases over the course of EM. However, the average constituent chunk length is 2.22. So, the PRLG chunker is predicting constituents that are longer than the ones targeted in the constituent chunking task: regardless of whether they are legitimate constituents or not, often they will likely be counted as false positives in this evaluation. This is confirmed by observing the constituent chunking precision in Fig. 6b, which peaks when the average predicted constituent length is about the same the actual average length of those in the evaluation. The question, then, is whether the longer chunks predicted correspond to actual constituents or not. Fig. 6c shows that the PRLG, when constrained by phrasal punctuation, does continue to improve its constituent prediction accuracy over the course of EM. These correctly predicted constituents are not counted as such in the constituent chunking or base NP evaluations, but they factor directly into

improved accuracy when this model is part of a cascade.

7 Related work

Our task is the unsupervised analogue of chunking (Abney, 1991), popularized by the 1999 and 2000 Conference on Natural Language Learning shared tasks (Tjong et al., 2000). In fact, our models follow Ramshaw and Marcus (1995), treating structure prediction as sequence prediction using BIO tagging.

In addition to Seginer's CCL model, the unsupervised parsing model of Gao and Suzuki (2003) and Gao et al. (2004) also operates on raw text. Like us, their model gives special treatment to local constituents, using a language model to characterize phrases which are linked via a dependency model. Their output is not evaluated directly using treebanks, but rather applied to several information retrieval problems.

In the supervised realm, Hollingshead et al. (2005) compare context-free parsers with finite-state partial parsing methods. They find that full parsing maintains a number of benefits, in spite of the greater training time required: they can train on less data more effectively than chunkers, and are more robust to shifts in textual domain.

Brants (1999) reports a supervised cascaded chunking strategy for parsing which is strikingly similar to the methods proposed here. In both, Markov models are used in a cascade to predict hierarchical constituent structure; and in both, the parameters for the model at each level are estimated independently. There are major differences, though: the models here are learned from raw text without tree annotations, using EM to train parameters; Brants' cascaded Markov models use supervised maximum likelihood estimation. Secondly, between the separate levels of the cascade, we collapse constituents into symbols which are treated as tokens in subsequent chunking levels; the Markov models in the higher cascade levels in Brants' work actually emit constituent structure. A related approach is that of Schuler et al. (2010), who report a supervised hierarchical hidden Markov model which uses a right-corner transform. This allows the model to predict more complicated trees with fewer levels than in Brants' work or this paper.

8 Conclusion

In this paper we have introduced a new subproblem of unsupervised parsing: unsupervised partial parsing, or unsupervised chunking. We have proposed a model for unsupervised chunking from raw text that is based on standard probabilistic finite-state methods. This model produces better local constituent predictions than the current best unsupervised parser, CCL, across datasets in English, German, and Chinese. By extending these probabilistic finite-state methods in a cascade, we obtain a general unsupervised parsing model. This model outperforms CCL in PARSEVAL evaluation on English, German, and Chinese.

Like CCL, our models operate from raw (albeit segmented) text, and like it our models decode very quickly; however, unlike CCL, our models are based on standard and well-understood computational linguistics technologies (hidden Markov models and related formalisms), and may benefit from new research into these core technologies. For instance, our models may be improved by the application of (unsupervised) discriminative learning techniques with features (Berg-Kirkpatrick et al., 2010); or by incorporating topic models and document information (Griffiths et al., 2005; Moon et al., 2010).

UPPARSE, the software used for the experiments in this paper, is available under an open-source license to facilitate replication and extensions.[6]

Acknowledgments. This material is based upon work supported in part by the U. S. Army Research Laboratory and the U. S. Army Research Office under grant number W911NF-10-1-0533. Support for the first author was also provided by Mike Hogg Endowment Fellowship, the Office of Graduate Studies at The University of Texas at Austin.

This paper benefited from discussion in the Natural Language Learning reading group at UT Austin, especially from Collin Bannard, David Beaver, Matthew Lease, Taesun Moon and Ray Mooney. We also thank the three anonymous reviewers for insightful questions and helpful comments.

[6] `http://elias.ponvert.net/upparse`.

References

S. Abney. 1991. Parsing by chunks. In R. Berwick, S. Abney, and C. Tenny, editors, *Principle-based Parsing*. Kluwer.

P. W. Adriaans, M. Trautwein, and M. Vervoort. 2000. Towards high speed grammar induction on large text corpora. In *SOFSEM*.

T. Berg-Kirkpatrick, A. Bouchard-Côté, J. DeNero, and D. Klein. 2010. Painless unsupervised learning with features. In *HLT-NAACL*.

R. Bod. 2006. Unsupervised parsing with U-DOP. In *CoNLL*.

T. Brants. 1999. Cascaded markov models. In *EACL*.

E. Charniak. 1993. *Statistical Language Learning*. MIT.

S. B. Cohen and N. A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *HLT-NAACL*.

B. Cramer. 2007. Limitations of current grammar induction algorithms. In *ACL-SRW*.

X. Duan, J. Zhao, and B. Xu. 2007. Probabilistic models for action-based Chinese dependency parsing. In *ECML/PKDD*.

J. Gao and H. Suzuki. 2003. Unsupervised learning of dependency structure for language modeling. In *ACL*.

J. Gao, J.Y. Nie, G. Wu, and G. Cao. 2004. Dependence language model for information retrieval. In *SIGIR*.

T. L. Griffiths, M. Steyvers, D. M. Blei, and J. M. Tenenbaum. 2005. Integrating topics and syntax. In *NIPS*.

C. Hänig. 2010. Improvements in unsupervised co-occurence based parsing. In *CoNLL*.

W. P. Headden III, M. Johnson, and D. McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *HLT-NAACL*.

K. Hollingshead, S. Fisher, and B. Roark. 2005. Comparing and combining finite-state and context-free parsers. In *HLT-EMNLP*.

D. Klein and C. D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *ACL*.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL*.

B. Krenn, T. Brants, W. Skut, and Hans Uszkoreit. 1998. A linguistically interpreted corpus of German newspaper text. In *Proceedings of the ESSLLI Workshop on Recent Advances in Corpus Annotation*.

K. Lari and S. J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech & Language*, 4:35 – 56.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Compuational Linguistics*, pages 313–330.

M.P. Marcus, B. Santorini, M.A. Marcinkiewicz, and A. Taylor, 1999. *Treebank-3*. LDC.

T. Moon, J. Baldridge, and K. Erk. 2010. Crouching Dirichlet, hidden Markov model: Unsupervised POS tagging with context local tag generation. In *EMNLP*.

M. Palmer, F. D. Chiou, N. Xue, and T. K. Lee, 2005. *Chinese Treebank 5.0*. LDC.

F. Pereira and Y. Schabes. 1992. Inside-outside reestimation from paritally bracketed corpora. In *ACL*.

E. Ponvert, J. Baldridge, and K. Erk. 2010. Simple unsupervised prediction of low-level constituents. In *ICSC*.

L.R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*.

L. A. Ramshaw and M. P. Marcus. 1995. Text chunking using transformation-based learning. In *Proc. of Third Workshop on Very Large Corpora*.

R. Reichart and A. Rappoport. 2010. Improved fully unsupervised parsing with Zoomed Learning. In *EMNLP*.

W. Schuler, S. AbdelRahman, T. Miller, and L. Schwartz. 2010. Broad-coverage parsing using human-like memory constraints. *Compuational Linguistics*, 3(1).

Y. Seginer. 2007. Fast unsupervised incremental parsing. In *ACL*.

N. A. Smith and J. Eisner. 2004. Annealing techniques for unsupervised statistical language learning. In *ACL*.

N. A. Smith and M. Johnson. 2007. Weighted and probabilistic CFGs. *Computational Lingusitics*.

Z. Solan, D. Horn, E. Ruppin, and S. Edelman. 2005. Unsupervised learning of natural languages. *PNAS*, 102.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2010. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *NAACL-HLT*.

E. F. Tjong, K. Sang, and S. Buchholz. 2000. Introduction to the CoNLL-2000 Shared Task: Chunking. In *CoNLL-LLL*.

M. van Zaanen. 2000. ABL: Alignment-based learning. In *COLING*.

Extracting Paraphrases from Definition Sentences on the Web

Chikara Hashimoto[*] **Kentaro Torisawa**[†] **Stijn De Saeger**[‡]
Jun'ichi Kazama[§] **Sadao Kurohashi**[¶]
[*†‡§] National Institute of Information and Communications Technology
Kyoto, 619-0237, JAPAN
[*¶] Graduate School of Informatics, Kyoto University
Kyoto, 606-8501, JAPAN
{[*] ch, [†] torisawa, [‡] stijn, [§] kazama}@nict.go.jp
[¶] kuro@i.kyoto-u.ac.jp

Abstract

We propose an automatic method of extracting paraphrases from definition sentences, which are also automatically acquired from the Web. We observe that a huge number of concepts are defined in Web documents, and that the sentences that define the same concept tend to convey mostly the same information using different expressions and thus contain many paraphrases. We show that a large number of paraphrases can be automatically extracted with high precision by regarding the sentences that define the same concept as parallel corpora. Experimental results indicated that with our method it was possible to extract about 300,000 paraphrases from 6×10^8 Web documents with a precision rate of about 94%.

1 Introduction

Natural language allows us to express the same information in many ways, which makes natural language processing (NLP) a challenging area. Accordingly, many researchers have recognized that automatic paraphrasing is an indispensable component of intelligent NLP systems (Iordanskaja et al., 1991; McKeown et al., 2002; Lin and Pantel, 2001; Ravichandran and Hovy, 2002; Kauchak and Barzilay, 2006; Callison-Burch et al., 2006) and have tried to acquire a large amount of paraphrase knowledge, which is a key to achieving robust automatic paraphrasing, from corpora (Lin and Pantel, 2001; Barzilay and McKeown, 2001; Shinyama et al., 2002; Barzilay and Lee, 2003).

We propose a method to extract phrasal paraphrases from pairs of sentences that define the same concept. The method is based on our observation that two sentences defining the same concept can be regarded as a parallel corpus since they largely convey the same information using different expressions. Such definition sentences abound on the Web. This suggests that we may be able to extract a large amount of phrasal paraphrase knowledge from the definition sentences on the Web.

For instance, the following two sentences, both of which define the same concept "osteoporosis", include two pairs of phrasal paraphrases, which are indicated by underlines ① and ②, respectively.

(1) a. Osteoporosis is a disease that ① decreases the quantity of bone and ② makes bones fragile.
b. Osteoporosis is a disease that ① reduces bone mass and ② increases the risk of bone fracture.

We define *paraphrase* as a pair of expressions between which entailment relations of both directions hold. (Androutsopoulos and Malakasiotis, 2010).

Our objective is to extract phrasal paraphrases from pairs of sentences that define the same concept. We propose a supervised method that exploits various kinds of lexical similarity features and contextual features. Sentences defining certain concepts are acquired automatically on a large scale from the Web by applying a quite simple supervised method.

Previous methods most relevant to our work used parallel corpora such as multiple translations of the same source text (Barzilay and McKeown, 2001) or automatically acquired parallel news texts (Shinyama et al., 2002; Barzilay and Lee, 2003; Dolan et al., 2004). The former requires a large amount of manual labor to translate the same texts

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1087–1097,
Portland, Oregon, June 19-24, 2011.

in several ways. The latter would suffer from the fact that it is not easy to automatically retrieve large bodies of parallel news text with high accuracy. On the contrary, recognizing definition sentences for the same concept is quite an easy task at least for Japanese, as we will show, and we were able to find a huge amount of definition sentence pairs from normal Web texts. In our experiments, about 30 million definition sentence pairs were extracted from 6×10^8 Web documents, and the estimated number of paraphrases recognized in the definition sentences using our method was about 300,000, for a precision rate of about 94%. Also, our experimental results show that our method is superior to well-known competing methods (Barzilay and McKeown, 2001; Koehn et al., 2007) for extracting paraphrases from definition sentence pairs.

Our evaluation is based on bidirectional checking of entailment relations between paraphrases that considers the context dependence of a paraphrase.

Note that using definition sentences is only the beginning of our research on paraphrase extraction. We have a more general hypothesis that sentences fulfilling the same *pragmatic* function (e.g. definition) for the same topic (e.g. osteoporosis) convey mostly the same information using different expressions. Such functions other than definition may include the usage of the same Linux command, the recipe for the same cuisine, or the description of related work on the same research issue.

Section 2 describes related works. Section 3 presents our proposed method. Section 4 reports on evaluation results. Section 5 concludes the paper.

2 Related Work

The existing work for paraphrase extraction is categorized into two groups. The first involves a distributional similarity approach pioneered by Lin and Pantel (2001). Basically, this approach assumes that two expressions that have a large distributional similarity are paraphrases. There are also variants of this approach that address entailment acquisition (Geffet and Dagan, 2005; Bhagat et al., 2007; Szpektor and Dagan, 2008; Hashimoto et al., 2009). These methods can be applied to a normal monolingual corpus, and it has been shown that a large number of paraphrases or entailment rules could be extracted. However, the precision of these methods has been relatively low. This is due to the fact that the evidence, i.e., distributional similarity, is just indirect evidence of paraphrase/entailment. Accordingly, these methods occasionally mistake antonymous pairs for paraphrases/entailment pairs, since an expression and its antonymous counterpart are also likely to have a large distributional similarity. Another limitation of these methods is that they can find only paraphrases consisting of frequently observed expressions since they must have *reliable* distributional similarity values for expressions that constitute paraphrases.

The second category is a parallel corpus approach (Barzilay and McKeown, 2001; Shinyama et al., 2002; Barzilay and Lee, 2003; Dolan et al., 2004). Our method belongs to this category. This approach aligns expressions between two sentences in parallel corpora, based on, for example, the overlap of words/contexts. The aligned expressions are assumed to be paraphrases. In this approach, the expressions do not need to appear frequently in the corpora. Furthermore, the approach rarely mistakes antonymous pairs for paraphrases/entailment pairs. However, its limitation is the difficulty in preparing a large amount of parallel corpora, as noted before. We avoid this by using definition sentences, which can be easily acquired on a large scale from the Web, as parallel corpora.

Murata et al. (2004) used definition sentences in two manually compiled dictionaries, which are considerably fewer in the number of definition sentences than those on the Web. Thus, the coverage of their method should be quite limited. Furthermore, the precision of their method is much poorer than ours as we report in Section 4.

For a more extensive survey on paraphrasing methods, see Androutsopoulos and Malakasiotis (2010) and Madnani and Dorr (2010).

3 Proposed method

Our method, targeting the Japanese language, consists of two steps: definition sentence acquisition and paraphrase extraction. We describe them below.

3.1 Definition sentence acquisition

We acquire sentences that define a concept (definition sentences) as in Example (2), which defines "骨

粗鬆症" (osteoporosis), from the 6×10^8 Web pages (Akamine et al., 2010) and the Japanese Wikipedia.

(2) 骨粗鬆症**とは、**骨がもろくなってしまう病気だ。
(Osteoporosis is a disease that makes bones fragile.)

Fujii and Ishikawa (2002) developed an unsupervised method to find definition sentences from the Web using 18 sentential templates and a language model constructed from an encyclopedia. On the other hand, we developed a supervised method to achieve a higher precision.

We use one sentential template and an SVM classifier. Specifically, we first collect definition sentence candidates by a template "^NP **とは**.*", where ^ is the beginning of sentence and NP is the noun phrase expressing the concept to be defined followed by a particle sequence, "**と**" (comitative) and "**は**" (topic) (and optionally followed by comma), as exemplified in (2). As a result, we collected 3,027,101 sentences. Although the particle sequence tends to mark the topic of the definition sentence, it can also appear in interrogative sentences and normal assertive sentences in which a topic is strongly emphasized. To remove such non-definition sentences, we classify the candidate sentences using an SVM classifier with a polynominal kernel ($d = 2$).[1] Since Japanese is a head-final language and we can judge whether a sentence is interrogative or not from the last words in the sentence, we included morpheme N-grams and bag-of-words (with the window size of N) at the end of sentences in the feature set. The features are also useful for confirming that the head verb is in the present tense, which definition sentences should be. Also, we added the morpheme N-grams and bag-of-words right after the particle sequence in the feature set since we observe that non-definition sentences tend to have interrogative related words like "何" (what) or "一体" ((what) on earth) right after the particle sequence. We chose 5 as N from our preliminary experiments.

Our training data was constructed from 2,911 sentences randomly sampled from all of the collected sentences. 61.1% of them were labeled as positive. In the 10-fold cross validation, the classifier's accuracy, precision, recall, and F1 were 89.4, 90.7, 92.2, and 91.4, respectively. Using the classifier, we acquired 1,925,052 positive sentences from all of the collected sentences. After adding definition sentences from Wikipedia articles, which are typically the first sentence of the body of each article (Kazama and Torisawa, 2007), we obtained a total of 2,141,878 definition sentence candidates, which covered 867,321 concepts ranging from weapons to rules of baseball. Then, we coupled two definition sentences whose defined concepts were the same and obtained 29,661,812 definition sentence pairs.

Obviously, our method is tailored to Japanese. For a language-independent method of definition acquisition, see Navigli and Velardi (2010) as an example.

3.2 Paraphrase extraction

Paraphrase extraction proceeds as follows. First, each sentence in a pair is parsed by the dependency parser KNP[2] and dependency tree fragments that constitute linguistically well-formed constituents are extracted. The extracted dependency tree fragments are called *candidate phrases* hereafter. We restricted candidate phrases to predicate phrases that consist of at least one dependency relation, do not contain demonstratives, and in which all the leaf nodes are nominal and all of the constituents are consecutive in the sentence. KNP indicates whether each candidate phrase is a predicate based on the POS of the head morpheme. Then, we check all the pairs of candidate phrases between two definition sentences to find paraphrase pairs.[3] In (1), repeated in (3), candidate phrase pairs to be checked include (① decreases the quantity of bone, ① reduces bone mass), (① decreases the quantity of bone, ② increases the risk of bone fracture), (② makes bones fragile, ① reduces bone mass), and (② makes bones fragile, ② increases the risk of bone fracture).

(3) a. Osteoporosis is a disease that ① decreases the quantity of bone and ② makes bones fragile.
b. Osteoporosis is a disease that ① reduces bone mass and ② increases the risk of bone fracture.

[1] We use SVM^{light} available at `http://svmlight.joachims.org/`.

[2] `http://nlp.kuee.kyoto-u.ac.jp/nl-resource/knp.html`.

[3] Our method discards candidate phrase pairs in which one subsumes the other in terms of their character string, or the difference is only one proper noun like "toner cartridges that Apple Inc. made" and "toner cartridges that Xerox made." Proper nouns are recognized by KNP.

f1	The ratio of the number of morphemes shared between two candidate phrases to the number of all of the morphemes in the two phrases.
f2	The ratio of the number of a candidate phrase's morphemes, for which there is a morpheme with small edit distance (1 in our experiment) in another candidate phrase, to the number of all of the morphemes in the two phrases. Note that Japanese has many orthographical variations and edit distance is useful for identifying them.
f3	The ratio of the number of a candidate phrase's morphemes, for which there is a morpheme with the same pronunciation in another candidate phrase, to the number of all of the morphemes in the two phrases. Pronunciation is also useful for identifying orthographic variations. Pronunciation is given by KNP.
f4	The ratio of the number of morphemes of a shorter candidate phrase to that of a longer one.
f5	The identity of the inflected form of the head morpheme between two candidate phrases: 1 if they are identical, 0 otherwise.
f6	The identity of the POS of the head morpheme between two candidate phrases: 1 or 0.
f7	The identity of the inflection (conjugation) of the head morpheme between two candidate phrases: 1 or 0.
f8	The ratio of the number of morphemes that appear in a candidate phrase segment of a definition sentence s_1 and in a segment that is NOT a part of the candidate phrase of another definition sentence s_2 to the number of all of the morphemes of s_1's candidate phrase, i.e. how many extra morphemes are incorporated into s_1's candidate phrase.
f9	The reversed ($s_1 \leftrightarrow s_2$) version of **f8**.
f10	The ratio of the number of parent dependency tree fragments that are shared by two candidate phrases to the number of all of the parent dependency tree fragments of the two phrases. Dependency tree fragments are represented by the pronunciation of their component morphemes.
f11	A variation of **f10**; tree fragments are represented by the base form of their component morphemes.
f12	A variation of **f10**; tree fragments are represented by the POS of their component morphemes.
f13	The ratio of the number of unigrams (morphemes) that appear in the child context of both candidate phrases to the number of all of the child context morphemes of both candidate phrases. Unigrams are represented by the pronunciation of the morpheme.
f14	A variation of **f13**; unigrams are represented by the base form of the morpheme.
f15	A variation of **f14**; the numerator is the number of child context unigrams that are adjacent to both candidate phrases.
f16	The ratio of the number of trigrams that appear in the child context of both candidate phrases to the number of all of the child context morphemes of both candidate phrases. Trigrams are represented by the pronunciation of the morpheme.
f17	Cosine similarity between two definition sentences from which a candidate phrase pair is extracted.

Table 1: Features used by paraphrase classifier.

The paraphrase checking of candidate phrase pairs is performed by an SVM classifier with a linear kernel that classifies each pair of candidate phrases into a paraphrase or a non-paraphrase.[4] Candidate phrase pairs are ranked by their distance from the SVM's hyperplane. Features for the classifier are based on our observation that two candidate phrases tend to be paraphrases if the candidate phrases themselves are sufficiently similar and/or their surrounding contexts are sufficiently similar. Table 1 lists the features used by the classifier.[5] Basically, they represent either the similarity of candidate phrases (**f1-9**) or that of their contexts (**f10-17**). We think that they have various degrees of discriminative power, and thus we use the SVM to adjust their weights. Figure 1 illustrates features **f8-12**, for which you may need supplemental remarks. English is used for ease of explanation. In the figure, **f8** has a positive value since the candidate phrase of s_1 contains morphemes "of bone", which do not appear in the candidate phrase of s_2 but do appear in the other part of s_2, i.e. they are extra morphemes for s_1's candidate phrase. On the other hand, **f9** is zero since there is no such extra morpheme in s_2's candidate phrase. Also, features **f10-12** have positive values since the two candidate phrases share two parent dependency tree fragments, *(that increases)* and *(of fracture)*.

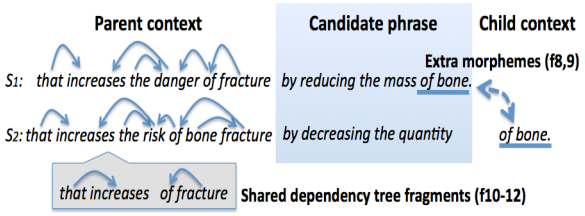


Figure 1: Illustration of features **f8-12**.

We have also tried the following features, which we do not detail due to space limitation: the similarity of candidate phrases based on semantically similar nouns (Kazama and Torisawa, 2008), entailing/entailed verbs (Hashimoto et al., 2009), and the identity of the pronunciation and base form of the head morpheme; N-grams (N=1,2,3) of child and parent contexts represented by either the inflected form, base form, pronunciation, or POS of mor-

[4] We use SVM^{perf} available at `http://svmlight.joachims.org/svm_perf.html`.

[5] In the table, the parent context of a candidate phrase consists of expressions that appear in ancestor nodes of the candidate phrase in terms of the dependency structure of the sentence. Child contexts are defined similarly.

Original definition sentence pair (s_1, s_2)	Paraphrased definition sentence pair (s'_1, s'_2)
s_1: Osteoporosis is a disease that **reduces bone mass** and makes bones fragile.	s'_1: Osteoporosis is a disease that **decreases the quantity of bone** and makes bones fragile.
s_2: Osteoporosis is a disease that **decreases the quantity of bone** and increases the risk of bone fracture.	s'_2: Osteoporosis is a disease that **reduces bone mass** and increases the risk of bone fracture.

Figure 2: Bidirectional checking of entailment relation ($\rightarrow$) of $p_1 \rightarrow p_2$ and $p_2 \rightarrow p_1$. p_1 is "**reduces bone mass**" in s_1 and p_2 is "**decreases the quantity of bone**" in s_2. p_1 and p_2 are exchanged between s_1 and s_2 to generate corresponding paraphrased sentences s'_1 and s'_2. $p_1 \rightarrow p_2$ ($p_2 \rightarrow p_1$) is verified if $s_1 \rightarrow s'_1$ ($s_2 \rightarrow s'_2$) holds. In this case, both of them hold. English is used for ease of explanation.

pheme; parent/child dependency tree fragments represented by either the inflected form, base form, pronunciation, or POS; adjacent versions (cf. **f15**) of N-gram features and parent/child dependency tree features. These amount to 78 features, but we eventually settled on the 17 features in Table 1 through ablation tests to evaluate the discriminative power of each feature.

The ablation tests were conducted using training data that we prepared. In preparing the training data, we faced the problem that the completely random sampling of candidate paraphrase pairs provided us with only a small number of positive examples. Thus, we automatically collected candidate paraphrase pairs that were expected to have a high likelihood of being positive as examples to be labeled. The likelihood was calculated by simply summing all of the 78 feature values that we have tried, since they indicate the likelihood of a given candidate paraphrase pair's being a paraphrase. Note that values of the features f8 and f9 are weighted with -1, since they indicate the unlikelihood. Specifically, we first randomly sampled 30,000 definition sentence pairs from the 29,661,812 pairs, and collected 3,000 candidate phrase pairs that had the highest likelihood from them. The manual labeling of each candidate phrase pair (p_1, p_2) was based on bidirectional checking of entailment relation, $p_1 \rightarrow p_2$ and $p_2 \rightarrow p_1$, with p_1 and p_2 embedded in contexts.

This scheme is similar to the one proposed by Szpektor et al. (2007). We adopt this scheme since paraphrase judgment might be unstable between annotators unless they are given a particular context based on which they make a judgment. As described below, we use definition sentences as contexts. We admit that annotators might be biased by this in some unexpected way, but we believe that this is a more stable method than that without contexts. The labeling process is as follows. First, from each candidate phrase pair (p_1, p_2) and its source definition sentence pair (s_1, s_2), we create two paraphrase sentence pairs (s'_1, s'_2) by exchanging p_1 and p_2 between s_1 and s_2. Then, annotators check if s_1 entails s'_1 and s_2 entails s'_2 so that entailment relations of both directions $p_1 \rightarrow p_2$ and $p_2 \rightarrow p_1$ are checked. Figure 2 shows an example of bidirectional checking. In this example, both entailment relations, $s_1 \rightarrow s'_1$ and $s_2 \rightarrow s'_2$, hold, and thus the candidate phrase pair (p_1, p_2) is judged as positive. We used (p_1, p_2), for which entailment relations of both directions held, as positive examples (1,092 pairs) and the others as negative ones (1,872 pairs).[6]

We built the paraphrase classifier from the training data. As mentioned, candidate phrase pairs were ranked by the distance from the SVM's hyperplane.

4 Experiment

In this paper, our claims are twofold.

I. Definition sentences on the Web are a treasure trove of paraphrase knowledge (Section 4.2).

II. Our method of paraphrase acquisition from definition sentences is more accurate than well-known competing methods (Section 4.1).

We first verify claim II by comparing our method with that of Barzilay and McKeown (2001) (BM method), Moses[7] (Koehn et al., 2007) (SMT method), and that of Murata et al. (2004) (Mrt method). The first two methods are well known for accurately extracting semantically equivalent phrase pairs from parallel corpora.[8] Then, we verify claim

[6] The remaining 36 pairs were discarded as they contained garbled characters of Japanese.

[7] `http://www.statmt.org/moses/`

[8] As anonymous reviewers pointed out, they are unsupervised methods and thus unable to be adapted to definition sen-

I by comparing definition sentence pairs with sentence pairs that are acquired from the Web using Yahoo!JAPAN API[9] as a paraphrase knowledge source. In the latter data set, two sentences of each pair are expected to be semantically similar regardless of whether they are definition sentences. Both sets contain 100,000 pairs.

Three annotators (not the authors) checked evaluation samples. Fleiss' kappa (Fleiss, 1971) was 0.69 (substantial agreement (Landis and Koch, 1977)).

4.1 Our method vs. competing methods

In this experiment, paraphrase pairs are extracted from 100,000 definition sentence pairs that are randomly sampled from the 29,661,812 pairs. Before reporting the experimental results, we briefly describe the BM, SMT, and Mrt methods.

BM method Given parallel sentences like multiple translations of the same source text, the BM method works iteratively as follows. First, it collects from the parallel sentences identical word pairs and their contexts (POS N-grams with indices indicating corresponding words between paired contexts) as positive examples and those of different word pairs as negative ones. Then, each context is ranked based on the frequency with which it appears in positive (negative) examples. The most likely K positive (negative) contexts are used to extract positive (negative) paraphrases from the parallel sentences. Extracted positive (negative) paraphrases and their morpho-syntactic patterns are used to collect additional positive (negative) contexts. All the positive (negative) contexts are ranked, and additional paraphrases and their morpho-syntactic patterns are extracted again. This iterative process finishes if no further paraphrase is extracted or the number of iterations reaches a predefined threshold T. In this experiment, following Barzilay and McKeown (2001), K is 10 and N is 1 to 3. The value of T is not given in their paper. We chose 3 as its value based on our preliminary experiments. Note that paraphrases extracted by this method are not ranked.

SMT method Our SMT method uses Moses (Koehn et al., 2007) and extracts a phrase table, a set of two phrases that are *translations* of each other, given a set of two sentences that are *translations* of each other. If you give Moses *monolingual* parallel sentence pairs, it should extract a set of two phrases that are *paraphrases* of each other. In this experiment, default values were used for all parameters. To rank extracted phrase pairs, we assigned each of them the product of two phrase translation probabilities of both directions that were given by Moses. For other SMT methods, see Quirk et al. (2004) and Bannard and Callison-Burch (2005) among others.

Mrt method Murata et al. (2004) proposed a method to extract paraphrases from two manually compiled dictionaries. It simply regards a difference between two definition sentences of the same word as a paraphrase candidate. Paraphrase candidates are ranked according to an unsupervised scoring scheme that implements their assumption. They assume that a paraphrase candidate tends to be a valid paraphrase if it is surrounded by infrequent strings and/or if it appears multiple times in the data.

In this experiment, we evaluated the unsupervised version of our method in addition to the supervised one described in Section 3.2, in order to compare it fairly with the other methods. The unsupervised method works in the same way as the supervised one, except that it ranks candidate phrase pairs by the sum of all 17 feature values, instead of the distance from the SVM's hyperplane. In other words, no supervised learning is used. All the feature values are weighted with 1, except for f8 and f9, which are weighted with -1 since they indicate the unlikelihood of a candidate phrase pair being paraphrases. BM, SMT, Mrt, and the two versions of our method were used to extract paraphrase pairs from the same 100,000 definition sentence pairs.

Evaluation scheme Evaluation of each paraphrase pair (p_1, p_2) was based on bidirectional checking of entailment relations $p_1 \rightarrow p_2$ and $p_2 \rightarrow p_1$ in a way similar to the labeling of the training data. The difference is that contexts for evaluation are two sentences that are retrieved from the Web and contain p_1 and p_2, instead of definition sentences from which p_1 and p_2 are extracted. This

tences. Nevertheless, we believe that comparing these methods with ours is very informative, since they are known to be accurate and have been influential.

[9] http://developer.yahoo.co.jp/webapi/

is intended to check whether extracted paraphrases are also valid for contexts other than those from which they are extracted. The evaluation proceeds as follows. For the top m paraphrase pairs of each method (in the case of the BM method, randomly sampled m pairs were used, since the method does not rank paraphrase pairs), we retrieved a sentence pair (s_1, s_2) for each paraphrase pair (p_1, p_2) from the Web, such that s_1 contains p_1 and s_2 contains p_2. In doing so, we make sure that neither s_1 nor s_2 are the definition sentences from which p_1 and p_2 are extracted. For each method, we randomly sample n samples from all of the paraphrase pairs (p_1, p_2) for which both s_1 and s_2 are retrieved. Then, from each (p_1, p_2) and (s_1, s_2), we create two paraphrase sentence pairs (s'_1, s'_2) by exchanging p_1 and p_2 between s_1 and s_2. All samples, each consisting of (p_1, p_2), (s_1, s_2), and (s'_1, s'_2), are checked by three human annotators to determine whether s_1 entails s'_1 and s_2 entails s'_2 so that entailment relations of both directions are verified. In advance of evaluation annotation, all the evaluation samples are shuffled so that the annotators cannot find out which sample is given by which method for fairness. We regard each paraphrase pair as correct if at least two annotators judge that entailment relations of both directions hold for it. You may wonder whether only one pair of sentences (s_1, s_2) is enough for evaluation since a correct (wrong) paraphrase pair might be judged as wrong (correct) accidentally. Nevertheless, we suppose that the final evaluation results are reliable if the number of evaluation samples is sufficient. In this experiment, m is 5,000 and n is 200. We use Yahoo!JAPAN API to retrieve sentences.

Graph (a) in Figure 3 shows a precision curve for each method. *Sup* and *Uns* respectively indicate the supervised and unsupervised versions of our method. The figure indicates that *Sup* outperforms all the others and shows a high precision rate of about 94% at the top 1,000. Remember that this is the result of using 100,000 definition sentence pairs. Thus, we estimate that *Sup* can extract about 300,000 paraphrase pairs with a precision rate of about 94%, if we use all 29,661,812 definition sentence pairs that we acquired.

Furthermore, we measured precision after trivial paraphrase pairs were discarded from the evaluation samples of each method. A candidate phrase pair (p_1, p_2) is regarded as trivial if the pronunciation is the same between p_1 and p_2,[10] or all of the content words contained in p_1 are the same as those of p_2. Graph (b) gives a precision curve for each method. Again, *Sup* outperforms the others too, and maintains a precision rate of about 90% until the top 1,000. These results support our claim II.

Definition sentence pairs	Sup	Uns	BM	SMT	Mrt
with trivial	1,381,424		24,049	9,562	18,184
without trivial	1,377,573		23,490	7,256	18,139
Web sentence pairs	Sup	Uns	BM	SMT	Mrt
with trivial	277,172		5,101	4,586	4,978
without trivial	274,720		4,399	2,342	4,958

Table 2: Number of extracted paraphrases.

The upper half of Table 2 shows the number of extracted paraphrases with/without trivial pairs for each method.[11] *Sup* and *Uns* extracted many more paraphrases. It is noteworthy that *Sup* performed the best in terms of both precision rate and the number of extracted paraphrases.

Table 3 shows examples of correct and incorrect outputs of *Sup*. As the examples indicate, many of the extracted paraphrases are not specific to definition sentences and seem very reusable. However, there are few paraphrases involving metaphors or idioms in the outputs due to the nature of definition sentences. In this regard, we do not claim that our method is almighty. We agree with Sekine (2005) who claims that several different methods are required to discover a wider variety of paraphrases.

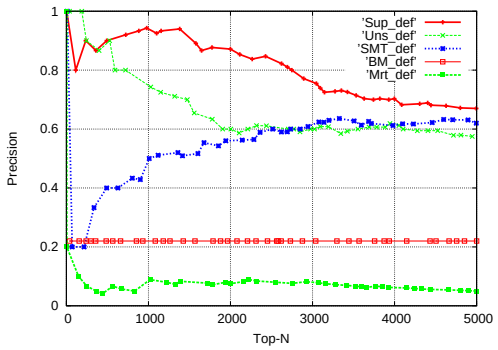
In graphs (a) and (b), the precision of the SMT method goes up as rank goes down. This strange behavior is due to the scoring by Moses that worked poorly for the data; it gave 1.0 to 82.5% of all the samples, 38.8% of which were incorrect. We suspect SMT methods are poor at monolingual alignment for paraphrasing or entailment tasks since, in the tasks, data is much noisier than that used for SMT. See MacCartney et al. (2008) for similar discussion.

4.2 Definition pairs vs. Web sentence pairs

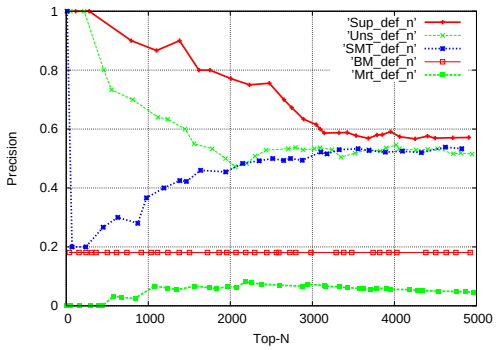
To collect Web sentence pairs, first, we randomly sampled 1.8 million sentences from the Web corpus.

[10] There are many kinds of orthographic variants in Japanese, which can be identified by their pronunciation.

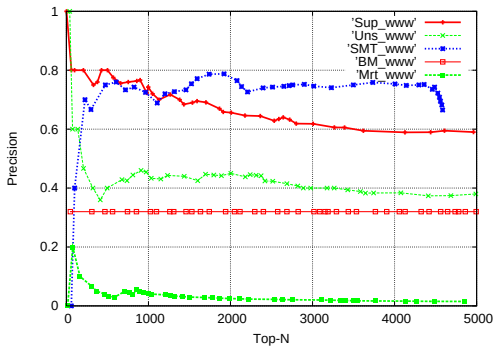
[11] We set no threshold for candidate phrase pairs of each method, and counted all the candidate phrase pairs in Table 2.



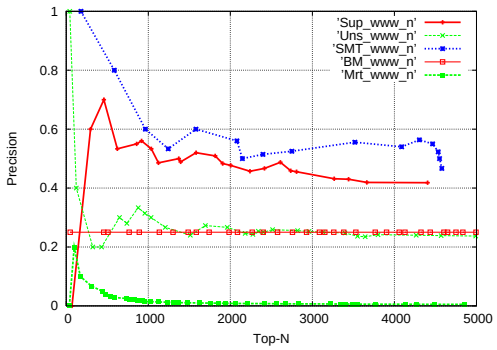
(a) Definition sentence pairs with trivial paraphrases



(b) Definition sentence pairs without trivial paraphrases



(c) Web sentence pairs with trivial paraphrases



(d) Web sentence pairs without trivial paraphrases

Figure 3: Precision curves of paraphrase extraction.

	Rank	Paraphrase pair
Correct	13	メールアドレスにメールを送る (send a message to the e-mail address) ⇔ メールアドレスに電子メールを送る (send an e-mail message to the e-mail address)
	19	お客様の依頼による (requested by a customer) ⇔ お客様の委託による (commissioned by a customer)
	70	企業の財政状況を表す (describe the fiscal condition of company) ⇔ 企業の財政状態を示す (indicate the fiscal state of company)
	112	インフォメーションを得る (get information) ⇔ ニュースを得る (get news)
	656	きまりのことです (it is a convention) ⇔ ルールのことです (it is a rule)
	841	地震のエネルギー規模をあらわす (represent the energy scale of earthquake) ⇔ 地震の規模を表す (represent the scale of earthquake)
	929	細胞を酸化させる (cause the oxidation of cells) ⇔ 細胞を老化させる (cause cellular aging)
	1,553	角質を取り除く (remove dead skin cells) ⇔ 角質をはがす (peel off dead skin cells)
	2,243	胎児の発育に必要だ (required for the development of fetus) ⇔ 胎児の発育成長に必要不可欠だ (indispensable for the growth and development of fetus)
	2,855	視力を矯正する (correct eyesight) ⇔ 視力矯正を行う (perform eyesight correction)
	2,931	チャラにしてもらう (call it even) ⇔ 帳消しにしてもらう (call it quits)
	3,667	ハードディスク上に蓄積される (accumulated on a hard disk) ⇔ ハードディスクドライブに保存される (stored on a hard disk drive)
	4,870	有害物質を排泄する (excrete harmful substance) ⇔ 有害毒素を排出する (discharge harmful toxin)
	5,501	１つのＣＰＵの内部に２つのプロセッサコアを搭載する (mount two processor cores on one CPU) ⇔ １つのパッケージに２つのプロセッサコアを集積する (build two processor cores into one package)
	10,675	外貨を売買する (trade foreign currencies) ⇔ 通貨を交換する (exchange one currency for another)
	112,819	派遣先企業の社員になる (become a regular staff member of the company where (s)he has worked as a temp) ⇔ 派遣先に直接雇用される (employed by the company where (s)he has worked as a temp)
	193,553	Ｗｅｂサイトにアクセスする (access Web sites) ⇔ WWWサイトを訪れる (visit WWW sites)
Incorrect	903	ブラウザに送信される (send to a Web browser) ⇔ パソコンに送信される (send to a PC)
	2,530	調和をはかる (intend to balance) ⇔ リフレッシュを図る (intend to refresh)
	3,008	消化酵素では消化できない (unable to digest with digestive enzymes) ⇔ 消化酵素で消化され難い (hard to digest with digestive enzymes)

Table 3: Examples of correct and incorrect paraphrases extracted by our supervised method with their rank.

We call them sampled sentences. Then, using Yahoo!JAPAN API, we retrieved up to 20 snippets relevant to each sampled sentence using all of the nouns in each sentence as a query. After that, each snippet was split into sentences, which we call snippet sentences. We paired a sampled sentence and a snippet sentence that was the most similar to the sampled sentence. Similarity is the number of nouns shared by the two sentences. Finally, we randomly sampled 100,000 pairs from all the pairs.

Paraphrase pairs were extracted from the Web sentence pairs by using BM, SMT, Mrt and the supervised and unsupervised versions of our method. The features used with our methods were selected from all of the 78 features mentioned in Section 3.2 so that they performed well for Web sentence pairs. Specifically, the features were selected by ablation tests using training data that was tailored to Web sentence pairs. The training data consisted of 2,741 sentence pairs that were collected in the same way as the Web sentence pairs and was labeled in the same way as described in Section 3.2.

Graph (c) of Figure 3 shows precision curves. We also measured precision without trivial pairs in the same way as the previous experiment. Graph (d) shows the results. The lower half of Table 2 shows the number of extracted paraphrases with/without trivial pairs for each method.

Note that precision figures of our methods in graphs (c) and (d) are lower than those of our methods in graphs (a) and (b). Additionally, none of the methods achieved a precision rate of 90% using Web sentence pairs.[12] We think that a precision rate of at least 90% would be necessary if you apply automatically extracted paraphrases to NLP tasks without manual annotation. Only the combination of *Sup* and definition sentence pairs achieved that precision.

Also note that, for all of the methods, the numbers of extracted paraphrases from Web sentence pairs are fewer than those from definition sentence pairs.

From all of these results, we conclude that our claim I is verified.

[12]Precision of SMT is unexpectedly good. We found some Web sentence pairs consisting of two mostly identical sentences on rare occasions. The method worked relatively well for them.

5 Conclusion

We proposed a method of extracting paraphrases from definition sentences on the Web. From the experimental results, we conclude that the following two claims of this paper are verified.

1. Definition sentences on the Web are a treasure trove of paraphrase knowledge.

2. Our method extracts many paraphrases from the definition sentences on the Web accurately; it can extract about 300,000 paraphrases from 6×10^8 Web documents with a precision rate of about 94%.

Our future work is threefold. First, we will release extracted paraphrases from all of the 29,661,812 definition sentence pairs that we acquired, after human annotators check their validity. The result will be available through the ALAGIN forum.[13]

Second, we plan to induce paraphrase *rules* from paraphrase *instances*. Though our method can extract a variety of paraphrase *instances* on a large scale, their coverage might be insufficient for real NLP applications since some paraphrase phenomena are highly productive. Therefore, we need paraphrase *rules* in addition to paraphrase *instances*. Barzilay and McKeown (2001) induced simple POS-based paraphrase rules from paraphrase instances, which can be a good starting point.

Finally, as mentioned in Section 1, the work in this paper is only the beginning of our research on paraphrase extraction. We are trying to extract far more paraphrases from a set of sentences fulfilling the same *pragmatic* function (e.g. definition) for the same topic (e.g. osteoporosis) on the Web. Such functions other than definition may include the usage of the same Linux command, the recipe for the same cuisine, or the description of related work on the same research issue.

Acknowledgments

We would like to thank Atsushi Fujita, Francis Bond, and all of the members of the Information Analysis Laboratory, Universal Communication Research Institute at NICT.

[13]http://alagin.jp/

References

Susumu Akamine, Daisuke Kawahara, Yoshikiyo Kato, Tetsuji Nakagawa, Yutaka I. Leon-Suematsu, Takuya Kawada, Kentaro Inui, Sadao Kurohashi, and Yutaka Kidawara. 2010. Organizing information on the web to support user judgments on information credibility. In *Proceedings of 2010 4th International Universal Communication Symposium Proceedings (IUCS 2010)*, pages 122–129.

Ion Androutsopoulos and Prodromos Malakasiotis. 2010. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research*, 38:135–187.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-2005)*, pages 597–604.

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

Regina Barzilay and Kathleen R. McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th Annual Meeting of the ACL joint with the 10th Meeting of the European Chapter of the ACL (ACL/EACL 2001)*, pages 50–57.

Rahul Bhagat, Patrick Pantel, and Eduard Hovy. 2007. Ledir: An unsupervised algorithm for learning directionality of inference rules. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP2007)*, pages 161–170.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proceedings of the 2006 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2006)*, pages 17–24.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics (COLING 2004)*, pages 350–356.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

Atsushi Fujii and Tetsuya Ishikawa. 2002. Extraction and organization of encyclopedic knowledge information using the World Wide Web (written in Japanese). *Institute of Electronics, Information, and Communication Engineers*, J85-D-II(2):300–307.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 107–114.

Chikara Hashimoto, Kentaro Torisawa, Kow Kuroda, Stijn De Saeger, Masaki Murata, and Jun'ichi Kazama. 2009. Large-scale verb entailment acquisition from the web. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*, pages 1172–1181.

Lidija Iordanskaja, Richard Kittredge, and Alain Polguère. 1991. Lexical selection and paraphrase in a meaning-text generation model. In Cécile L. Paris, William R. Swartout, and William C. Mann, editors, *Natural language generation in artificial intelligence and computational linguistics*, pages 293–312. Kluwer Academic Press.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proceedings of the 2006 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2006)*, pages 455–462.

Jun'ichi Kazama and Kentaro Torisawa. 2007. Exploiting Wikipedia as external knowledge for named entity recognition. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 698–707, June.

Jun'ichi Kazama and Kentaro Torisawa. 2008. Inducing gazetteers for named entity recognition by large-scale clustering of dependency relations. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT)*, pages 407–415.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)*, pages 177–180.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.

Bill MacCartney, Michel Galley, and Christopher D. Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of the 2008*

Conference on Empirical Methods in Natural Language Processing (EMNLP-2008), pages 802–811.

Nitin Madnani and Bonnie Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3).

Kathleen R. McKeown, Regina Barzilay, David Evans, Vasileios Hatzivassiloglou, Judith L. Klavans, Ani Nenkova, Carl Sable, Barry Schiffman, and Sergey Sigelman. 2002. Tracking and summarizing news on a daily basis with columbia's newsblaster. In *Proceedings of the 2nd international conference on Human Language Technology Research*, pages 280–285.

Masaki Murata, Toshiyuki Kanemaru, and Hitoshi Isahara. 2004. Automatic paraphrase acquisition based on matching of definition sentences in plural dictionaries (written in Japanese). *Journal of Natural Language Processing*, 11(5):135–149.

Roberto Navigli and Paola Velardi. 2010. Learning word-class lattices for definition and hypernym extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, pages 1318–1327.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-2004)*, pages 142–149.

Deepak Ravichandran and Eduard H. Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 41–47.

Satoshi Sekine. 2005. Automatic paraphrase discovery based on context and keywords between ne pairs. In *Proceedings of the Third International Workshop on Paraphrasing (IWP-2005)*, pages 80–87.

Yusuke Shinyama, Satoshi Sekine, and Kiyoshi Sudo. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of the 2nd international Conference on Human Language Technology Research (HLT2002)*, pages 313–318.

Idan Szpektor and Ido Dagan. 2008. Learning entailment rules for unary template. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING2008)*, pages 849–856.

Idan Szpektor, Eyal Shnarch, and Ido Dagan. 2007. Instance-based evaluation of entailment rule acquisition. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*, pages 456–463.

Learning From Collective Human Behavior to Introduce Diversity in Lexical Choice

Vahed Qazvinian
Department of EECS
University of Michigan
Ann Arbor, MI
vahed@umich.edu

Dragomir R. Radev
School of Information
Department of EECS
University of Michigan
Ann Arbor, MI
radev@umich.edu

Abstract

We analyze collective discourse, a collective human behavior in content generation, and show that it exhibits diversity, a property of general collective systems. Using extensive analysis, we propose a novel paradigm for designing summary generation systems that reflect the diversity of perspectives seen in real-life collective summarization. We analyze 50 sets of summaries written by human about the same story or artifact and investigate the diversity of perspectives across these summaries. We show how different summaries use various phrasal information units (i.e., *nuggets*) to express the same atomic semantic units, called *factoids*. Finally, we present a ranker that employs *distributional similarities* to build a network of words, and captures the diversity of perspectives by detecting communities in this network. Our experiments show how our system outperforms a wide range of other document ranking systems that leverage diversity.

1 Introduction

In sociology, the term *collective behavior* is used to denote mass activities that are not centrally coordinated (Blumer, 1951). Collective behavior is different from group behavior in the following ways: (a) it involves limited social interaction, (b) membership is fluid, and (c) it generates weak and unconventional norms (Smelser, 1963). In this paper, we focus on the computational analysis of *collective discourse*, a collective behavior seen in interactive content contribution and text summarization in online social media. In collective discourse each individual's behavior is largely independent of that of other individuals.

In social media, discourse (Grosz and Sidner, 1986) is often a *collective reaction* to an event. One scenario leading to collective reaction to a well-defined subject is when an event occurs (a movie is released, a story occurs, a paper is published) and people independently write about it (movie reviews, news headlines, citation sentences). This process of content generation happens over time, and each person chooses the aspects to cover. Each event has an onset and a time of death after which nothing is written about it. Tracing the generation of content over many instances will reveal temporal patterns that will allow us to make sense of the text generated around a particular event.

To understand collective discourse, we are interested in behavior that happens over a short period of time. We focus on topics that are relatively well-defined in scope such as a particular event or a single news event that does not evolve over time. This can eventually be extended to events and issues that are evolving either in time or scope such as elections, wars, or the economy.

In social sciences and the study of complex systems a lot of work has been done to study such collective systems, and their properties such as self-organization (Page, 2007) and diversity (Hong and Page, 2009; Fisher, 2009). However, there is little work that studies a collective system in which members individually write summaries.

In most of this paper, we will be concerned with developing a complex systems view of the set of collectively written summaries, and give evidence of

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1098–1108,
Portland, Oregon, June 19-24, 2011.

the diversity of perspectives and its cause. We believe that out experiments will give insight into new models of text generation, which is aimed at modeling the process of producing natural language texts, and is best characterized as the process of making choices between alternate linguistic realizations, also known as lexical choice (Elhadad, 1995; Barzilay and Lee, 2002; Stede, 1995).

2 Prior Work

In summarization, a number of previous methods have focused on diversity. (Mei et al., 2010) introduce a diversity-focused ranking methodology based on reinforced random walks in information networks. Their random walk model introduces the rich-gets-richer mechanism to PageRank with reinforcements on transition probabilities between vertices. A similar ranking model is the *Grasshopper* ranking model (Zhu et al., 2007), which leverages an absorbing random walk. This model starts with a regular time-homogeneous random walk, and in each step the node with the highest weight is set as an absorbing state. The multi-view point summarization of opinionated text is discussed in (Paul et al., 2010). Paul et al. introduce *Comparative LexRank*, based on the LexRank ranking model (Erkan and Radev, 2004). Their random walk formulation is to score sentences and pairs of sentences from opposite viewpoints (clusters) based on both their representativeness of the collection as well as their contrastiveness with each other. Once a lexical similarity graph is built, they modify the graph based on cluster information and perform LexRank on the modified cosine similarity graph.

The most well-known paper that address diversity in summarization is (Carbonell and Goldstein, 1998), which introduces Maximal Marginal Relevance (MMR). This method is based on a greedy algorithm that picks sentences in each step that are the least similar to the summary so far. There are a few other diversity-focused summarization systems like C-LexRank (Qazvinian and Radev, 2008), which employs document clustering. These papers try to increase diversity in summarizing documents, but do not explain the type of the diversity in their inputs. In this paper, we give an insightful discussion on the nature of the diversity seen in collective discourse, and will explain why some of the mentioned methods may not work under such environments.

In prior work on evaluating independent contributions in content generation, Voorhees (Voorhees, 1998) studied IR systems and showed that relevance judgments differ significantly between humans but relative rankings show high degrees of stability across annotators. However, perhaps the closest work to this paper is (van Halteren and Teufel, 2004) in which 40 Dutch students and 10 NLP researchers were asked to summarize a BBC news report, resulting in 50 different summaries. Teufel and van Halteren also used 6 DUC[1]-provided summaries, and annotations from 10 student participants and 4 additional researchers, to create 20 summaries for another news article in the DUC datasets. They calculated the Kappa statistic (Carletta, 1996; Krippendorff, 1980) and observed high agreement, indicating that the task of atomic semantic unit (factoid) extraction can be robustly performed in naturally occurring text, without any copy-editing.

The diversity of perspectives and the unprecedented growth of the factoid inventory also affects evaluation in text summarization. Evaluation methods are either extrinsic, in which the summaries are evaluated based on their quality in performing a specific task (Spärck-Jones, 1999) or intrinsic where the quality of the summary itself is evaluated, regardless of any applied task (van Halteren and Teufel, 2003; Nenkova and Passonneau, 2004). These evaluation methods assess the information content in the summaries that are generated automatically.

Finally, recent research on analyzing online social media shown a growing interest in mining news stories and headlines because of its broad applications ranging from "meme" tracking and spike detection (Leskovec et al., 2009) to text summarization (Barzilay and McKeown, 2005). In similar work on blogs, it is shown that detecting topics (Kumar et al., 2003; Adar et al., 2007) and sentiment (Pang and Lee, 2004) in the blogosphere can help identify influential bloggers (Adar et al., 2004; Java et al., 2006) and mine opinions about products (Mishne and Glance, 2006).

[1] Document Understanding Conference

3 Data Annotation

The datasets used in our experiments represent two completely different categories: news headlines, and scientific citation sentences. The *headlines* datasets consist of 25 clusters of news headlines collected from Google News[2], and the *citations* datasets have 25 clusters of citations to specific scientific papers from the ACL Anthology Network (AAN)[3]. Each cluster consists of a number of unique summaries (headlines or citations) about the same artifact (non-evolving news story or scientific paper) written by different people. Table 1 lists some of the clusters with the number of summaries in them.

ID	type	Name	Story/Title	#
1	hdl	miss	Miss Venezuela wins miss universe'09	125
2	hdl	typhoon	Second typhoon hit philippines	100
3	hdl	russian	Accident at Russian hydro-plant	101
4	hdl	redsox	Boston Red Sox win world series	99
5	hdl	gervais	"Invention of Lying" movie reviewed	97
	...	...	...	
25	hdl	yale	Yale lab tech in court	10
26	cit	N03-1017	Statistical Phrase-Based Translation	172
27	cit	P02-1006	Learning Surface Text Patterns ...	72
28	cit	P05-1012	On-line Large-Margin Training ...	71
29	cit	C96-1058	Three New Probabilistic Models ...	66
30	cit	P05-1033	A Hierarchical Phrase-Based Model ...	65
	...	...	...	
50	cit	H05-1047	A Semantic Approach to Recognizing ...	7

Table 1: Some of the annotated datasets and the number of summaries in each of them (hdl = headlines; cit = citations)

3.1 Nuggets vs. Factoids

We define an annotation task that requires explicit definitions that distinguish between phrases that represent the same or different information units. Unfortunately, there is little consensus in the literature on such definitions. Therefore, we follow (van Halteren and Teufel, 2003) and make the following distinction. We define a *nugget* to be a phrasal information unit. Different nuggets may all represent the same atomic semantic unit, which we call as a *factoid*. In the following headlines, which are randomly extracted from the `redsox` dataset, nuggets are manually underlined.

red sox win 2007 world series
boston red sox blank rockies to clinch world series
boston fans celebrate world series win; 37 arrests reported

These 3 headlines contain 9 nuggets, which represent 5 factoids or classes of equivalent nuggets.

f_1 : {red sox, boston, boston red sox}
f_2 : {2007 world series, world series win, world series}
f_3 : {rockies}
f_4 : {37 arrests}
f_5 : {fans celebrate}

This example suggests that different headlines on the *same story* written independently of one another use different phrases (nuggets) to refer to the same semantic unit (e.g., "red sox" vs. "boston" vs. "boston red sox") or to semantic units corresponding to different aspects of the story (e.g., "37 arrests" vs. "rockies"). In the former case different nuggets are used to represent the same factoid, while in the latter case different nuggets are used to express different factoids. This analogy is similar to the definition of factoids in (van Halteren and Teufel, 2004).

The following citation sentences to Koehn's work suggest that a similar phenomenon also happens in citations.

We also compared our model with pharaoh (Koehn et al, 2003).
Koehn et al (2003) find that phrases longer than three words improve performance little.
Koehn et al (2003) suggest limiting phrase length to three words or less.
For further information on these parameter settings, confer (koehn et al, 2003).

where the first author mentions "pharaoh" as a contribution of Koehn et al, but the second and third use different nuggets to represent the same contribution: use of trigrams. However, as the last citation shows, a citation sentence, unlike news headlines, may cover no information about the target paper.

The use of phrasal information as nuggets is an essential element to our experiments, since some headline writers often try to use uncommon terms to refer to a factoid. For instance, two headlines from the `redsox` cluster are:

Short wait for bossox this time
Soxcess started upstairs

[2]news.google.com
[3]http://clair.si.umich.edu/clair/anthology/

Following these examples, we asked two annotators to annotate all $1,390$ headlines, and 926 citations. The annotators were asked to follow precise guidelines in nugget extraction. Our guidelines instructed annotators to extract non-overlapping phrases from each headline as nuggets. Therefore, each nugget should be a substring of the headline that represents a semantic unit[4].

Previously (Lin and Hovy, 2002) had shown that information overlap judgment is a difficult task for human annotators. To avoid such a difficulty, we enforced our annotators to extract non-overlapping nuggets from a summary to make sure that they are mutually independent and that information overlap between them is minimized.

Finding agreement between annotated well-defined nuggets is straightforward and can be calculated in terms of Kappa. However, when nuggets themselves are to be extracted by annotators, the task becomes less obvious. To calculate the agreement, we annotated 10 randomly selected headline clusters twice and designed a simple evaluation scheme based on Kappa[5]. For each n-gram, w, in a given headline, we look if w is part of any nugget in either human annotations. If w occurs in both or neither, then the two annotators agree on it, and otherwise they do not. Based on this agreement setup, we can formalize the κ statistic as $\kappa = \frac{\Pr(a)-\Pr(e)}{1-\Pr(e)}$ where $Pr(a)$ is the relative observed agreement among annotators, and $Pr(e)$ is the probability that annotators agree by chance if each annotator is randomly assigning categories.

Table 2 shows the unigram, bigram, and trigram-based average κ between the two human annotators (**Human1**, **Human2**). These results suggest that human annotators can reach substantial agreement when bigram and trigram nuggets are examined, and has reasonable agreement for unigram nuggets.

4 Diversity

We study the diversity of ways with which human summarizers talk about the same story or event and explain why such a diversity exists.

[4]Before the annotations, we lower-cased all summaries and removed duplicates

[5]Previously (Qazvinian and Radev, 2010) have shown high agreement in human judgments in a similar task on citation annotation

	Average κ		
	unigram	bigram	trigram
Human1 vs. Human2			
	0.76 ± 0.4	0.80 ± 0.4	0.89 ± 0.3

Table 2: Agreement between different annotators in terms of average Kappa in 25 headline clusters.

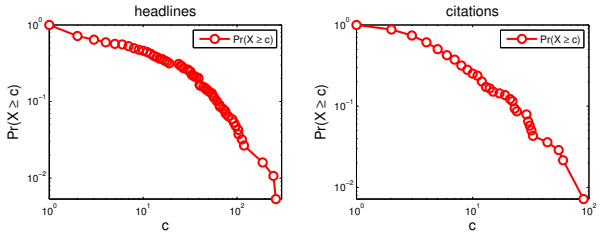


Figure 1: The cumulative probability distribution for the frequency of factoids (i.e., the probability that a factoid will be mentioned in c different summaries) across in each category.

4.1 Skewed Distributions

Our first experiment is to analyze the popularity of different factoids. For each factoid in the annotated clusters, we extract its count, X, which is equal to the number of summaries it has been mentioned in, and then we look at the distribution of X. Figure 1 shows the cumulative probability distribution for these counts (i.e., the probability that a factoid will be mentioned in at least c different summaries) in both categories.

These highly skewed distributions indicate that a large number of factoids (more than 28%) are only mentioned once across different clusters (e.g., "poor pitching of colorado" in the `redsox` cluster), and that a few factoids are mentioned in a large number of headlines (likely using different nuggets). The large number of factoids that are only mentioned in one headline indicates that different summarizers increase diversity by focusing on different aspects of a story or a paper. The set of nuggets also exhibit similar skewed distributions. If we look at individual nuggets, the `redsox` set shows that about 63 (or 80%) of the nuggets get mentioned in only one headline, resulting in a right-skewed distribution.

The factoid analysis of the datasets reveals two main causes for the content diversity seen in headlines: (1) writers focus on different aspects of the story and therefore write about different factoids

(e.g., "celebrations" vs. "poor pitching of colorado"). (2) writer use different nuggets to represent the same factoid (e.g., "redsox" vs. "bosox"). In the following sections we analyze the extent at which each scenario happens.

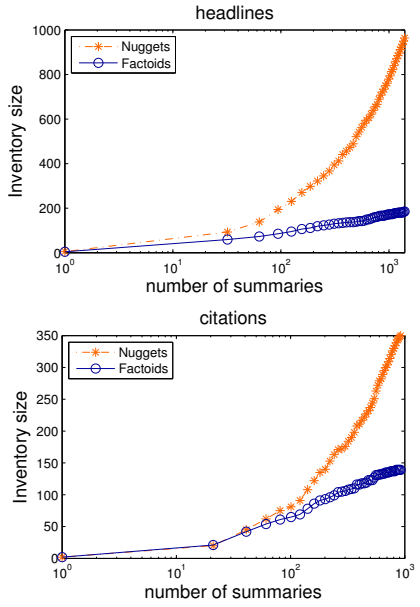


Figure 2: The number of unique factoids and nuggets observed by reading n random summaries in all the clusters of each category

4.2 Factoid Inventory

The emergence of diversity in covering different factoids suggests that looking at more summaries will capture a larger number of factoids. In order to analyze the growth of the factoid inventory, we perform a simple experiment. We shuffle the set of summaries from all 25 clusters in each category, and then look at the number of unique factoids and nuggets seen after reading n^{th} summary. This number shows the amount of information that a randomly selected subset of n writers represent. This is important to study in order to find out whether we need a large number of summaries to capture all aspects of a story and build a complete factoid inventory. The plot in Figure 4.1 shows, at each n, the number of unique factoids and nuggets observed by reading n random summaries from the 25 clusters in each category. These curves are plotted on a semi-log scale to emphasize the difference between the growth patterns of the nugget inventories and the factoid inventories[6].

This finding numerically confirms a similar observation on human summary annotations discussed in (van Halteren and Teufel, 2003; van Halteren and Teufel, 2004). In their work, van Halteren and Teufel indicated that more than 10-20 human summaries are needed for a full factoid inventory. However, our experiments with nuggets of nearly $2,400$ independent human summaries suggest that neither the nugget inventory nor the number of factoids will be likely to show asymptotic behavior. However, these plots show that the nugget inventory grows at a much faster rate than factoids. This means that a lot of the diversity seen in human summarization is a result of the so called different *lexical choices* that represent the same semantic units or factoids.

4.3 Summary Quality

In previous sections we gave evidence for the diversity seen in human summaries. However, a more important question to answer is whether these summaries all cover important aspects of the story. Here, we examine the quality of these summaries, study the distribution of information coverage in them, and investigate the number of summaries required to build a complete factoid inventory.

The information covered in each summary can be determined by the set of factoids (and not nuggets) and their frequencies across the datasets. For example, in the `redsox` dataset, "red sox", "boston", and "boston red sox" are nuggets that all represent the same piece of information: the red sox team. Therefore, different summaries that use these nuggets to refer to the red sox team should not be seen as very different.

We use the Pyramid model (Nenkova and Passonneau, 2004) to value different summary factoids. Intuitively, factoids that are mentioned more frequently are more salient aspects of the story. Therefore, our pyramid model uses the normalized frequency at which a factoid is mentioned across a dataset as its weight. In the pyramid model, the individual factoids fall in tiers. If a factoid appears in more summaries, it falls in a higher tier. In principle, if the term w_i appears $|w_i|$ times in the set of

[6]Similar experiment using individual clusters exhibit similar behavior

headlines it is assigned to the tier $T_{|w_i|}$. The pyramid score that we use is computed as follows. Suppose the pyramid has n tiers, T_i, where tier T_n is the top tier and T_1 is the bottom. The weight of the factoids in tier T_i will be i (i.e. they appeared in i summaries). If $|T_i|$ denotes the number of factoids in tier T_i, and D_i is the number of factoids in the summary that appear in T_i, then the total factoid weight for the summary is $D = \sum_{i=1}^{n} i \times D_i$. Additionally, the optimal pyramid score for a summary is $Max = \sum_{i=1}^{n} i \times |T_i|$. Finally, the pyramid score for a summary can be calculated as

$$P = \frac{D}{Max}$$

Based on this scoring scheme, we can use the annotated datasets to determine the quality of individual headlines. First, for each set we look at the variation in pyramid scores that individual summaries obtain in their set. Figure 3 shows, for each cluster, the variation in the pyramid scores (25th to 75th percentile range) of individual summaries evaluated against the factoids of that cluster. This figure indicates that the pyramid score of almost all summaries obtain values with high variations in most of the clusters For instance, individual headlines from `redsox` obtain pyramid scores as low as 0.00 and as high as 0.93. This high variation confirms the previous observations on diversity of information coverage in different summaries.

Additionally, this figure shows that headlines generally obtain higher values than citations when considered as summaries. One reason, as explained before, is that a citation may not cover any important contribution of the paper it is citing, when headlines generally tend to cover some aspects of the story.

High variation in quality means that in order to capture a larger information content we need to read a greater number of summaries. But how many headlines should one read to capture a desired level of information content? To answer this question, we perform an experiment based on drawing random summaries from the pool of all the clusters in each category. We perform a Monte Carlo simulation, in which for each n, we draw n random summaries, and look at the pyramid score achieved by reading these headlines. The pyramid score is calculated using the factoids from all 25 clusters in each category[7]. Each experiment is repeated $1,000$ times to find the statistical significance of the experiment and the variation from the average pyramid scores.

Figure 4.3 shows the average pyramid scores over different n values in each category on a log-log scale. This figure shows how pyramid score grows and approaches 1.00 rapidly as more randomly selected summaries are seen.

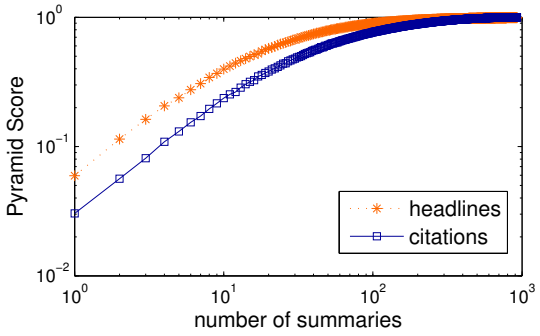


Figure 4: Average pyramid score obtained by reading n random summaries shows rapid asymptotic behavior.

5 Diversity-based Ranking

In previous sections we showed that the diversity seen in human summaries could be according to different nuggets or phrases that represent the same factoid. Ideally, a summarizer that seeks to increase diversity should capture this phenomenon and avoid covering redundant nuggets. In this section, we use different state of the art summarization systems to rank the set of summaries in each cluster with respect to information content and diversity. To evaluate each system, we cut the ranked list at a constant length (in terms of the number of words) and calculate the pyramid score of the remaining text.

5.1 Distributional Similarity

We have designed a summary ranker that will produce a ranked list of documents with respect to the diversity of their contents. Our model works based on ranking individual words and using the ranked list of words to rank documents that contain them.

In order to capture the nuggets of equivalent semantic classes, we use a *distributional similarity* of

[7] Similar experiment using individual clusters exhibit similar results

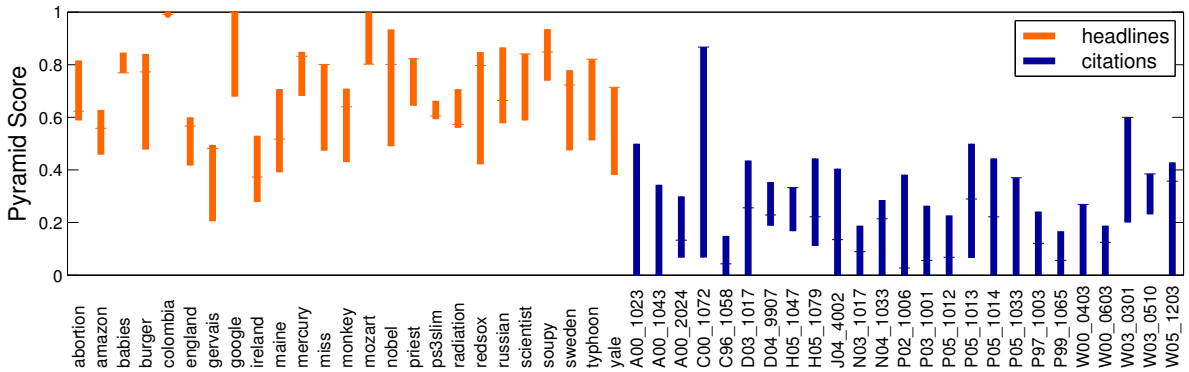


Figure 3: The 25th to 75th percentile pyramid score range in individual clusters

words that is inspired by (Lee, 1999). We represent each word by its context in the cluster and find the similarity of such contexts. Particularly, each word w_i is represented by a bag of words, ℓ_i, that have a surface distance of 3 or smaller to w_i anywhere in the cluster. In other words, ℓ_i contains any word that co-occurs with w_i in a 4-gram in the cluster. This *bag of words* representation of words enables us to find the word-pair similarities.

$$sim(w_i, w_j) = \frac{\vec{\ell_i} \cdot \vec{\ell_j}}{\sqrt{|\vec{\ell_i}||\vec{\ell_j}|}} \quad (1)$$

We use the pair-wise similarities of words in each cluster, and build a network of words and their similarities. Intuitively, words that appear in similar contexts are more similar to each other and will have a stronger edge between them in the network. Therefore, similar words, or words that appear in similar contexts, will form communities in this graph. Ideally, each community in the word similarity network would represent a factoid. To find the communities in the word network we use (Clauset et al., 2004), a hierarchical agglomeration algorithm which works by greedily optimizing the modularity in a linear running time for sparse graphs.

The community detection algorithm will assign to each word w_i, a community label C_i. For each community, we use LexRank to rank the words using the similarities in Equation 1, and assign a score to each word w_i as $S(w_i) = \frac{R_i}{|C_i|}$, where R_i is the rank of w_i in its community, and $|C_i|$ is the number of words that belong to C_i. Figure 5.1 shows part

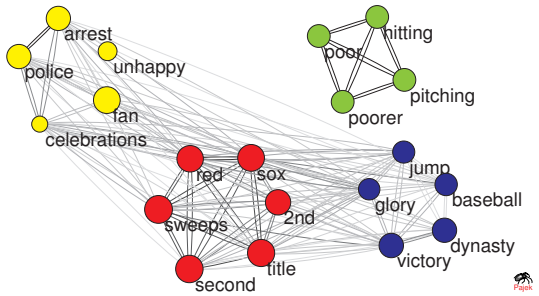


Figure 5: Part of the word similarity graph in the redsox cluster

of the word similarity graph in the `redsox` cluster, in which each node is color-coded with its community. This figure illustrates how words that are semantically related to the same aspects of the story fall in the same communities (e.g., "police" and "arrest"). Finally, to rank sentences, we define the score of each document D_j as the sum of the scores of its words.

$$p_{ds}(D_j) = \sum_{w_i \in D_j} S(w_i)$$

Intuitively, sentences that contain higher ranked words in highly populated communities will have a smaller score. To rank the sentences, we sort them in an ascending order, and cut the list when its size is greater than the length limit.

5.2 Other Methods

5.2.1 Random

For each cluster in each category (citations and headlines), this method simply gets a random per-

mutations of the summaries. In the headlines datasets, where most of the headlines cover some factoids about the story, we expect this method to perform reasonably well since randomization will increase the chances of covering headlines that focus on different factoids. However, in the citations dataset, where a citing sentence may cover no information about the cited paper, randomization has the drawback of selecting citations that have no valuable information in them.

5.2.2 LexRank

LexRank (Erkan and Radev, 2004) works by first building a graph of all the documents (D_i) in a cluster. The edges between corresponding nodes (d_i) represent the cosine similarity between them is above a threshold (0.10 following (Erkan and Radev, 2004)). Once the network is built, the system finds the most central sentences by performing a random walk on the graph.

$$p(d_j) = (1-\lambda)\frac{1}{|D|} + \lambda \sum_{d_i} p(d_i) P(d_i \rightarrow d_j) \qquad (2)$$

5.2.3 MMR

Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998) uses the pairwise cosine similarity matrix and greedily chooses sentences that are the least similar to those already in the summary. In particular,

$MMR = \arg\min_{D_i \in D-A} \left[\max_{D_j \in A} Sim(D_i, D_j) \right]$

where A is the set of documents in the summary, initialized to $A = \emptyset$.

5.2.4 DivRank

Unlike other time-homogeneous random walks (e.g., PageRank), DivRank does not assume that the transition probabilities remain constant over time. DivRank uses a *vertex-reinforced random walk* model to rank graph nodes based on a diversity based centrality. The basic assumption in DivRank is that the transition probability from a node to other is reinforced by the number of previous visits to the target node (Mei et al., 2010). Particularly, let's assume $p_T(u, v)$ is the transition probability from any node u to node v at time T. Then,

$$p_T(d_i, d_j) = (1-\lambda).p^*(d_j) + \lambda.\frac{p_0(d_i, d_j).N_T(d_j)}{D_T(d_i)} \qquad (3)$$

where $N_T(d_j)$ is the number of times the walk has visited d_j up to time T and

$$D_T(d_i) = \sum_{d_j \in V} p_0(d_i, d_j) N_T(d_j) \qquad (4)$$

Here, $p^*(d_j)$ is the prior distribution that determines the preference of visiting vertex d_j. We try two variants of this algorithm: **DivRank**, in which $p^*(d_j)$ is uniform, and **DivRank with priors** in which $p^*(d_j) \propto l(D_j)^{-\beta}$, where $l(D_j)$ is the number of the words in the document D_j and β is a parameter ($\beta = 0.8$).

5.2.5 C-LexRank

C-LexRank is a clustering-based model in which the cosine similarities of document pairs are used to build a network of documents. Then the the network is split into communities, and the most salient documents in each community are selected (Qazvinian and Radev, 2008). C-LexRank focuses on finding communities of documents using their cosine similarity. The intuition is that documents that are more similar to each other contain similar factoids. We expect C-LexRank to be a strong ranker, but incapable of capturing the diversity caused by using different phrases to express the same meaning. The reason is that different nuggets that represent the same factoid often have no words in common (e.g., "victory" and "glory") and won't be captured by a lexical measure like cosine similarity.

5.3 Experiments

We use each of the systems explained above to rank the summaries in each cluster. Each ranked list is then cut at a certain length (50 words for headlines, and 150 for citations) and the information content in the remaining text is examined using the pyramid score.

Table 3 shows the average pyramid score achieved by different methods in each category. The method based on the distributional similarities of words outperforms other methods in the citations category. All methods show similar results in the headlines category, where most headlines cover at least 1 factoid about the story and a random ranker performs reasonably well. Table 4 shows top 3 headlines from 3 rankers: word distributional similarity (WDS), C-LexRank, and MMR. In this example, the first 3

Method	headlines		citations		Mean
	pyramid	95% C.I.	pyramid	95% C.I.	
R	0.928	[0.896, 0.959]	0.716	[0.625, 0.807]	0.822
MMR	0.930	[0.902, 0.960]	0.766	[0.684, 0.847]	0.848
LR	0.918	[0.891, 0.945]	0.728	[0.635, 0.822]	0.823
DR	0.927	[0.900, 0.955]	0.736	[0.667, 0.804]	0.832
DR(p)	0.916	[0.884, 0.949]	0.764	[0.697, 0.831]	0.840
C-LR	0.942	[0.919, 0.965]	0.781	[0.710, 0.852]	0.862
WDS	0.931	[0.905, 0.958]	0.813	[0.738, 0.887]	**0.872**

R=Random; LR=LexRank; DR=DivRank; DR(p)=DivRank with Priors; C-LR=C-LexRank; WDS=Word Distributional Similarity; C.I.=Confidence Interval

Table 3: Comparison of different ranking systems

Method	Top 3 headlines
WDS	1: how sweep it is 2: fans celebrate red sox win 3: red sox take title
C-LR	1: world series: red sox sweep rockies 2: red sox take world series 3: red sox win world series
MMR	1:red sox scale the rockies 2: boston sweep colorado to win world series 3: rookies respond in first crack at the big time

C-LR=C-LexRank; WDS=Word Distributional Similarity

Table 4: Top 3 ranked summaries of the redsox cluster using different methods

headlines produced by WDS cover two important factoids: "red sox winning the title" and "fans celebrating". However, the second factoid is absent in the other two.

6 Conclusion and Future Work

Our experiments on two different categories of human-written summaries (headlines and citations) showed that a lot of the diversity seen in human summarization comes from different nuggets that may actually represent the same semantic information (i.e., factoids). We showed that the factoids exhibit a skewed distribution model, and that the size of the nugget inventory asymptotic behavior even with a large number of summaries. We also showed high variation in summary quality across different summaries in terms of pyramid score, and that the information covered by reading n summaries has a rapidly growing asymptotic behavior as n increases. Finally, we proposed a ranking system that employs word distributional similarities to identify semantically equivalent words, and compared it with a wide range of summarization systems that leverage diversity.

In the future, we plan to move to content from other collective systems on Web. In order to generalize our findings, we plan to examine blog comments, online reviews, and tweets (that discuss the same URL). We also plan to build a generation system that employs the Yule model (Yule, 1925) to determine the importance of each aspect (e.g. who, when, where, etc.) in order to produce summaries that include diverse aspects of a story.

Our work has resulted in a publicly available dataset [8] of 25 annotated news clusters with nearly $1,400$ headlines, and 25 clusters of citation sentences with more than 900 citations. We believe that this dataset can open new dimensions in studying diversity and other aspects of automatic text generation.

7 Acknowledgments

This work is supported by the National Science Foundation grant number IIS-0705832 and grant number IIS-0968489. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the supporters.

References

Eytan Adar, Li Zhang, Lada A. Adamic, and Rajan M. Lukose. 2004. Implicit structure and the dynamics of

[8] http://www-personal.umich.edu/~vahed/data.html

Blogspace. In *WWW'04, Workshop on the Weblogging Ecosystem*.

Eytan Adar, Daniel S. Weld, Brian N. Bershad, and Steven S. Gribble. 2007. Why we search: visualizing and predicting user behavior. In *WWW'07*, pages 161–170, New York, NY, USA.

Regina Barzilay and Lillian Lee. 2002. Bootstrapping lexical choice via multiple-sequence alignment. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 164–171.

Regina Barzilay and Kathleen R. McKeown. 2005. Sentence fusion for multidocument news summarization. *Comput. Linguist.*, 31(3):297–328.

Herbert Blumer. 1951. Collective behavior. *In Lee, Alfred McClung, Ed., Principles of Sociology.*

Jaime G. Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR'98*, pages 335–336.

Jean Carletta. 1996. Assessing agreement on classification tasks: the kappa statistic. *Comput. Linguist.*, 22(2):249–254.

Aaron Clauset, Mark E. J. Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Phys. Rev. E*, 70(6).

Michael Elhadad. 1995. Using argumentation in text generation. *Journal of Pragmatics*, 24:189–220.

Güneş Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research (JAIR)*.

Len Fisher. 2009. *The Perfect Swarm: The Science of Complexity in Everyday Life*. Basic Books.

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Comput. Linguist.*, 12:175–204, July.

Lu Hong and Scott Page. 2009. Interpreted and generated signals. *Journal of Economic Theory*, 144(5):2174–2196.

Akshay Java, Pranam Kolari, Tim Finin, and Tim Oates. 2006. Modeling the spread of influence on the blogosphere. In *WWW'06*.

Klaus Krippendorff. 1980. *Content Analysis: An Introduction to its Methodology*. Beverly Hills: Sage Publications.

Ravi Kumar, Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. 2003. On the bursty evolution of blogspace. In *WWW'03*, pages 568–576, New York, NY, USA.

Lillian Lee. 1999. Measures of distributional similarity. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 25–32.

Jure Leskovec, Lars Backstrom, and Jon Kleinberg. 2009. Meme-tracking and the dynamics of the news cycle. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 497–506.

Chin-Yew Lin and Eduard Hovy. 2002. Manual and automatic evaluation of summaries. In *ACL-Workshop on Automatic Summarization*.

Qiaozhu Mei, Jian Guo, and Dragomir Radev. 2010. Divrank: the interplay of prestige and diversity in information networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1009–1018.

Gilad Mishne and Natalie Glance. 2006. Predicting movie sales from blogger sentiment. In *AAAI 2006 Spring Symposium on Computational Approaches to Analysing Weblogs (AAAI-CAAW 2006)*.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. *Proceedings of the HLT-NAACL conference*.

Scott E. Page. 2007. *The Difference: How the Power of Diversity Creates Better Groups, Firms, Schools, and Societies*. Princeton University Press.

Bo Pang and Lillian Lee. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL'04*, Morristown, NJ, USA.

Michael Paul, ChengXiang Zhai, and Roxana Girju. 2010. Summarizing contrastive viewpoints in opinionated text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 66–76.

Vahed Qazvinian and Dragomir R. Radev. 2008. Scientific paper summarization using citation summary networks. In *COLING 2008*, Manchester, UK.

Vahed Qazvinian and Dragomir R. Radev. 2010. Identifying non-explicit citing sentences for citation-based summarization. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 555–564, Uppsala, Sweden, July. Association for Computational Linguistics.

Neil J. Smelser. 1963. *Theory of Collective Behavior*. Free Press.

Karen Spärck-Jones. 1999. Automatic summarizing: factors and directions. In Inderjeet Mani and Mark T. Maybury, editors, *Advances in automatic text summarization*, chapter 1, pages 1 – 12. The MIT Press.

Manfred Stede. 1995. Lexicalization in natural language generation: a survey. *Artificial Intelligence Review*, (8):309–336.

Hans van Halteren and Simone Teufel. 2003. Examining the consensus between human summaries: initial experiments with factoid analysis. In *Proceedings of*

the HLT-NAACL 03 on Text summarization workshop, pages 57–64, Morristown, NJ, USA. Association for Computational Linguistics.

Hans van Halteren and Simone Teufel. 2004. Evaluating information content by factoid analysis: human annotation and stability. In *EMNLP'04*, Barcelona.

Ellen M. Voorhees. 1998. Variations in relevance judgments and the measurement of retrieval effectiveness. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 315–323.

G. Udny Yule. 1925. A mathematical theory of evolution, based on the conclusions of dr. j. c. willis, f.r.s. *Philosophical Transactions of the Royal Society of London. Series B, Containing Papers of a Biological Character*, 213:21–87.

Xiaojin Zhu, Andrew Goldberg, Jurgen Van Gael, and David Andrzejewski. 2007. Improving diversity in ranking using absorbing random walks. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 97–104.

Ordering Prenominal Modifiers with a Reranking Approach

Jenny Liu
MIT CSAIL
jyliu@csail.mit.edu

Aria Haghighi
MIT CSAIL
me@aria42.com

Abstract

In this work, we present a novel approach to the generation task of ordering prenominal modifiers. We take a maximum entropy reranking approach to the problem which admits arbitrary features on a permutation of modifiers, exploiting hundreds of thousands of features in total. We compare our error rates to the state-of-the-art and to a strong Google n-gram count baseline. We attain a maximum error reduction of 69.8% and average error reduction across all test sets of 59.1% compared to the state-of-the-art and a maximum error reduction of 68.4% and average error reduction across all test sets of 41.8% compared to our Google n-gram count baseline.

1 Introduction

Speakers rarely have difficulty correctly ordering modifiers such as adjectives, adverbs, or gerunds when describing some noun. The phrase "beautiful blue Macedonian vase" sounds very natural, whereas changing the modifier ordering to "blue Macedonian beautiful vase" is awkward (see Table 1 for more examples). In this work, we consider the task of ordering an unordered set of prenominal modifiers so that they sound fluent to native language speakers. This is an important task for natural language generation systems.

Much linguistic research has investigated the semantic constraints behind prenominal modifier orderings. One common line of research suggests that modifiers can be organized by the underlying semantic property they describe and that there is an ordering on semantic properties which in turn restricts modifier orderings. For instance, Sproat and Shih (1991) contend that the *size* property precedes the *color* property and thus "small black cat" sounds more fluent than "black small cat". Using $>$ to denote precedence of semantic groups, some commonly proposed orderings are: *quality* $>$ *size* $>$ *shape* $>$ *color* $>$ *provenance* (Sproat and Shih, 1991), *age* $>$ *color* $>$ *participle* $>$ *provenance* $>$ *noun* $>$ *denominal* (Quirk et al., 1974), and *value* $>$ *dimension* $>$ *physical property* $>$ *speed* $>$ *human propensity* $>$ *age* $>$ *color* (Dixon, 1977). However, correctly classifying modifiers into these groups can be difficult and may be domain dependent or constrained by the context in which the modifier is being used. In addition, these methods do not specify how to order modifiers within the same class or modifiers that do not fit into any of the specified groups.

a. the vegetarian French lawyer
b. the French vegetarian lawyer
a. the beautiful small black purse
b. the beautiful black small purse
c. the small beautiful black purse
d. the small black beautiful purse

Table 1: Examples of restrictions on modifier orderings from Teodorescu (2006). The most natural sounding ordering is in bold, followed by other possibilities that may only be appropriate in certain situations.

There have also been a variety of corpus-based, computational approaches. Mitchell (2009) uses

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1109–1116,
Portland, Oregon, June 19-24, 2011.

a class-based approach in which modifiers are grouped into classes based on which positions they prefer in the training corpus, with a predefined ordering imposed on these classes. Shaw and Hatzivassiloglou (1999) developed three different approaches to the problem that use counting methods and clustering algorithms, and Malouf (2000) expands upon Shaw and Hatzivassiloglou's work.

This paper describes a computational solution to the problem that uses relevant features to model the modifier ordering process. By mapping a set of features across the training data and using a maximum entropy reranking model, we can learn optimal weights for these features and then order each set of modifiers in the test data according to our features and the learned weights. This approach has not been used before to solve the prenominal modifier ordering problem, and as we demonstrate, vastly outperforms the state-of-the-art, especially for sequences of longer lengths.

Section 2 of this paper describes previous computational approaches. In Section 3 we present the details of our maximum entropy reranking approach. Section 4 covers the evaluation methods we used, and Section 5 presents our results. In Section 6 we compare our approach to previous methods, and in Section 7 we discuss future work and improvements that could be made to our system.

2 Related Work

Mitchell (2009) orders sequences of at most 4 modifiers and defines nine classes that express the broad positional preferences of modifiers, where position 1 is closest to the noun phrase (NP) head and position 4 is farthest from it. Classes 1 through 4 comprise those modifiers that prefer only to be in positions 1 through 4, respectively. Class 5 through 7 modifiers prefer positions 1-2, 2-3, and 3-4, respectively, while class 8 modifiers prefer positions 1-3, and finally, class 9 modifiers prefer positions 2-4. Mitchell counts how often each word type appears in each of these positions in the training corpus. If any modifier's probability of taking a certain position is greater than a uniform distribution would allow, then it is said to prefer that position. Each word type is then assigned a class, with a global ordering defined over the nine classes.

Given a set of modifiers to order, if the entire set has been seen at training time, Mitchell's system looks up the class of each modifier and then orders the sequence based on the predefined ordering for the classes. When two modifiers have the same class, the system picks between the possibilities randomly. If a modifier was not seen at training time and thus cannot be said to belong to a specific class, the system favors orderings where modifiers whose classes are known are as close to their classes' preferred positions as possible.

Shaw and Hatzivassiloglou (1999) use corpus-based counting methods as well. For a corpus with w word types, they define a $w \times w$ matrix where $Count[A, B]$ indicates how often modifier A precedes modifier B. Given two modifiers a and b to order, they compare $Count[a, b]$ and $Count[b, a]$ in their training data. Assuming a null hypothesis that the probability of either ordering is 0.5, they use a binomial distribution to compute the probability of seeing the ordering $< a, b >$ for $Count[a, b]$ number of times. If this probability is above a certain threshold then they say that a precedes b. Shaw and Hatzivassiloglou also use a transitivity method to fill out parts of the *Count* table where bigrams are not actually seen in the training data but their counts can be inferred from other entries in the table, and they use a clustering method to group together modifiers with similar positional preferences.

These methods have proven to work well, but they also suffer from sparsity issues in the training data. Mitchell reports a prediction accuracy of 78.59% for NPs of all lengths, but the accuracy of her approach is greatly reduced when two modifiers fall into the same class, since the system cannot make an informed decision in those cases. In addition, if a modifier is not seen in the training data, the system is unable to assign it a class, which also limits accuracy. Shaw and Hatzivassiloglou report a highest accuracy of 94.93% and a lowest accuracy of 65.93%, but since their methods depend heavily on bigram counts in the training corpus, they are also limited in how informed their decisions can be if modifiers in the test data are not present at training time.

In this next section, we describe our maximum entropy reranking approach that tries to develop a more comprehensive model of the modifier ordering process to avoid the sparsity issues that previous ap-

proaches have faced.

3 Model

We treat the problem of prenominal modifier ordering as a reranking problem. Given a set B of prenominal modifiers and a noun phrase head H which B modifies, we define $\pi(B)$ to be the set of all possible permutations, or orderings, of B. We suppose that for a set B there is some $x^* \in \pi(B)$ which represents a "correct" natural-sounding ordering of the modifiers in B.

At test time, we choose an ordering $x \in \pi(B)$ using a maximum entropy reranking approach (Collins and Koo, 2005). Our distribution over orderings $x \in \pi(B)$ is given by:

$$P(x|H, B, W) = \frac{\exp\{W^T \phi(B, H, x)\}}{\sum_{x' \in \pi(B)} \exp\{W^T \phi(B, H, x')\}}$$

where $\phi(B, H, x)$ is a feature vector over a particular ordering of B and W is a learned weight vector over features. We describe the set of features in section 3.1, but note that we are free under this formulation to use arbitrary features on the full ordering x of B as well as the head noun H, which we implicitly condition on throughout. Since the size of the set of prenominal modifiers B is typically less than six, enumerating $\pi(B)$ is not expensive.

At training time, our data consists of sequences of prenominal orderings and their corresponding nominal heads. We treat each sequence as a training example where the labeled ordering $x^* \in \pi(B)$ is the one we observe. This allows us to extract any number of 'labeled' examples from part-of-speech text. Concretely, at training time, we select W to maximize:

$$\mathcal{L}(W) = \left(\prod_{(B,H,x^*)} P(x^*|H, B, W) \right) - \frac{\|W\|^2}{2\sigma^2}$$

where the first term represents our observed data likelihood and the second the ℓ_2 regularization, where σ^2 is a fixed hyperparameter; we fix the value of σ^2 to 0.5 throughout. We optimize this objective using standard L-BFGS optimization techniques.

The key to the success of our approach is using the flexibility afforded by having arbitrary features $\phi(B, H, x)$ to capture all the salient elements of the prenominal ordering data. These features can be used to create a richer model of the modifier ordering process than previous corpus-based counting approaches. In addition, we can encapsulate previous approaches in terms of features in our model. Mitchell's class-based approach can be expressed as a binary feature that tells us whether a given permuation satisfies the class ordering constraints in her model. Previous counting approaches can be expressed as a real-valued feature that, given all n-grams generated by a permutation of modifiers, returns the count of all these n-grams in the original training data.

3.1 Feature Selection

Our features are of the form $\phi(B, H, x)$ as expressed in the model above, and we include both indicator features and real-valued numeric features in our model. We attempt to capture aspects of the modifier permutations that may be significant in the ordering process. For instance, perhaps the majority of words that end with *-ly* are adverbs and should usually be positioned farthest from the head noun, so we can define an indicator function that captures this feature as follows:

$$\phi(B, H, x) = \begin{cases} 1 & \text{if the modifier in position } i \\ & \text{of ordering } x \text{ ends in -}ly \\ 0 & \text{otherwise} \end{cases}$$

We create a feature of this form for every possible modifier position i from 1 to 4.

We might also expect permutations that contain n-grams previously seen in the training data to be more natural sounding than other permutations that generate n-grams that have not been seen before. We can express this as a real-valued feature:

$$\phi(B, H, x) = \begin{cases} \text{count in training data of all} \\ n\text{-grams present in } x \end{cases}$$

See Table 2 for a summary of our features. Many of the features we use are similar to those in Dunlop et al. (2010), which uses a feature-based multiple sequence alignment approach to order modifiers.

Numeric Features	
n-gram Count	If N is the set of all n-grams present in the permutation, returns the sum of the counts of each element of N in the training data. A separate feature is created for 2-gms through 5-gms.
Count of Head Noun and Closest Modifier	Returns the count of $< M, H >$ in the training data where H is the head noun and M is the modifier closest to H.
Length of Modifier*	Returns the length of modifier in position i
Indicator Features	
Hyphenated*	Modifier in position i contains a hyphen.
Is Word *w**	Modifier in position i is word $w \epsilon W$, where W is the set of all word types in the training data.
Ends In *e**	Modifier in position i ends in suffix $e \epsilon E$, where E = {*-al -ble -ed -er -est -ic -ing -ive -ly -ian*}
Is A Color*	Modifier in position i is a color, where we use a list of common colors
Starts With a Number*	Modifier in position i starts with a number
Is a Number*	Modifier in position i is a number
Satisfies Mitchell Class Ordering	The permutation's class ordering satisfies the Mitchell class ordering constraints

Table 2: Features Used In Our Model. Features with an asterisk (*) are created for all possible modifier positions i from 1 to 4.

4 Experiments

4.1 Data Preprocessing and Selection

We extracted all noun phrases from four corpora: the Brown, Switchboard, and Wall Street Journal corpora from the Penn Treebank, and the North American Newswire corpus (NANC). Since there were very few NPs with more than 5 modifiers, we kept those with 2-5 modifiers and with tags *NN* or *NNS* for the head noun. We also kept NPs with only 1 modifier to be used for generating <*modifier, head noun*> bigram counts at training time. We then filtered all these NPs as follows: If the NP contained a *PRP*, *IN*, *CD*, or *DT* tag and the corresponding modifier was farthest away from the head noun, we removed this modifier and kept the rest of the NP. If the modifier was not the farthest away from the head noun, we discarded the NP. If the NP contained a *POS* tag we only kept the part of the phrase up to this tag. Our final set of NPs had tags from the following list: *JJ*, *NN*, *NNP*, *NNS*, *JJS*, *JJR*, *VBG*, *VBN*, *RB*, *NNPS*, *RBS*. See Table 3 for a summary of the number of NPs of lengths 1-5 extracted from the four corpora.

Our system makes several passes over the data during the training process. In the first pass, we collect statistics about the data, to be used later on when calculating our numeric features. To collect the statistics, we take each NP in the training data and consider all possible 2-gms through 5-gms that are present in the NP's modifier sequence, allowing for non-consecutive n-grams. For example, the NP "the beautiful blue Macedonian vase" generates the following bigrams: <*beautiful blue*>, <*blue Macedonian*>, and <*beautiful Macedonian*>, along with the 3-gram <*beautiful blue Macedonian*>. We keep a table mapping each unique n-gram to the number of times it has been seen in the training data. In addition, we also store a table that keeps track of bigram counts for $< M, H >$, where H is the head noun of an NP and M is the modifier closest to it. In the example "the beautiful blue Macedonian vase," we would increment the count of $< Macedonian, vase >$ in the table. The n-gram and $< M, H >$ counts are used to compute numeric fea-

Number of Sequences (Token)

	1	2	3	4	5	Total
Brown	11,265	1,398	92	8	2	12,765
WSJ	36,313	9,073	1,399	229	156	47,170
Switchboard	10,325	1,170	114	4	1	11,614
NANC	15,456,670	3,399,882	543,894	80,447	14,840	19,495,733

Number of Sequences (Type)

	1	2	3	4	5	Total
Brown	4,071	1,336	91	8	2	5,508
WSJ	7,177	6,687	1,205	182	42	15,293
Switchboard	2,122	950	113	4	1	3,190
NANC	241,965	876,144	264,503	48,060	8,451	1,439,123

Table 3: Number of NPs extracted from our data for NP sequences with 1 to 5 modifiers.

ture values.

4.2 Google n-gram Baseline

The Google n-gram corpus is a collection of n-gram counts drawn from public webpages with a total of one trillion tokens – around 1 billion each of unique 3-grams, 4-grams, and 5-grams, and around 300,000 unique bigrams. We created a Google n-gram baseline that takes a set of modifiers B, determines the Google n-gram count for each possible permutation in $\pi(B)$, and selects the permutation with the highest n-gram count as the winning ordering x^*. We will refer to this baseline as GOOGLE N-GRAM.

4.3 Mitchell's Class-Based Ordering of Prenominal Modifiers (2009)

Mitchell's original system was evaluated using only three corpora for both training and testing data: Brown, Switchboard, and WSJ. In addition, the evaluation presented by Mitchell's work considers a prediction to be correct if the ordering of classes in that prediction is the same as the ordering of classes in the original test data sequence, where a class refers to the positional preference groupings defined in the model. We use a more stringent evaluation as described in the next section.

We implemented our own version of Mitchell's system that duplicates the model and methods but allows us to scale up to a larger training set and to apply our own evaluation techniques. We will refer to this baseline as CLASS BASED.

4.4 Evaluation

To evaluate our system (MAXENT) and our baselines, we partitioned the corpora into training and testing data. For each NP in the test data, we generated a set of modifiers and looked at the predicted orderings of the MAXENT, CLASS BASED, and GOOGLE N-GRAM methods. We considered a predicted sequence ordering to be correct if it matches the original ordering of the modifiers in the corpus. We ran four trials, the first holding out the Brown corpus and using it as the test set, the second holding out the WSJ corpus, the third holding out the Switchboard corpus, and the fourth holding out a randomly selected tenth of the NANC. For each trial we used the rest of the data as our training set.

5 Results

The MAXENT model consistently outperforms CLASS BASED across all test corpora and sequence lengths for both tokens and types, except when testing on the Brown and Switchboard corpora for modifier sequences of length 5, for which neither approach is able to make any correct predictions. However, there are only 3 sequences total of length 5 in the Brown and Swichboard corpora combined.

Test Corpus		Token Accuracy (%)					Type Accuracy (%)				
		2	**3**	**4**	**5**	**Total**	**2**	**3**	**4**	**5**	**Total**
Brown	GOOGLE N-GRAM	82.4	35.9	12.5	0	79.1	81.8	36.3	12.5	0	78.4
	CLASS BASED	79.3	54.3	25.0	0	77.3	78.9	54.9	25.0	0	77.0
	MAXENT	**89.4**	**70.7**	**87.5**	0	**88.1**	**89.1**	**70.3**	**87.5**	0	**87.8**
WSJ	GOOGLE N-GRAM	84.8	53.5	31.4	71.8	79.4	82.6	49.7	23.1	16.7	76.0
	CLASS BASED	85.5	51.6	16.6	0.6	78.5	85.1	50.1	19.2	0	78.0
	MAXENT	**95.9**	**84.1**	**71.2**	**80.1**	**93.5**	**94.7**	**81.9**	**70.3**	**45.2**	**92.0**
Switchboard	GOOGLE N-GRAM	**92.8**	68.4	0	0	**90.3**	**91.7**	68.1	0	0	**88.8**
	CLASS BASED	80.1	52.6	0	0	77.3	79.1	53.1	0	0	75.9
	MAXENT	91.4	**74.6**	**25.0**	0	89.6	90.3	**75.2**	**25.0**	0	88.4
One Tenth of	GOOGLE N-GRAM	86.8	55.8	27.7	43.0	81.1	79.2	44.6	20.5	12.3	70.4
NANC	CLASS BASED	86.1	54.7	20.1	1.9	80.0	80.3	51.0	18.4	3.3	74.5
	MAXENT	**95.2**	**83.8**	**71.6**	**62.2**	**93.0**	**91.6**	**78.8**	**63.8**	**44.4**	**88.0**

Test Corpus	Number of Features Used In MaxEnt Model
Brown	655,536
WSJ	654,473
Switchboard	655,791
NANC	565,905

Table 4: Token and type prediction accuracies for the GOOGLE N-GRAM, MAXENT, and CLASS BASED approaches for modifier sequences of lengths 2-5. Our data consisted of four corpuses: Brown, Switchboard, WSJ, and NANC. The test data was held out and each approach was trained on the rest of the data. Winning scores are in bold. The number of features used during training for the MAXENT approach for each test corpus is also listed.

MAXENT also outperforms the GOOGLE N-GRAM baseline for almost all test corpora and sequence lengths. For the Switchboard test corpus token and type accuracies, the GOOGLE N-GRAM baseline is more accurate than MAXENT for sequences of length 2 and overall, but the accuracy of MAXENT is competitive with that of GOOGLE N-GRAM.

If we examine the error reduction between MAXENT and CLASS BASED, we attain a maximum error reduction of 69.8% for the WSJ test corpus across modifier sequence tokens, and an average error reduction of 59.1% across all test corpora for tokens. MAXENT also attains a maximum error reduction of 68.4% for the WSJ test corpus and an average error reduction of 41.8% when compared to GOOGLE N-GRAM.

It should also be noted that on average the MAXENT model takes three hours to train with several hundred thousand features mapped across the training data (the exact number used during each test run is listed in Table 4) – this tradeoff is well worth the increase we attain in system performance.

6 Analysis

MAXENT seems to outperform the CLASS BASED baseline because it learns more from the training data. The CLASS BASED model classifies each modifier in the training data into one of nine broad categories, with each category representing a different set of positional preferences. However, many of the modifiers in the training data get classified to the same category, and CLASS BASED makes a random choice when faced with orderings of modifiers all in the same category. When applying CLASS BASED

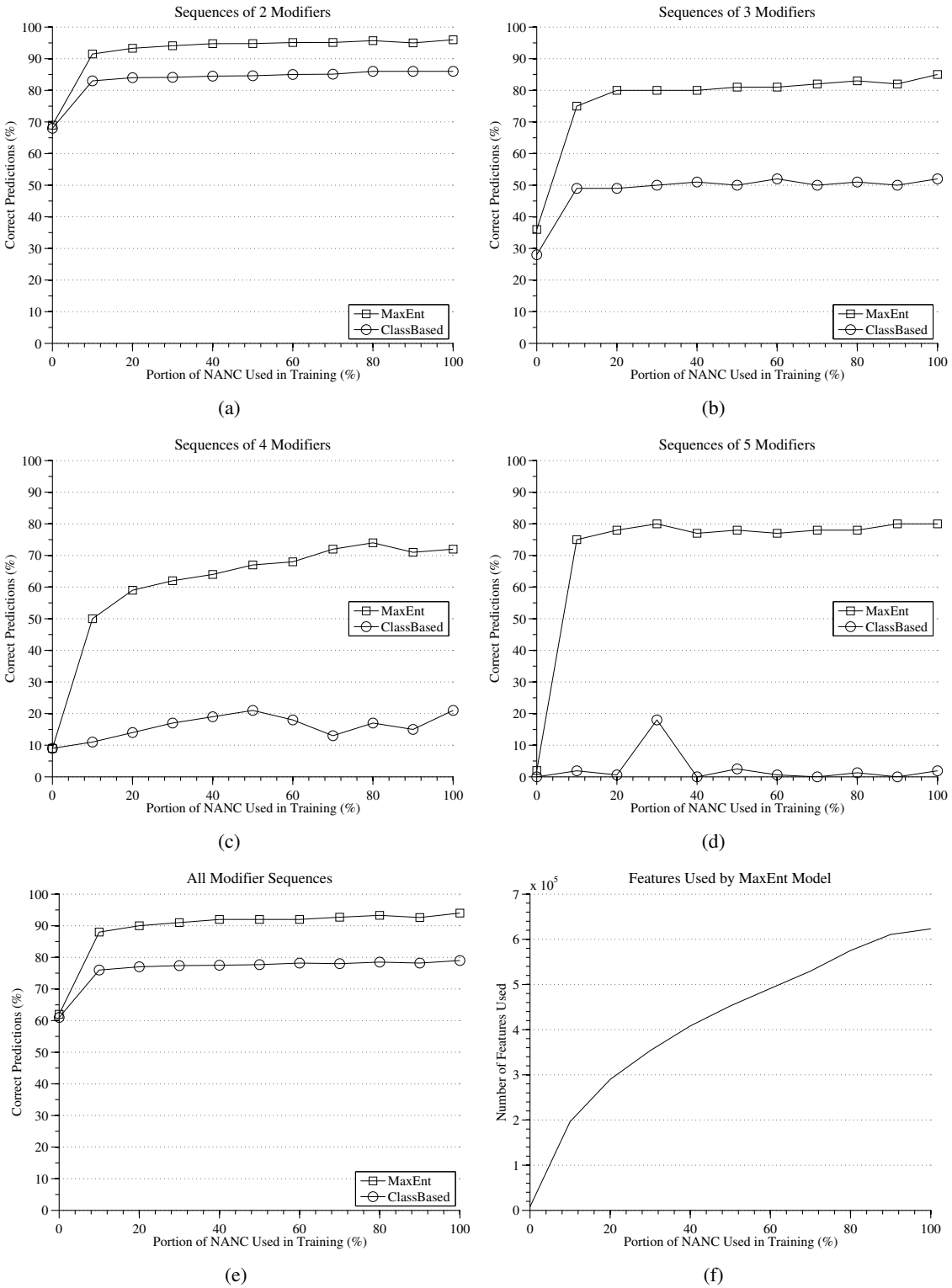


Figure 1: Learning curves for the MAXENT and CLASS BASED approaches. We start by training each approach on just the Brown and Switchboard corpora while testing on WSJ. We incrementally add portions of the NANC corpus. Graphs (a) through (d) break down the total correct predictions by the number of modifiers in a sequence, while graph (e) gives accuracies over modifier sequences of all lengths. Prediction percentages are for sequence tokens. Graph (f) shows the number of features active in the MaxEnt model as the training data scales up.

to WSJ as the test data and training on the other corpora, 74.7% of the incorrect predictions contained at least 2 modifiers that were of the same positional preferences class. In contrast, MAXENT allows us to learn much more from the training data. As a result, we see much higher numbers when trained and tested on the same data as CLASS BASED.

The GOOGLE N-GRAM method does better than the CLASS BASED approach because it contains n-gram counts for more data than the WSJ, Brown, Switchboard, and NANC corpora combined. However, GOOGLE N-GRAM suffers from sparsity issues as well when testing on less common modifier combinations. For example, our data contains rarely heard sequences such as "Italian, state-owned, holding company" or "armed Namibian nationalist guerrillas." While MAXENT determines the correct ordering for both of these examples, none of the permutations of either example show up in the Google n-gram corpus, so the GOOGLE N-GRAM method is forced to randomly select from the six possibilities. In addition, the Google n-gram corpus is composed of sentence fragments that may not necessarily be NPs, so we may be overcounting certain modifier permutations that can function as different parts of a sentence.

We also compared the effect that increasing the amount of training data has when using the CLASS BASED and MAXENT methods by initially training each system with just the Brown and Switchboard corpora and testing on WSJ. Then we incrementally added portions of NANC, one tenth at a time, until the training set included all of it. The results (see Figure 1) show that we are able to benefit from the additional data much more than the CLASS BASED approach can, since we do not have a fixed set of classes limiting the amount of information the model can learn. In addition, adding the first tenth of NANC made the biggest difference in increasing accuracy for both approaches.

7 Conclusion

The straightforward maximum entropy reranking approach is able to significantly outperform previous computational approaches by allowing for a richer model of the prenominal modifier ordering process. Future work could include adding more features to the model and conducting ablation testing. In addition, while many sets of modifiers have stringent ordering requirements, some variations on orderings, such as "former famous actor" vs. "famous former actor," are acceptable in both forms and have different meanings. It may be beneficial to extend the model to discover these ambiguities.

Acknowledgements

Many thanks to Margaret Mitchell, Regina Barzilay, Xiao Chen, and members of the CSAIL NLP group for their help and suggestions.

References

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

R. M. W. Dixon. 1977. Where Have all the Adjectives Gone? *Studies in Language*, 1(1):19–80.

A. Dunlop, M. Mitchell, and B. Roark. 2010. Prenominal modifier ordering via multiple sequence alignment. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 600–608. Association for Computational Linguistics.

R. Malouf. 2000. The order of prenominal adjectives in natural language generation. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 85–92. Association for Computational Linguistics.

M. Mitchell. 2009. Class-based ordering of prenominal modifiers. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 50–57. Association for Computational Linguistics.

R. Quirk, S. Greenbaum, R.A. Close, and R. Quirk. 1974. *A university grammar of English*, volume 1985. Longman London.

J. Shaw and V. Hatzivassiloglou. 1999. Ordering among premodifiers. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 135–143. Association for Computational Linguistics.

R. Sproat and C. Shih. 1991. The cross-linguistic distribution of adjective ordering restrictions. *Interdisciplinary approaches to language*, pages 565–593.

A. Teodorescu. 2006. Adjective Ordering Restrictions Revisited. In *Proceedings of the 25th West Coast Conference on Formal Linguistics*, pages 399–407. West Coast Conference on Formal Linguistics.

Unsupervised Semantic Role Induction via Split-Merge Clustering

Joel Lang and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB, UK
J.Lang-3@sms.ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

In this paper we describe an unsupervised method for semantic role induction which holds promise for relieving the data acquisition bottleneck associated with supervised role labelers. We present an algorithm that iteratively splits and merges clusters representing semantic roles, thereby leading from an initial clustering to a final clustering of better quality. The method is simple, surprisingly effective, and allows to integrate linguistic knowledge transparently. By combining role induction with a rule-based component for argument identification we obtain an unsupervised end-to-end semantic role labeling system. Evaluation on the CoNLL 2008 benchmark dataset demonstrates that our method outperforms competitive unsupervised approaches by a wide margin.

1 Introduction

Recent years have seen increased interest in the *shallow semantic analysis* of natural language text. The term is most commonly used to describe the automatic identification and labeling of the semantic roles conveyed by sentential constituents (Gildea and Jurafsky, 2002). Semantic roles describe the relations that hold between a predicate and its arguments, abstracting over surface syntactic configurations. In the example sentences below. *window* occupies different syntactic positions — it is the object of *broke* in sentences (1a,b), and the subject in (1c) — while bearing the same semantic role, i.e., the physical object affected by the breaking event. Analogously, *rock* is the instrument of *break* both when realized as a prepositional phrase in (1a) and as a subject in (1b).

(1) a. [Joe]$_{A0}$ **broke** the [window]$_{A1}$ with a [rock]$_{A2}$.
b. The [rock]$_{A2}$ **broke** the [window]$_{A1}$.
c. The [window]$_{A1}$ **broke**.

The semantic roles in the examples are labeled in the style of PropBank (Palmer et al., 2005), a broad-coverage human-annotated corpus of semantic roles and their syntactic realizations. Under the PropBank annotation framework (which we will assume throughout this paper) each predicate is associated with a set of core roles (named *A*0, *A*1, *A*2, and so on) whose interpretations are specific to that predicate[1] and a set of adjunct roles (e.g., *location* or *time*) whose interpretation is common across predicates. This type of semantic analysis is admittedly shallow but relatively straightforward to automate and useful for the development of broad coverage, domain-independent language understanding systems. Indeed, the analysis produced by existing semantic role labelers has been shown to benefit a wide spectrum of applications ranging from information extraction (Surdeanu et al., 2003) and question answering (Shen and Lapata, 2007), to machine translation (Wu and Fung, 2009) and summarization (Melli et al., 2005).

Since both argument identification and labeling can be readily modeled as classification tasks, most state-of-the-art systems to date conceptualize se-

[1] More precisely, *A*0 and *A*1 have a common interpretation across predicates as *proto-agent* and *proto-patient* in the sense of Dowty (1991).

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1117–1126,
Portland, Oregon, June 19-24, 2011.

mantic role labeling as a supervised learning problem. Current approaches have high performance — a system will recall around 81% of the arguments correctly and 95% of those will be assigned a correct semantic role (see Màrquez et al. (2008) for details), however only on languages and domains for which large amounts of role-annotated training data are available. For instance, systems trained on PropBank demonstrate a marked decrease in performance (approximately by 10%) when tested on out-of-domain data (Pradhan et al., 2008).

Unfortunately, the reliance on role-annotated data which is expensive and time-consuming to produce for every language and domain, presents a major bottleneck to the widespread application of semantic role labeling. Given the data requirements for supervised systems and the current paucity of such data, unsupervised methods offer a promising alternative. They require no human effort for training thus leading to significant savings in time and resources required for annotating text. And their output can be used in different ways, e.g., as a semantic preprocessing step for applications that require broad coverage understanding or as training material for supervised algorithms.

In this paper we present a simple approach to *unsupervised* semantic role labeling. Following common practice, our system proceeds in two stages. It first identifies the semantic arguments of a predicate and then assigns semantic roles to them. Both stages operate over syntactically analyzed sentences without access to any data annotated with semantic roles. Argument identification is carried out through a small set of linguistically-motivated rules, whereas role induction is treated as a clustering problem. In this setting, the goal is to assign argument instances to clusters such that each cluster contains arguments corresponding to a specific semantic role and each role corresponds to exactly one cluster. We formulate a clustering algorithm that executes a series of *split* and *merge* operations in order to transduce an initial clustering into a final clustering of better quality. Split operations leverage syntactic cues so as to create "pure" clusters that contain arguments of the same role whereas merge operations bring together argument instances of a particular role located in different clusters. We test the effectiveness of our induction method on the CoNLL 2008 benchmark dataset and demonstrate improvements over competitive unsupervised methods by a wide margin.

2 Related Work

As mentioned earlier, much previous work has focused on building supervised SRL systems (Màrquez et al., 2008). A few semi-supervised approaches have been developed within a framework known as *annotation projection*. The idea is to combine labeled and unlabeled data by projecting annotations from a labeled source sentence onto an unlabeled target sentence within the same language (Fürstenau and Lapata, 2009) or across different languages (Padó and Lapata, 2009). Outwith annotation projection, Gordon and Swanson (2007) attempt to increase the coverage of PropBank by leveraging existing labeled data. Rather than annotating new sentences that contain previously unseen verbs, they find syntactically similar verbs and use their annotations as surrogate training data.

Swier and Stevenson (2004) induce role labels with a bootstrapping scheme where the set of labeled instances is iteratively expanded using a classifier trained on previously labeled instances. Their method is unsupervised in that it starts with a dataset containing no role annotations at all. However, it requires significant human effort as it makes use of VerbNet (Kipper et al., 2000) in order to identify the arguments of predicates and make initial role assignments. VerbNet is a broad coverage lexicon organized into verb classes each of which is explicitly associated with argument realization and semantic role specifications.

Abend et al. (2009) propose an algorithm that identifies the arguments of predicates by relying only on part of speech annotations, without, however, assigning semantic roles. In contrast, Lang and Lapata (2010) focus solely on the role induction problem which they formulate as the process of detecting alternations and finding a canonical syntactic form for them. Verbal arguments are then assigned roles, according to their position in this canonical form, since each position references a specific role. Their model extends the logistic classifier with hidden variables and is trained in a manner that makes use of the close relationship between syntactic functions and semantic roles. Grenager and Manning

(2006) propose a directed graphical model which relates a verb, its semantic roles, and their possible syntactic realizations. Latent variables represent the semantic roles of arguments and role induction corresponds to inferring the state of these latent variables.

Our own work also follows the unsupervised learning paradigm. We formulate the induction of semantic roles as a clustering problem and propose a split-merge algorithm which iteratively manipulates clusters representing semantic roles. The motivation behind our approach was to design a conceptually simple system, that allows for the incorporation of linguistic knowledge in a straightforward and transparent manner. For example, arguments occurring in similar syntactic positions are likely to bear the same semantic role and should therefore be grouped together. Analogously, arguments that are lexically similar are likely to represent the same semantic role. We operationalize these notions using a scoring function that quantifies the compatibility between arbitrary cluster pairs. Like Lang and Lapata (2010) and Grenager and Manning (2006) our method operates over syntactically parsed sentences, without, however, making use of any information pertaining to semantic roles (e.g., in form of a lexical resource or manually annotated data). Performing role-semantic analysis without a treebank-trained parser is an interesting research direction, however, we leave this to future work.

3 Learning Setting

We follow the general architecture of supervised semantic role labeling systems. Given a sentence and a designated verb, the SRL task consists of identifying the arguments of the verbal predicate (argument identification) and labeling them with semantic roles (role induction).

In our case neither argument identification nor role induction relies on role-annotated data or other semantic resources although we assume that the input sentences are syntactically analyzed. Our approach is not tied to a specific syntactic representation — both constituent- and dependency-based representations could be used. However, we opted for a dependency-based representation, as it simplifies argument identification considerably and is consistent with the CoNLL 2008 benchmark dataset used for evaluation in our experiments.

Given a dependency parse of a sentence, our system identifies argument instances and assigns them to clusters. Thereafter, argument instances can be labeled with an identifier corresponding to the cluster they have been assigned to, similar to PropBank core labels (e.g., *A*0, *A*1).

4 Argument Identification

In the supervised setting, a classifier is employed in order to decide for each node in the parse tree whether it represents a semantic argument or not. Nodes classified as arguments are then assigned a semantic role. In the unsupervised setting, we slightly reformulate argument identification as the task of discarding as many non-semantic arguments as possible. This means that the argument identification component does not make a final positive decision for any of the argument candidates; instead, this decision is deferred to role induction. The rules given in Table 1 are used to discard or select argument candidates. They primarily take into account the parts of speech and the syntactic relations encountered when traversing the dependency tree from predicate to argument. For each candidate, the first matching rule is applied.

We will exemplify how the argument identification component works for the predicate *expect* in the sentence "*The company said it expects its sales to remain steady*" whose parse tree is shown in Figure 1. Initially, all words save the predicate itself are treated as argument candidates. Then, the rules from Table 1 are applied as follows. Firstly, words *the* and *to* are discarded based on their part of speech (rule (1)); then, *remain* is discarded because the path ends with the relation IM and *said* is discarded as the path ends with an upward-leading OBJ relation (rule (2)). Rule (3) does not match and is therefore not applied. Next, *steady* is discarded because there is a downward-leading OPRD relation along the path and the words *company* and *its* are discarded because of the OBJ relations along the path (rule (4)). Rule (5) does not apply but words *it* and *sales* are kept as likely arguments (rule (6)). Finally, rule (7) does not apply, because there are no candidates left.

1. Discard a candidate if it is a determiner, infinitival marker, coordinating conjunction, or punctuation.
2. Discard a candidate if the path of relations from predicate to candidate ends with coordination, subordination, etc. (see the Appendix for the full list of relations).
3. Keep a candidate if it is the closest subject (governed by the subject-relation) to the left of a predicate and the relations from predicate p to the governor g of the candidate are all upward-leading (directed as $g \rightarrow p$).
4. Discard a candidate if the path between the predicate and the candidate, excluding the last relation, contains a subject relation, adjectival modifier relation, etc. (see the Appendix for the full list of relations).
5. Discard a candidate if it is an auxiliary verb.
6. Keep a candidate if the predicate is its parent.
7. Keep a candidate if the path from predicate to candidate leads along several verbal nodes (verb chain) and ends with arbitrary relation.
8. Discard all remaining candidates.

Table 1: Argument identification rules.

5 Split-Merge Role Induction

We treat role induction as a clustering problem with the goal of assigning argument instances (i.e., specific arguments occurring in an input sentence) to clusters such that these represent semantic roles. In accordance with PropBank, we induce a separate set of clusters for each verb and each cluster thus represents a verb-specific role.

Our algorithm works by iteratively splitting and merging clusters of argument instances in order to arrive at increasingly accurate representations of semantic roles. Although splits and merges could be arbitrarily interleaved, our algorithm executes a single split operation (split phase), followed by a series of merges (merge phase). The split phase partitions the seed cluster containing all argument instances of a particular verb into more fine-grained (sub-)clusters. This initial split results in a clustering with high purity but low collocation, i.e., argument instances in each cluster tend to belong to the same role but argument instances of a particular role are located in many clusters. The degree of dislocation is reduced in the consecutive merge phase, in which clusters that are likely to represent the same role are merged.

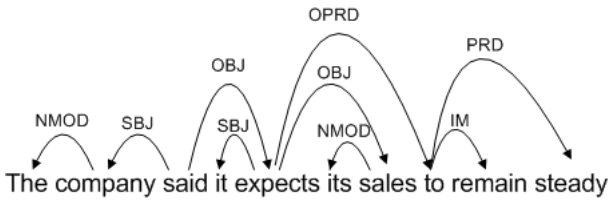


Figure 1: A sample dependency parse with dependency labels SBJ (subject), OBJ (object), NMOD (nominal modifier), OPRD (object predicative complement), PRD (predicative complement), and IM (infinitive marker). See Surdeanu et al. (2008) for more details on this variant of dependency syntax.

5.1 Split Phase

Initially, all arguments of a particular verb are placed in a single cluster. The goal then is to partition this cluster in such a way that the split-off clusters have high purity, i.e., contain argument instances of the same role. Towards this end, we characterize each argument instance by a key, formed by concatenating the following syntactic cues:

- verb voice (active/passive);
- argument linear position relative to predicate (left/right);
- syntactic relation of argument to its governor;
- preposition used for argument realization.

A cluster is allocated for each key and all argument instances with a matching key are assigned to that cluster. Since each cluster encodes fine-grained syntactic distinctions, we assume that arguments occurring in the same position are likely to bear the same semantic role. The assumption is largely supported by our empirical results (see Section 7); the clusters emerging from the initial split phase have a purity of approximately 90%. While the incorporation of additional cues (e.g., indicating the part of speech of the subject or transitivity) would result in even greater purity, it would also create problematically small clusters, thereby negatively affecting the successive merge phase.

5.2 Merge Phase

The split phase creates clusters with high purity, however, argument instances of a particular role are often scattered amongst many clusters resulting in a cluster assignment with low collocation. The goal of the merge phase is to improve collocation by executing a series of merge steps. At each step, pairs of clusters are considered for merging. Each pair is scored by a function that reflects how likely the two clusters are to contain arguments of the same role and the best scoring pair is chosen for merging. In the following, we will specify which pairs of clusters are considered (candidate search), how they are scored, and when the merge phase terminates.

5.2.1 Candidate Search

In principle, we could simply enumerate and score all possible cluster pairs at each iteration. In practice however, such a procedure has a number of drawbacks. Besides being inefficient, it requires a scoring function with comparable scores for arbitrary pairs of clusters. For example, let a, b, c, and d denote clusters. Then, $score(a,b)$ and $score(c,d)$ must be comparable. This is a stronger requirement than demanding that only scores involving some common cluster (e.g., $score(a,b)$ and $score(a,c)$) be comparable. Moreover, it would be desirable to exclude pairings involving small clusters (i.e., with few instances) as scores for these tend to be unreliable. Rather than considering all cluster pairings, we therefore select a specific cluster at each step and score merges between this cluster and certain other clusters. If a sufficiently good merge is found, it is executed, otherwise the clustering does not change. In addition, we prioritize merges between large clusters and avoid merges between small clusters.

Algorithm 1 implements our merging procedure. Each pass through the inner loop (lines 4–12) selects a different cluster to consider at that step. Then, merges between the selected cluster and all larger clusters are considered. The highest-scoring merge is executed, unless all merges are ruled out, i.e., have a score below the threshold α. After each completion of the inner loop, the thresholds contained in the scoring function (discussed below) are adjusted and this is repeated until some termination criterion is met (discussed in Section 5.2.3).

Algorithm 1: Cluster merging procedure. Operation $merge(L_i, L_j)$ merges cluster L_i into cluster L_j and removes L_i from the list L.

```
1  while not done do
2      L ← a list of all clusters sorted by number
       of instances in descending order
3      i ← 1
4      while i < length(L) do
5          j ← arg max_{0≤j'<i} score(L_i, L_j')
6          if score(L_i, L_j) ≥ α then
7              merge(L_i, L_j)
8          end
9          else
10             i ← i + 1
11         end
12     end
13     adjust thresholds
14 end
```

5.2.2 Scoring Function

Our scoring function quantifies whether two clusters are likely to contain arguments of the same role and was designed to reflect the following criteria:

1. whether the arguments found in the two clusters are lexically similar;
2. whether clause-level constraints are satisfied, specifically the constraint that all arguments of a particular clause have different semantic roles, i.e., are assigned to different clusters;
3. whether the arguments present in the two clusters have similar parts of speech.

Qualitatively speaking, criteria (2) and (3) provide negative evidence in the sense that they can be used to rule out incorrect merges but not to identify correct ones. For example, two clusters with drastically different parts of speech are unlikely to represent the same role. However, the converse is not necessarily true as part of speech similarity does not imply role-semantic similarity. Analogously, the fact that clause-level constraints are not met provides evidence against a merge, but the fact that these are satisfied is not reliable evidence in favor of a merge. In contrast, lexical similarity implies that the clus-

ters are likely to represent the same semantic role. It is reasonable to assume that due to selectional restrictions, verbs will be associated with lexical units that are semantically related and assume similar syntactic positions (e.g., *eat* prefers as an object edible things such as *apple, biscuit, meat*), thus bearing the same semantic role. Unavoidably, lexical similarity will be more reliable for arguments with overt lexical content as opposed to pronouns, however this should not impact the scoring of sufficiently large clusters.

Each of the criteria mentioned above is quantified through a separate score and combined into an overall similarity function, which scores two clusters c and c' as follows:

$$score(c,c') = \begin{cases} 0 & \text{if } pos(c,c') < \beta, \\ 0 & \text{if } cons(c,c') < \gamma, \\ lex(c,c') & \text{otherwise.} \end{cases} \quad (2)$$

The particular form of this function is motivated by the distinction between positive and negative evidence. When the part-of-speech similarity (*pos*) is below a certain threshold β or when clause-level constraints (*cons*) are satisfied to a lesser extent than threshold γ, the score takes value zero and the merge is ruled out. If this is not the case, the lexical similarity score (*lex*) determines the magnitude of the overall score. In the remainder of this section we will explain how the individual scores (*pos*, *cons*, and *lex*) are defined and then move on to discuss how the thresholds β and γ are adjusted.

Lexical Similarity We measure lexical similarity between two clusters through cosine similarity. Specifically, each cluster is represented as a vector whose components correspond to the occurrence frequencies of the argument head words in the cluster. The similarity on such vectors x and y is then quantified as:

$$lex(x,y) = cossim(x,y) = \frac{x \cdot y}{\|x\| \|y\|} \quad (3)$$

Clause-Level Constraints Arguments occurring in the same clause cannot bear the same role. Therefore, clusters should not merge if the resulting cluster contains (many) arguments of the same clause. For two clusters c and c' we assess how well they satisfy this clause-level constraint by computing:

$$cons(c,c') = 1 - \frac{2 * viol(c,c')}{NC + NC'} \quad (4)$$

where $viol(c,c')$ refers to the number of pairs of instances $(d,d') \in c \times c'$ for which d and d' occur in the same clause (each instance can participate in at most one pair) and NC and NC' are the number of instances in clusters c and c', respectively.

Part-of-speech Similarity Part-of-speech similarity is also measured through cosine-similarity (equation (3)). Clusters are again represented as vectors x and y whose components correspond to argument part-of-speech tags and values to their occurrence frequency.

5.2.3 Threshold Adaptation and Termination

As mentioned earlier the thresholds β and γ which parametrize the scoring function are adjusted at each iteration. The idea is to start with a very restrictive setting (high values) in which the negative evidence rules out merges more strictly, and then to gradually relax the requirement for a merge by lowering the threshold values. This procedure prioritizes reliable merges over less reliable ones.

More concretely, our threshold adaptation procedure starts with β and γ both set to value 0.95. Then β is lowered by 0.05 at each step, leaving γ unchanged. When β becomes zero, γ is lowered by 0.05 and β is reset to 0.95. Then β is iteratively decreased again until it becomes zero, after which γ is decreased by another 0.05. This is repeated until γ becomes zero, at which point the algorithm terminates. Note that the termination criterion is not tied explicitly to the number of clusters, which is therefore determined automatically.

6 Experimental Setup

In this section we describe how we assessed the performance of our system. We discuss the dataset on which our experiments were carried out, explain how our system's output was evaluated and present the methods used for comparison with our approach.

Data For evaluation purposes, the system's output was compared against the CoNLL 2008 shared task dataset (Surdeanu et al., 2008) which provides

	Syntactic Function			Lang and Lapata			Split-Merge		
	PU	CO	F1	PU	CO	F1	PU	CO	F1
auto/auto	72.9	73.9	73.4	73.2	**76.0**	74.6	**81.9**	71.2	**76.2**
gold/auto	77.7	**80.1**	**78.9**	75.6	79.4	77.4	**84.0**	74.4	**78.9**
auto/gold	77.0	71.0	73.9	77.9	**74.4**	76.2	**86.5**	69.8	**77.3**
gold/gold	81.6	**77.5**	79.5	79.5	76.5	78.0	**88.7**	73.0	**80.1**

Table 2: Clustering results with our split-merge algorithm, the unsupervised model proposed in Lang and Lapata (2010) and a baseline that assigns arguments to clusters based on their syntactic function.

PropBank-style gold standard annotations. The dataset was taken from the Wall Street Journal portion of the Penn Treebank corpus and converted into a dependency format (Surdeanu et al., 2008). In addition to gold standard dependency parses, the dataset also contains automatic parses obtained from the MaltParser (Nivre et al., 2007). Although the dataset provides annotations for verbal and nominal predicate-argument constructions, we only considered the former, following previous work on semantic role labeling (Màrquez et al., 2008).

Evaluation Metrics For each verb, we determine the extent to which argument instances in a cluster share the same gold standard role (purity) and the extent to which a particular gold standard role is assigned to a single cluster (collocation).

More formally, for each group of verb-specific clusters we measure the purity of the clusters as the percentage of instances belonging to the majority gold class in their respective cluster. Let N denote the total number of instances, G_j the set of instances belonging to the j-th gold class and C_i the set of instances belonging to the i-th cluster. Purity can then be written as:

$$PU = \frac{1}{N}\sum_i \max_j |G_j \cap C_i| \quad (5)$$

Collocation is defined as follows. For each gold role, we determine the cluster with the largest number of instances for that role (the role's *primary* cluster) and then compute the percentage of instances that belong to the primary cluster for each gold role as:

$$CO = \frac{1}{N}\sum_j \max_i |G_j \cap C_i| \quad (6)$$

The per-verb scores are aggregated into an overall score by averaging over all verbs. We use the micro-average obtained by weighting the scores for individual verbs proportionately to the number of instances for that verb.

Finally, we use the harmonic mean of purity and collocation as a single measure of clustering quality:

$$F_1 = \frac{2 \times CO \times PU}{CO + PU} \quad (7)$$

Comparison Models We compared our split-merge algorithm against two competitive approaches. The first one assigns argument instances to clusters according to their syntactic function (e.g., subject, object) as determined by a parser. This baseline has been previously used as point of comparison by other unsupervised semantic role labeling systems (Grenager and Manning, 2006; Lang and Lapata, 2010) and shown difficult to outperform. Our implementation allocates up to $N = 21$ clusters[2] for each verb, one for each of the 20 most frequent functions in the CoNLL dataset and a default cluster for all other functions. The second comparison model is the one proposed in Lang and Lapata (2010) (see Section 2). We used the same model settings (with 10 latent variables) and feature set proposed in that paper. Our method's only parameter is the threshold α which we heuristically set to 0.1. On average our method induces 10 clusters per verb.

7 Results

Our results are summarized in Table 2. We report cluster purity (PU), collocation (CO) and their harmonic mean (F1) for the baseline (Syntactic Function), Lang and Lapata's (2010) model and our split-merge algorithm (Split-Merge) on four

[2]This is the number of gold standard roles.

Verb	Freq	Syntactic Function PU	CO	F1	Split-Merge PU	CO	F1
say	15238	91.4	91.3	**91.4**	93.6	81.7	87.2
make	4250	68.6	71.9	70.2	73.3	72.9	**73.1**
go	2109	45.1	56.0	49.9	52.7	51.9	**52.3**
increase	1392	59.7	68.4	63.7	68.8	71.4	**70.1**
know	983	62.4	72.7	**67.1**	63.7	65.9	64.8
tell	911	61.9	76.8	68.6	77.5	70.8	**74.0**
consider	753	63.5	65.6	64.5	79.2	61.6	**69.3**
acquire	704	75.9	79.7	77.7	80.1	76.6	**78.3**
meet	574	76.7	76.0	76.3	88.0	69.7	**77.8**
send	506	69.6	63.8	66.6	83.6	65.8	**73.6**
open	482	63.1	73.4	67.9	77.6	62.2	**69.1**
break	246	53.7	58.9	56.2	68.7	53.3	**60.0**

Table 3: Clustering results for individual verbs with our split-merge algorithm and the syntactic function baseline.

Role	Syntactic Function PU	CO	F1	Split-Merge PU	CO	F1
A0	74.5	87.0	80.3	79.0	88.7	**83.6**
A1	82.3	72.0	76.8	87.1	73.0	**79.4**
A2	65.0	67.3	66.1	82.8	66.2	**73.6**
A3	48.7	76.7	59.6	79.6	76.3	**77.9**
ADV	37.2	77.3	**50.2**	78.8	37.3	**50.6**
CAU	81.8	74.4	**77.9**	84.8	67.2	75.0
DIR	62.7	67.9	**65.2**	71.0	50.7	59.1
EXT	51.4	87.4	64.7	90.4	87.2	**88.8**
LOC	71.5	74.6	**73.0**	82.6	56.7	67.3
MNR	62.6	58.8	**60.6**	81.5	44.1	57.2
TMP	80.5	74.0	**77.1**	80.1	38.7	52.2
MOD	68.2	44.4	53.8	90.4	89.6	**90.0**
NEG	38.2	98.5	55.0	49.6	98.8	**66.1**
DIS	42.5	87.5	57.2	62.2	75.4	**68.2**

Table 4: Clustering results for individual semantic roles with our split-merge algorithm and the syntactic function baseline.

datasets. These result from the combination of automatic parses with automatically identified arguments (auto/auto), gold parses with automatic arguments (gold/auto), automatic parses with gold arguments (auto/gold) and gold parses with gold arguments (gold/gold). Bold-face is used to highlight the best performing system under each measure on each dataset (e.g., auto/auto, gold/auto and so on).

On all datasets, our method achieves the highest purity and outperforms both comparison models by a wide margin which in turn leads to a considerable increase in F1. On the auto/auto dataset the split-merge algorithm results in 9% higher purity than the baseline and increases F1 by 2.8%. Lang and Lapata's (2010) logistic classifier achieves higher collocation but lags behind our method on the other two measures.

Not unexpectedly, we observe an increase in performance for all models when using gold standard parses. On the gold/auto dataset, F1 increases by 2.7% for the split-merge algorithm, 2.7% for the logistic classifier, and 5.5% for the syntactic function baseline. Split-Merge maintains the highest purity and levels the baseline in terms of F1. Performance also increases if gold standard arguments are used instead of automatically identified arguments. Consequently, each model attains its best scores on the gold/gold dataset.

We also assessed the argument identification component on its own (settings auto/auto and gold/auto). It obtained a precision of 88.1% (percentage of semantic arguments out of those identified) and recall of 87.9% (percentage of identified arguments out of all gold arguments). However, note that these figures are not strictly comparable to those reported for supervised systems, due to the fact that our argument identification component only discards non-argument candidates.

Tables 3 and 4 shows how performance varies across verbs and roles, respectively. We compare the syntactic function baseline and the split-merge system on the auto/auto dataset. Table 3 presents results for 12 verbs which we selected so as to exhibit varied occurrence frequencies and alternation patterns. As can be seen, the macroscopic result — increase in F1 (shown in bold face) and purity — also holds across verbs. Some caution is needed in interpreting the results in Table 4[3] since core roles A0–A3 are defined on a per-verb basis and do not necessarily have a uniform corpus-wide interpretation. Thus, conflating scores across verbs is only meaningful to the extent that these labels actually signify the same

[3]Results are shown for four core roles (A0–A3) and all subtypes of the ArgM role, i.e., adjuncts denoting general purpose (ADV), cause (CAU), direction (DIR), extent (EXT), location (LOC), manner (MNR), and time (TMP), modal verbs (MOD), negative markers (NEG), and discourse connectives (DIS).

role (which is mostly true for A0 and A1). Furthermore, the purity scores given here represent the average purity of those clusters for which the specified role is the majority role. We observe that for most roles shown in Table 4 the split-merge algorithm improves upon the baseline with regard to F1, whereas this is uniformly the case for purity.

What are the practical implications of these results, especially when considering the collocation-purity tradeoff? If we were to annotate the clusters induced by our system, low collocation would result in higher annotation effort while low purity would result in poorer data quality. Our system improves purity substantially over the baselines, without affecting collocation in a way that would massively increase the annotation effort. As an example, consider how our system could support humans in labeling an unannotated corpus. (The following numbers are derived from the CoNLL dataset[4] in the auto/auto setting.) We might decide to annotate all induced clusters with more than 10 instances. This means we would assign labels to 74% of instances in the dataset (excluding those discarded during argument identification) and attain a role classification with 79.4% precision (purity).[5] However, instead of labeling all 165,662 instances contained in these clusters individually we would only have to assign labels to 2,869 clusters. Since annotating a cluster takes roughly the same time as annotating a single instance, the annotation effort is reduced by a factor of about 50.

8 Conclusions

In this paper we presented a novel approach to unsupervised role induction which we formulated as a clustering problem. We proposed a split-merge algorithm that iteratively manipulates clusters representing semantic roles whilst trading off cluster purity with collocation. The split phase creates "pure" clusters that contain arguments of the same role whereas the merge phase attempts to increase collocation by merging clusters which are likely to represent the same role. The approach is simple, intuitive and requires no manual effort for training. Coupled with a rule-based component for automatically identifying argument candidates our split-merge algorithm forms an end-to-end system that is capable of inducing role labels without any supervision.

Our approach holds promise for reducing the data acquisition bottleneck for supervised systems. It could be usefully employed in two ways: (a) to create preliminary annotations, thus supporting the "annotate automatically, correct manually" methodology used for example to provide high volume annotation in the Penn Treebank project; and (b) in combination with supervised methods, e.g., by providing useful out-of-domain data for training. An important direction for future work lies in investigating how the approach generalizes across languages as well as reducing our system's reliance on a treebank-trained parser.

Acknowledgments We are grateful to Charles Sutton for his valuable feedback on this work. The authors acknowledge the support of EPSRC (grant GR/T04540/01).

Appendix

The relations in Rule (2) from Table 1 are IM↑↓, PRT↓, COORD↑↓, P↑↓, OBJ↑, PMOD↑, ADV↑, SUB↑↓, ROOT↑, TMP↑, SBJ↑, OPRD↑. The symbols ↑ and ↓ denote the direction of the dependency arc (upward and downward, respectively).

The relations in Rule (3) are ADV↑↓, AMOD↑↓, APPO↑↓, BNF↑↓-, CONJ↑↓, COORD↑↓, DIR↑↓, DTV↑↓-, EXT↑↓, EXTR↑↓, HMOD↑↓, IOBJ↑↓, LGS↑↓, LOC↑↓, MNR↑↓, NMOD↑↓, OBJ↑↓, OPRD↑↓, POSTHON↑↓, PRD↑↓, PRN↑↓, PRP↑↓, PRT↑↓, PUT↑↓, SBJ↑↓, SUB↑↓, SUFFIX↑↓. Dependency labels are abbreviated here. A detailed description is given in Surdeanu et al. (2008), in their Table 4.

References

O. Abend, R. Reichart, and A. Rappoport. 2009. Unsupervised Argument Identification for Semantic Role Labeling. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 28–36, Singapore.

[4] Of course, it makes no sense to label this dataset as it is already labeled.

[5] Purity here is slightly lower than the score reported in Table 2 (auto/auto setting), because it is computed over a different number of clusters (only those with at least 10 instances).

D. Dowty. 1991. Thematic Proto Roles and Argument Selection. *Language*, 67(3):547–619.

H. Fürstenau and M. Lapata. 2009. Graph Aligment for Semi-Supervised Semantic Role Labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 11–20, Singapore.

D. Gildea and D. Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288.

A. Gordon and R. Swanson. 2007. Generalizing Semantic Role Annotations Across Syntactically Similar Verbs. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 192–199, Prague, Czech Republic.

T. Grenager and C. Manning. 2006. Unsupervised Discovery of a Statistical Verb Lexicon. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, pages 1–8, Sydney, Australia.

K. Kipper, H. T. Dang, and M. Palmer. 2000. Class-Based Construction of a Verb Lexicon. In *Proceedings of the 17th AAAI Conference on Artificial Intelligence*, pages 691–696. AAAI Press / The MIT Press.

J. Lang and M. Lapata. 2010. Unsupervised Induction of Semantic Roles. In *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 939–947, Los Angeles, California.

L. Màrquez, X. Carreras, K. Litkowski, and S. Stevenson. 2008. Semantic Role Labeling: an Introduction to the Special Issue. *Computational Linguistics*, 34(2):145–159, June.

G. Melli, Y. Wang, Y. Liu, M. M. Kashani, Z. Shi, B. Gu, A. Sarkar, and F. Popowich. 2005. Description of SQUASH, the SFU Question Answering Summary Handler for the DUC-2005 Summarization Task. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing Document Understanding Workshop*, Vancouver, Canada.

J. Nivre, J. Hall, J. Nilsson, G. Eryigit A. Chanev, S. Kübler, S. Marinov, and E. Marsi. 2007. MaltParser: A Language-independent System for Data-driven Dependency Parsing. *Natural Language Engineering*, 13(2):95–135.

S. Padó and M. Lapata. 2009. Cross-lingual Annotation Projection of Semantic Roles. *Journal of Artificial Intelligence Research*, 36:307–340.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

S. Pradhan, W. Ward, and J. Martin. 2008. Towards Robust Semantic Role Labeling. *Computational Linguistics*, 34(2):289–310.

D. Shen and M. Lapata. 2007. Using Semantic Roles to Improve Question Answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the Conference on Computational Natural Language Learning*, pages 12–21, Prague, Czech Republic.

M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. 2003. Using Predicate-Argument Structures for Information Extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 8–15, Sapporo, Japan.

M. Surdeanu, R. Johansson, A. Meyers, and L. Màrquez. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *Proceedings of the 12th CoNLL*, pages 159–177, Manchester, England.

R. Swier and S. Stevenson. 2004. Unsupervised Semantic Role Labelling. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, pages 95–102, Barcelona, Spain.

D. Wu and P. Fung. 2009. Semantic Roles for SMT: A Hybrid Two-Pass Model. In *Proceedings of North American Annual Meeting of the Association for Computational Linguistics HLT 2009: Short Papers*, pages 13–16, Boulder, Colorado.

Using Cross-Entity Inference to Improve Event Extraction

Yu Hong Jianfeng Zhang Bin Ma Jianmin Yao Guodong Zhou Qiaoming Zhu
School of Computer Science and Technology, Soochow University, Suzhou City, China
{hongy, jfzhang, bma, jyao, gdzhou, qmzhu}@suda.edu.cn

Abstract

Event extraction is the task of detecting certain specified types of events that are mentioned in the source language data. The state-of-the-art research on the task is transductive inference (e.g. cross-event inference). In this paper, we propose a new method of event extraction by well using cross-entity inference. In contrast to previous inference methods, we regard entity-type consistency as key feature to predict event mentions. We adopt this inference method to improve the traditional sentence-level event extraction system. Experiments show that we can get 8.6% gain in trigger (event) identification, and more than 11.8% gain for argument (role) classification in ACE event extraction.

1 Introduction

The event extraction task in ACE (Automatic Content Extraction) evaluation involves three challenging issues: distinguishing events of different types, finding the participants of an event and determining the roles of the participants.

The recent researches on the task show the availability of transductive inference, such as that of the following methods: cross-document, cross-sentence and cross-event inferences. Transductive inference is a process to use the known instances to predict the attributes of unknown instances. As an example, given a target event, the cross-event inference can predict its type by well using the related events co-occurred with it within the same document. From the sentence:

*(1)He **left** the company.*

it is hard to tell whether it is a *Transport* event in ACE, which means that he left the place; or an *End-Position* event, which means that he retired from the company. But cross-event inference can use a related event "*Then he went shopping*" within the same document to identify it as a *Transport* event correctly.

As the above example might suggest, the availability of transductive inference for event extraction relies heavily on the known evidences of an event occurrence in specific condition. However, the evidence supporting the inference is normally unclear or absent. For instance, the relation among events is the key clue for cross-event inference to predict a target event type, as shown in the inference process of the sentence (1). But event relation extraction itself is a hard task in Information Extraction. So cross-event inference often suffers from some false evidence (viz., misleading by unrelated events) or lack of valid evidence (viz., unsuccessfully extracting related events).

In this paper, we propose a new method of transductive inference, named cross-entity inference, for event extraction by well using the relations among entities. This method is firstly motivated by the inherent ability of entity types in revealing event types. From the sentences:

*(2)He left the **bathroom**.*
*(3)He left **Microsoft**.*

it is easy to identify the sentence (2) as a *Transport* event in ACE, which means that he left the place, because nobody would retire (*End-Position* type) from a bathroom. And compared to the entities in sentence (1) and (2), the entity "*Microsoft*" in (3) would give us more confidence to tag the "*left*" event as an *End-Position* type, because people are used to giving the full name of the place where they retired.

The cross-entity inference is also motivated by the phenomenon that the entities of the same type often attend similar events. That gives us a way to predict event type based on entity-type consistency. From the sentence:

*(4)**Obama** beats McCain.*

it is hard to identify it as an *Elect* event in ACE, which means Obama wins the Presidential Election,

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1127–1136,
Portland, Oregon, June 19-24, 2011.

or an *Attack* event, which means Obama roughs somebody up. But if we have the priori knowledge that the sentence "*Bush beats McCain*" is an *Elect* event, and "*Obama*" was a presidential contender just like "*Bush*" (strict type consistency), we have ample evidence to predict that the sentence (4) is also an *Elect* event.

Indeed above cross-entity inference for event-type identification is not the only use of entity-type consistency. As we shall describe below, we can make use of it at all issues of event extraction:

- For **event type**: the entities of the same type are most likely to attend similar events. And the events often use consistent or synonymous trigger.
- For **event argument (participant)**: the entities of the same type normally co-occur with similar participants in the events of the same type.
- For **argument role**: the arguments of the same type, for the most part, play the same roles in similar events.

With the help of above characteristics of entity, we can perform a step-by-step inference in this order:

- ***Step 1***: predicting event type and labeling trigger given the entities of the same type.
- ***Step 2***: identifying arguments in certain event given priori entity type, event type and trigger that obtained by step 1.
- ***Step 3***: determining argument roles in certain event given entity type, event type, trigger and arguments that obtained by step 1 and step 2.

On the basis, we give a blind cross-entity inference method for event extraction in this paper. In the method, we first regard entities as queries to retrieve their related documents from large-scale language resources, and use the global evidences of the documents to generate entity-type descriptions. Second we determine the type consistency of entities by measuring the similarity of the type descriptions. Finally, given the priori attributes of events in the training data, with the help of the entities of the same type, we perform the step-by-step cross-entity inference on the attributes of test events (candidate sentences).

In contrast to other transductive inference methods on event extraction, the cross-entity inference makes every effort to strengthen effects of entities in predicting event occurrences. Thus the inferential process can benefit from following aspects: 1) less false evidence, viz. less false entity-type consistency (the key clue of cross-entity inference), because the consistency can be more precisely determined with the help of fully entity-type description that obtained based on the related information from Web; 2) more valid evidence, viz. more entities of the same type (the key references for the inference), because any entity never lack its congeners.

2 Task Description

The event extraction task we addressing is that of the Automatic Content Extraction (ACE) evaluations, where an event is defined as a specific occurrence involving participants. And event extraction task requires that certain specified types of events that are mentioned in the source language data be detected. We first introduce some ACE terminology to understand this task more easily:

- **Entity**: an object or a set of objects in one of the semantic categories of interest, referred to in the document by one or more (co-referential) entity mentions.
- **Entity mention**: a reference to an entity (typically, a noun phrase).
- **Event trigger**: the main word that most clearly expresses an event occurrence (An ACE event trigger is generally a verb or a noun).
- **Event arguments**: the entity mentions that are involved in an event (viz., participants).
- **Argument roles**: the relation of arguments to the event where they participate.
- **Event mention**: a phrase or sentence within which an event is described, including trigger and arguments.

The 2005 ACE evaluation had 8 types of events, with 33 subtypes; for the purpose of this paper, we will treat these simply as 33 separate event types and do not consider the hierarchical structure among them. Besides, the ACE evaluation plan defines the following standards to determine the correctness of an event extraction:

- A trigger is correctly labeled if its event type and offset (viz., the position of the trigger word in text) match a reference trigger.
- An argument is correctly identified if its event type and offsets match any of the reference argument mentions, in other word, correctly recognizing participants in an event.
- An argument is correctly classified if its role matches any of the reference argument mentions.

Consider the sentence:

*(5) It has refused in **the last five years** to **revoke** the license of **a single doctor** for committing medical errors.*[1]

The event extractor should detect an *End-Position* event mention, along with the trigger word "*revoke*", the position "*doctor*", the person whose license should be revoked, and the time during which the event happened:

Event type	***End-Position***	
Trigger	***revoke***	
Arguments	***a single doctor***	Role=***Person***
	doctor	Role=***Position***
	the last five years	Role=***Time-within***

Table 1: Event extraction example

It is noteworthy that event extraction depends on previous phases like name identification, entity mention co-reference and classification. Thereinto, the name identification is another hard task in ACE evaluation and not the focus in this paper. So we skip the phase and instead directly use the entity labels provided by ACE.

3 Related Work

Almost all the current ACE event extraction systems focus on processing one sentence at a time (Grishman et al., 2005; Ahn, 2006; Hardyet al. 2006). However, there have been several studies using high-level information from a wider scope:

Maslennikov and Chua (2007) use discourse trees and local syntactic dependencies in a pattern-based framework to incorporate wider context to refine the performance of relation extraction. They claimed that discourse information could filter noisy dependency paths as well as increasing the reliability of dependency path extraction.

Finkel et al. (2005) used Gibbs sampling, a simple Monte Carlo method used to perform approximate inference in factored probabilistic models. By using simulated annealing in place of Viterbi decoding in sequence models such as HMMs, CMMs, and CRFs, it is possible to incorporate non-local structure while preserving tractable inference. They used this technique to augment an information extraction system with long-distance dependency models, enforcing label consistency and extraction template consistency constraints.

Ji and Grishman (2008) were inspired from the hypothesis of "One Sense Per Discourse" (Yarowsky, 1995); they extended the scope from a single document to a cluster of topic-related documents and employed a rule-based approach to propagate consistent trigger classification and event arguments across sentences and documents. Combining global evidence from related documents with local decisions, they obtained an appreciable improvement in both event and event argument identification.

Patwardhan and Riloff (2009) proposed an event extraction model which consists of two components: a model for sentential event recognition, which offers a probabilistic assessment of whether a sentence is discussing a domain-relevant event; and a model for recognizing plausible role fillers, which identifies phrases as role fillers based upon the assumption that the surrounding context is discussing a relevant event. This unified probabilistic model allows the two components to jointly make decisions based upon both the local evidence surrounding each phrase and the "peripheral vision".

Gupta and Ji (2009) used cross-event information within ACE extraction, but only for recovering implicit time information for events.

Liao and Grishman (2010) propose document level cross-event inference to improve event extraction. In contrast to Gupta's work, Liao do not limit themselves to time information for events, but rather use related events and event-type consistency to make predictions or resolve ambiguities regarding a given event.

4 Motivation

In event extraction, current transductive inference methods focus on the issue that many events are missing or spuriously tagged because the local information is not sufficient to make a confident decision. The solution is to mine credible evidences of event occurrences from global information and regard that as priori knowledge to predict unknown event attributes, such as that of cross-document and cross-event inference methods.

However, by analyzing the sentence-level baseline event extraction, we found that the entities within a sentence, as the most important local information, actually contain sufficient clues for event detection. It is only based on the premise that we know the backgrounds of the entities beforehand. For instance, if we knew the entity "*vesuvius*" is an active volcano, we could easily identify

[1] Selected from the file "CNN_CF_20030304.1900.02" in ACE-2005 corpus.

the word "*erupt*", which co-occurred with the entity, as the trigger of a "*volcanic eruption*" event but not that of a "*spotty rash*".

In spite of that, it is actually difficult to use an entity to directly infer an event occurrence because we normally don't know the inevitable connection between the background of the entity and the event attributes. But we can well use the entities of the same background to perform the inference. In detail, if we first know *entity(a)* has the same background with *entity(b)*, and we also know that *entity(a)*, as a certain role, participates in a specific event, then we can predict that *entity(b)* might participtes in a similar event as the same role.

Consider the two sentences[2] from ACE corpus:

(5) ***American*** *case for* ***war*** *against* ***Saddam***.

(6) ***Bush*** *should* ***torture*** *the* ***al Qaeda chief operations officer***.

The sentences are two event mentions which have the same attributes:

(5)	Event type	***Attack***	
	Trigger	***war***	
	Arguments	***American***	Role=***Attacker***
		Saddam	Role=***Target***
(6)	Event type	***Attack***	
	Trigger	***torture***	
	Arguments	***Bush***	Role=***Attacker***
		...Qaeda chief ...	Role=***Target***

Table 2: Cross-entity inference example

From the sentences, we can find that the entities "*Saddam*" and "*Qaeda chief*" have the same background (viz., terrorist leader), and they are both the arguments of *Attack* events as the role of *Target*. So if we previously know any of the event mentions, we can infer another one with the help of the entities of the same background.

In a word, the cross-entity inference, we proposed for event extraction, bases on the hypothesis:

Entities of the consistent type normally participate in similar events as the same role.

As we will introduce below, some statistical data from ACE training corpus can support the hypothesis, which show the consistency of event type and role in event mentions where entities of the same type occur.

4.1 Entity Consistency and Distribution

Within the ACE corpus, there is a strong entity consistency: if one entity mention appears in a type of event, other entity mentions of the same type will appear in similar events, and even use the same word to trigger the events. To see this we calculated the conditional probability (in the ACE corpus) of a certain entity type appearing in the 33 ACE event subtypes.

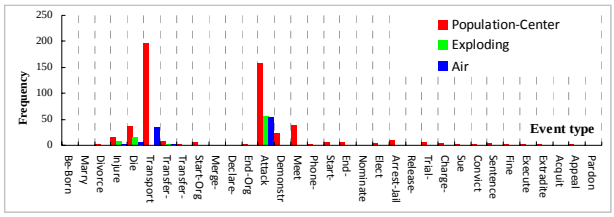


Figure 1. Conditional probability of a certain entity type appearing in the 33 ACE event subtypes (Here only the probabilities of *Population-Center*, *Exploding* and *Air* entities as examples)

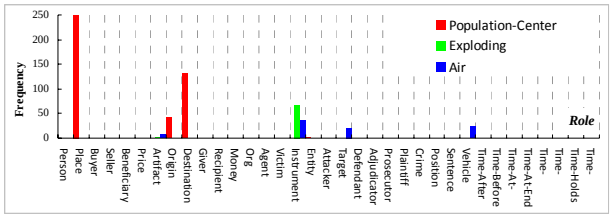


Figure 2. Conditional probability of an entity type appearing as the 34 ACE role types (Here only the probabilities of *Population-Center*, *Exploding* and *Air* entities as examples)

As there are 33 event subtypes and 43 entity types, there are potentially 33*43=1419 entity-event combinations. However, only a few of these appear with substantial frequency. For example, the *Population-Center* entities only occur in 4 types of event mentions with the conditional probability more than 0.05. From Table 3, we can find that only *Attack* and *Transport* events co-occur frequently with *Population-Center* entities (see Figure 1 and Table 3).

Event	**Cond.Prob.**	**Freq.**
Transport	0.368	197
Attack	0.295	158
Meet	0.073	39
Die	0.069	37

Table 3: Events co-occurring with *Population-Center* with the conditional probability > 0.05

Actually we find that most entity types appear in more restricted event mentions than *Population-Center* entity. For example, *Air* entity only co-occurs with 5 event types (*Attack*, *Transport*, *Die*, *Transfer-Ownership* and *Injure*), and *Exploding*

2 They are extracted from the files "CNN_CF_20030305.1900.00-1" and "CNN_CF_20030303.1900.06-1" respectively.

entity co-occurs with 4 event types (see Figure 1). Especially, they only co-occur with one or two event types with the conditional probability more than 0.05.

	Evnt.<=5	5<Evnt.<=10	Evnt.>10
Freq. > 0	24	7	12
Freq. >10	37	4	2
Freq. >50	41	1	1

Table 4: Distribution of entity-event combination corresponding to different co-occurrence frequency

Table 4 gives the distributions of whole ACE entity types co-occurring with event types. We can find that there are 37 types of entities (out of 43 in total) appearing in less than 5 types of event mentions when entity-event co-occurrence frequency is larger than 10, and only 2 (e.g. *Individual*) appearing in more than 10 event types. And when the frequency is larger than 50, there are 41 (95%) entity types co-occurring with less than 5 event types. These distributions show the fact that most instances of a certain entity type normally participate in events of the same type. And the distributions might be good predictors for event type detection and trigger determination.

Air (Entity type)	
Attack event	***Fighter plane*** (subtype 1): "*MiGs*" "*enemy planes*" "*warplanes*" "*allied aircraft*" "*U.S. jets*" "*a-10 tank killer*" "*b-1 bomber*" "*a-10 warthog*" "*f-14 aircraft*" "*apache helicopter*"
Transport event	***Spacecraft*** (subtype 2): "*russian soyuz capsule*" "*soyuz*"
	Civil aviation (subtype 3): "*airliners*" "*the airport*" "*Hooters Air executive*"
	Private plane (subtype 4): "*Marine One*" "*commercial flight*" "*private plane*"

Table 5: Event types co-occurred with *Air* entities

Besides, an ACE entity type actually can be divided into more cohesive subtypes according to similarity of background of entity, and such a subtype nearly always co-occur with unique event type. For example, the *Air* entities can be roughly divided into 4 subtypes: *Fighter plane*, *Spacecraft*, *Civil aviation* and *Private plane*, within which the *Fighter plane* entities all appear in *Attack* event mentions, and other three subtypes all co-occur with *Transport* events (see Table 5). This consistency of entities in a subtype is helpful to improve the precision of the event type predictor.

4.2 Role Consistency and Distribution

The same thing happens for entity-role combinations: entities of the same type normally play the same role, especially in the event mentions of the same type. For example, the *Population-Center* entities occur in ACE corpus as only 4 role types: *Place*, *Destination*, *Origin* and *Entity* respectively with conditional probability 0.615, 0.289, 0.093, 0.002 (see Figure 2). And They mainly appear in *Transport* event mentions as *Place*, and in *Attack* as *Destination*. Particularly the *Exploding* entities only occur as *Instrument* and *Artifact* respectively with the probability 0.986 and 0.014. They almost entirely appear in *Attack* events as *Instrument*.

	Evnt.<=5	5<Evnt.<=10	Evnt.>10
Freq. > 0	32	5	6
Freq. >10	38	3	2
Freq. >50	42	1	0

Table 6: Distribution of entity-role combination corresponding to different co-occurrence frequency

Table 6 gives the distributions of whole entity-role combinations in ACE corpus. We can find that there are 38 entity types (out of 43 in total) occur as less than 5 role types when the entity-role co-occurrence frequency is larger than 10. There are 42 (98%) when the frequency is larger than 50, and only 2 (e.g. *Individual*) when larger than 10. The distributions show that the instances of an entity type normally occur as consistent role, which is helpful for cross-entity inference to predict roles.

5 Cross-entity Approach

In this section we present our approach to using blind cross-entity inference to improve sentence-level ACE event extraction.

Our event extraction system extracts events independently for each sentence, because the definition of event mention constrains them to appear in the same sentence. Every sentence that at least involves one entity mention will be regarded as a candidate event mention, and a randomly selected entity mention from the candidate will be the staring of the whole extraction process. For the entity mention, information retrieval is used to mine its background knowledge from Web, and its type is determined by comparing the knowledge with those in training corpus. Based on the entity type, the extraction system performs our step-by-step cross-entity inference to predict the attributes of

the candidate event mention: trigger, event type, arguments, roles and whether or not being an event mention. The main frame of our event extraction system is shown in Figure 3, which includes both training and testing processes.

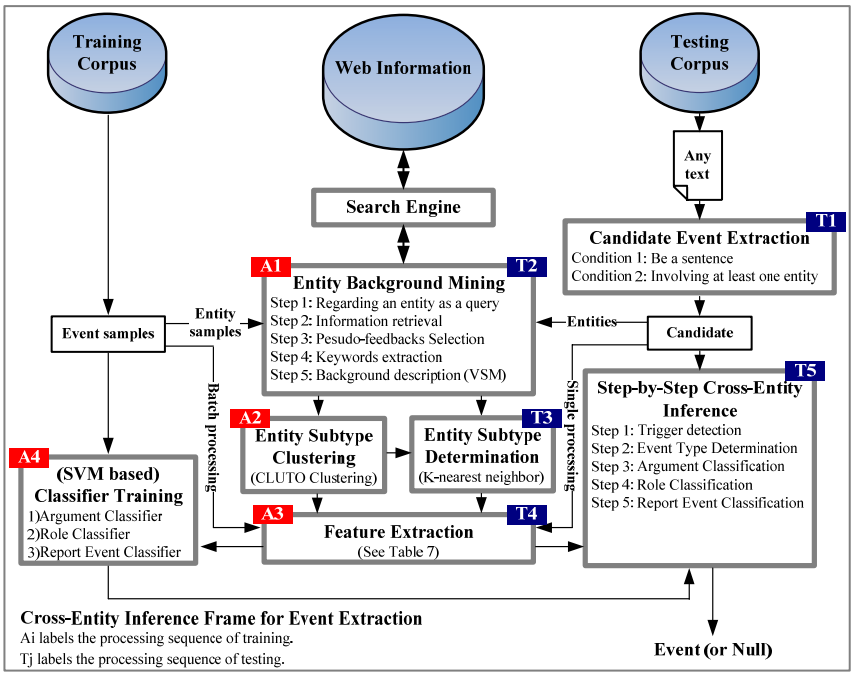


Figure 3. The frame of cross-entity inference for event extraction (including training and testing processes)

In the training process, for every entity type in the ACE training corpus, a clustering technique (CLUTO toolkit)[3] is used to divide it into different cohesive subtypes, each of which only contains the entities of the same background. For instance, the *Air* entities will be divided into *Fighter plane*, *Spacecraft*, *Civil aviation*, *Private plane*, etc (see Table 5). And for each subtype, we mine event mentions where this type of entities appear from ACE training corpus, and extract all the words which trigger the events to establish corresponding trigger list. Besides, a set of support vector machine (SVM) based classifiers are also trained:

- Argument Classifier: to distinguish arguments of a potential trigger from non-arguments[4];
- Role Classifier: to classify arguments by argument role;
- Reportable-Event Classifier (Trigger Classifier): Given entity types, a potential trigger, an event type, and a set of arguments, to determine whether there is a reportable event mention.

In the test process, for each candidate event mention, our event extraction system firstly predicts its triggers and event types: given an randomly selected entity mention from the candidate, the system determines the entity subtype it belonging to and the corresponding trigger list, and then all non-entity words in the candidate are scanned for a instance of triggers from the list. When an instance is found, the system tags the candidate as the event type that the most frequently co-occurs with the entity subtype in the events that triggered by the instance. Secondly the argument classifier is applied to the remaining mentions in the candidate; for any argument passing that classifier, the role classifier is used to assign a role to it. Finally, once all arguments have been assigned, the reportable-event classifier is applied to the candidate; if the result is successful, this event mention is reported.

5.1 Further Division of Entity Type

One of the most important pretreatments before our blind cross-entity inference is to divide the ACE entity type into more cohesive subtype. The greater consistency among backgrounds of entities in such a subtype might be good to improve the precision of cross-entity inference.

[3]http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA439508

[4] It is noteworthy that a sentence may include more than one event (more than one trigger). So it is necessary to distinguish arguments of a potential trigger from that of others.

For each ACE entity type, we collect all entity mentions of the type from training corpus, and regard each such mention as a query to retrieve the 50 most relevant documents from Web. Then we select 50 key words that the most weighted by TFIDF in the documents to roughly describe background of entity. After establishing the vector space model (VSM) for each entity mention of the type, we adopt a clustering toolkit (CLUTO) to further divide the mentions into different subtypes. Finally, for each subtype, we describe its centroid by using 100 key words which the most frequently occurred in relevant documents of entities of the subtype.

In the test process, for an entity mention in a candidate event mention, we determine its type by comparing its background against all centroids of subtypes in training corpus, and the subtype whose centroid has the most Cosine similarity with the background will be assigned to the entity. It is noteworthy that global information from the Web is only used to measure the entity-background consistency and not directly in the inference process. Thus our event extraction system actually still performs a sentence-level inference based on local information.

5.2 Cross-Entity Inference

Our event extraction system adopts a step-by-step cross-entity inference to predict event. As discussed above, the first step is to determine the trigger in a candidate event mention and tag its event type based on consistency of entity type. Given the domain of event mention that restrained by the known trigger, event type and entity subtype, the second step is to distinguish the most probable arguments that co-occurring in the domain from the non-arguments. Then for each of the arguments, the third step can use the co-occurring arguments in the domain as important contexts to predict its role. Finally, the inference process determines whether the candidate is a reportable event mention according to a confidence coefficient. In the following sections, we focus on introducing the three classifiers: argument classifier, role classifier and reportable-event classifier.

5.2.1 Cross-Entity Argument Classifier

For a candidate event mention, the first step gives its event type, which roughly restrains the domain of event mentions where the arguments of the candidate might co-occur. On the basis, given an entity mention in the candidate and its type (see the pretreatment process in section 5.1), the argument classifier could predict whether other entity mentions co-occur with it in such a domain, if yes, all the mentions will be the arguments of the candidate. In other words, if we know an entity of a certain type participates in some event, we will think of what entities also should participate in the event. For instance, when we know a *defendant* goes on trial, we can conclude that the *judge*, *lawyer* and *witness* should appear in *court*.

Argument Classifier
Feature 1: an event type (an event-mention domain)
Feature 2: an entity subtype
Feature 3: entity-subtype co-occurrence in domain
Feature 4: distance to trigger
Feature 5: distances to other arguments
Feature 6: co-occurrence with trigger in clause
Role Classifier
Feature 1 and ***Feature 2***
Feature 7: entity-subtypes of arguments
Reportable-Event Classifier
Feature 1
Feature 8: confidence coefficient of trigger in domain
Feature 9: confidence coefficient of role in domain

Table 7: Features selected for SVM-based cross-entity classifiers

A SVM-based argument classifier is used to determine arguments of candidate event mention. Each feature of this classifier is the conjunction of:

- The subtype of an entity
- The event type we are trying to assign an argument to
- A binary indicator of whether this entity subtype co-occurs with other subtypes in such an event type (There are 266 entity subtypes, and so 266 features for each instance)

Some minor features, such as another binary indicator of whether arguments co-occur with trigger in the same clause (see Table 7).

5.2.2 Cross-Entity Role Classifier

For a candidate event mention, the arguments that given by the second step (argument classifier) provide important contextual information for predicting what role the local entity (also one of the arguments) takes on. For instance, when *citizens* (Arg1) co-occur with *terrorist* (Arg2), most likely the role of Arg1 is *Victim*. On the basis, with the help of event type, the prediction might be more

precise. For instance, if the Arg1 and Arg2 co-occur in an *Attack* event mention, we will have more confidence in the *Victim* role of Arg1.

Besides, as discussed in section 4, entities of the same type normally take on the same role in similar events, especially when they co-occur with similar arguments in the events (see Table 2). Therefore, all instances of co-occurrence model {entity subtype, event type, arguments} in training corpus could provide effective evidences for predicting the role of argument in the candidate event mention. Based on this, we trained a SVM-based role classifier which uses following features:

- Feature 1 and Feature 2 (see Table 7)
- Given the event domain that restrained by the entity and event types, an indicator of what subtypes of arguments appear in the domain. (266 entity subtypes make 266 features for each instance)

5.2.3 Reportable-Event Classifier

At this point, there are still two issues need to be resolved. First, some triggers are common words which often mislead the extraction of candidate event mention, such as "*it*", "*this*", "*what*", etc. These words only appear in a few event mentions as trigger, but when they once appear in trigger list, a large quantity of noisy sentences will be regarded as candidates because of their commonness in sentences. Second, some arguments might be tagged as more than one role in specific event mentions, but as ACE event guideline, one argument only takes on one role in a sentence. So we need to remove those with low confidence.

A confidence coefficient is used to distinguish the correct triggers and roles from wrong ones. The coefficient calculate the frequency of a trigger (or a role) appearing in specific domain of event mentions and that in whole training corpus, then combines them to represent its confidence degree, just like TFIDF algorithm. Thus, the more typical triggers (or roles) will be given high confidence. Based on the coefficient, we use a SVM-based classifier to determine the reportable events. Each feature of this classifier is the conjunction of:

- An event type (domain of event mentions)
- Confidence coefficients of triggers in domain
- Confidence coefficients of roles in the domain.

6 Experiments

We followed Liao (2010)'s evaluation and randomly select 10 newswire texts from the ACE 2005 training corpus as our development set, which is used for parameter tuning, and then conduct a blind test on a separate set of 40 ACE 2005 newswire texts. We use the rest of the ACE training corpus (549 documents) as training data for our event extraction system.

To compare with the reported work on cross-event inference (Liao, 2010) and its sentence-level baseline system, we cross-validate our method on 10 separate sets of 40 ACE texts, and report the optimum, worst and mean performances (see Table 8) on the data by using Precision (P), Recall (R) and F-measure (F). In addition, we also report the performance of two human annotators on 40 ACE newswire texts (a random blind test set): one knows the rules of event extraction; the other knows nothing about it.

6.1 Main Results

From the results presented in Table 8, we can see that using the cross-entity inference, we can improve the F score of sentence-level event extraction for trigger classification by 8.59%, argument classification by 11.86%, and role classification by 11.9% (mean performance). Compared to the cross-event inference, we gains 2.87% improvement for argument classification, and 3.81% for role classification (mean performance). Especially, our worst results also have better performances than cross-event inference.

Nonetheless, the cross-entity inference has worse F score for trigger determination. As we can see, the low Recall score weaken its F score (see Table 8). Actually, we select the sentence which at least includes one entity mention as candidate event mention, but lots of event mentions in ACE never include any entity mention. Thus we have missed some mentions at the starting of inference process.

In addition, the annotator who knows the rules of event extraction has a similar performance trend with systems: high for trigger classification, middle for argument classification, and low for role classification (see Table 8). But the annotator who never works in this field obtains a different trend: higher performance for argument classification. This phenomenon might prove that the step-by-step inference is not the only way to predicate event mention because human can determine arguments without considering triggers and event types.

Performance / System/Human	Trigger (%)			Argument (%)			Role (%)		
	P	R	F	P	R	F	P	R	F
Sentence-level baseline	67.56	53.54	59.74	46.45	37.15	41.29	41.02	32.81	36.46
Cross-event inference	**68.71**	**68.87**	**68.79**	**50.85**	**49.72**	**50.28**	**45.06**	**44.05**	**44.55**
Cross-entity inference (optimum)	73.4	66.2	69.61	56.96	55.1	56	49.3	46.59	47.9
Cross-entity inference (worst)	71.3	64.17	66.1	51.28	50.3	50.78	46.3	44.3	45.28
Cross-entity inference (mean)	**72.9**	**64.3**	**68.33**	**53.4**	**52.9**	**53.15**	**51.6**	**45.5**	**48.36**
Human annotation 1 (blind)	58.9	59.1	59.0	62.6	65.9	64.2	50.3	57.69	53.74
Human annotation 2 (know rules)	74.3	76.2	75.24	68.5	75.8	71.97	61.3	68.8	64.86

Table 8: Overall performance on blind test data

6.2 Influence of Clustering on Inference

A main part of our blind inference system is the entity-type consistency detection, which relies heavily on the correctness of entity clustering and similarity measurement. In training, we used CLUTO clustering toolkit to automatically generate different types of entities based on their background-similarities. In testing, we use K-nearest neighbor algorithm to determine entity type.

Fighter plane (subtype 1 in ***Air*** entities): "*warplanes*" "*allied aircraft*" "*U.S. jets*" "*a-10 tank killer*" "*b-1 bomber*" "*a-10 warthog*" "*f-14 aircraft*" "*apache helicopter*" ***"terrorist" "Saddam" "Saddam Hussein" "Baghdad"…***

Table 9: Noises in subtype 1 of "*Air*" entities (The blod fonts are noises)

We obtained 129 entity subtypes from training set. By randomly inspecting 10 subtypes, we found nearly every subtype involves no less than 19.2% noises. For example, the subtype 1 of "*Air*" in Table 5 lost the entities of "*MiGs*" and "*enemy planes*", but involved "*terrorist*", "*Saddam"*, etc (See Table 9). Therefore, we manually clustered the subtypes and retry the step-by-step cross-entity inference. The results (denoted as "Visible 1") are shown in Table 10, within which, we additionally show the performance of the inference on the rough entity types provided by ACE (denoted as "Visible 2"), such as the type of "*Air*", "*Population-Center*", "*Exploding*", etc., which normally can be divided into different more cohesive subtypes. And the "Blind" in Table 10 denotes the performances on our subtypes obtained by CLUTO.

It is surprised that the performances (see Table 10, F-score) on "Visible 1" entity subtypes are just a little better than "Blind" inference. So it seems that the noises in our blind entity types (CLUTO clusters) don't hurt the inference much. But by re-inspecting the "Visible 1" subtypes, we found that their granularities are not enough small: the 89 manual entity clusters actually can be divided into more cohesive subtypes. So the improvements of inference on noise-free "Visible 1" subtypes are partly offset by loss on weakly consistent entities in the subtypes. It can be proved by the poor performances on "Visible 2" subtypes which are much more general than "Visible 1". Therefore, a reasonable clustering method is important in our inference process.

F-score	Trigger	Argument	Role
Blind	68.33	53.15	48.36
Visible 1	69.15	53.65	48.83
Visible 2	51.34	43.40	39.95

Table 10: Performances on visible VS blind

7 Conclusions and Future Work

We propose a blind cross-entity inference method for event extraction, which well uses the consistency of entity mention to achieve sentence-level trigger and argument (role) classification. Experiments show that the method has better performance than cross-document and cross-event inferences in ACE event extraction.

The inference presented here only considers the helpfulness of entity types of arguments to role classification. But as a superior feature, contextual roles can provide more effective assistance to role determination of local argument. For instance, when an *Attack* argument appears in a sentence, a *Target* might be there. So if we firstly identify simple roles, such as the condition that an argument has only a single role, and then use the roles as priori knowledge to classify hard ones, may be able to further improve performance.

Acknowledgments

We thank Ruifang He. And we acknowledge the support of the National Natural Science Foundation of China under Grant Nos. 61003152, 60970057, 90920004.

References

David Ahn. 2006. The stages of event extraction. In *Proc. COLING/ACL 2006 Workshop on Annotating and Reasoning about Time and Events.*Sydney, Australia.

Jenny Rose Finkel, Trond Grenager and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proc. 43rd Annual Meeting of the Association for Computational Linguistics,* pages 363–370, Ann Arbor, MI, June.

Prashant Gupta and Heng Ji. 2009. Predicting Unknown Time Arguments based on Cross-Event Propagation. In *Proc. ACL-IJCNLP 2009.*

Ralph Grishman, David Westbrook and Adam Meyers. 2005. NYU's English ACE 2005 System Description. In *Proc. ACE 2005 Evaluation Workshop,* Gaithersburg, MD.

Hilda Hardy, Vika Kanchakouskaya and Tomek Strzalkowski. 2006. Automatic Event Classification Using Surface Text Features. In *Proc. AAAI06 Workshop on Event Extraction and Synthesis.* Boston, MA.

Heng Ji and Ralph Grishman. 2008. Refining Event Extraction through Cross-Document Inference. In *Proc. ACL-08: HLT,* pages 254–262, Columbus, OH, June.

Shasha Liao and Ralph Grishman. 2010. Using Document Level Cross-Event Inference to Improve Event Extraction. *In Proc. ACL-2010,* pages 789-797, Uppsala, Sweden, July.

Mstislav Maslennikov and Tat-Seng Chua. 2007. A Multi resolution Framework for Information Extraction from Free Text. In *Proc. 45th Annual Meeting of the Association of Computational Linguistics,* pages 592–599, Prague, Czech Republic, June.

Siddharth Patwardhan and Ellen Riloff. 2007. Effective Information Extraction with Semantic Affinity Patterns and Relevant Regions. In *Proc. Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 2007,* pages 717–727, Prague, Czech Republic, June.

Siddharth Patwardhan and Ellen Riloff. 2009. A Unified Model of Phrasal and Sentential Evidence for Information Extraction. In *Proc. Conference on Empirical Methods in Natural Language Processing 2009, (EMNLP-09).*

David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proc. ACL 1995.* Cambridge, MA.

Peeling Back the Layers: Detecting Event Role Fillers in Secondary Contexts

Ruihong Huang and Ellen Riloff
School of Computing
University of Utah
Salt Lake City, UT 84112
{huangrh,riloff}@cs.utah.edu

Abstract

The goal of our research is to improve event extraction by learning to identify secondary role filler contexts in the absence of event keywords. We propose a multi-layered event extraction architecture that progressively "zooms in" on relevant information. Our extraction model includes a document genre classifier to recognize event narratives, two types of sentence classifiers, and noun phrase classifiers to extract role fillers. These modules are organized as a pipeline to gradually zero in on event-related information. We present results on the MUC-4 event extraction data set and show that this model performs better than previous systems.

1 Introduction

Event extraction is an information extraction (IE) task that involves identifying the role fillers for events in a particular domain. For example, the Message Understanding Conferences (MUCs) challenged NLP researchers to create event extraction systems for domains such as terrorism (e.g., to identify the perpetrators, victims, and targets of terrorism events) and management succession (e.g., to identify the people and companies involved in corporate management changes).

Most event extraction systems use either a learning-based classifier to label words as role fillers, or lexico-syntactic patterns to extract role fillers from pattern contexts. Both approaches, however, generally tackle event recognition and role filler extraction at the same time. In other words, most event extraction systems primarily recognize contexts that explicitly refer to a relevant event. For example, a system that extracts information about murders will recognize expressions associated with murder (e.g., "killed", "assassinated", or "shot to death") and extract role fillers from the surrounding context. But many role fillers occur in contexts that do not explicitly mention the event, and those fillers are often overlooked. For example, the perpetrator of a murder may be mentioned in the context of an arrest, an eyewitness report, or speculation about possible suspects. Victims may be named in sentences that discuss the aftermath of the event, such as the identification of bodies, transportation of the injured to a hospital, or conclusions drawn from an investigation. We will refer to these types of sentences as "secondary contexts" because they are generally not part of the main event description. Discourse analysis is one option to explicitly link these secondary contexts to the event, but discourse modelling is itself a difficult problem.

The goal of our research is to improve event extraction by learning to identify secondary role filler contexts in the absence of event keywords. We create a set of classifiers to recognize *role-specific contexts* that suggest the presence of a likely role filler regardless of whether a relevant event is mentioned or not. For example, our model should recognize that a sentence describing an arrest probably includes a reference to a perpetrator, even though the crime itself is reported elsewhere.

Extracting information from these secondary contexts can be risky, however, unless we know that the larger context is discussing a relevant event. To

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1137–1147,
Portland, Oregon, June 19-24, 2011.

address this, we adopt a two-pronged strategy for event extraction that handles *event narrative* documents differently from other documents. We define an event narrative as an article whose main purpose is to report the details of an event. We apply the *role-specific sentence classifiers* only to event narratives to aggressively search for role fillers in these stories. However, other types of documents can mention relevant events too. The MUC-4 corpus, for example, includes interviews, speeches, and terrorist propaganda that contain information about terrorist events. We will refer to these documents as *fleeting reference* texts because they mention a relevant event somewhere in the document, albeit briefly. To ensure that relevant information is extracted from all documents, we also apply a conservative extraction process to every document to extract facts from explicit event sentences.

Our complete event extraction model, called TIER, incorporates both document genre and role-specific context recognition into 3 layers of analysis: document analysis, sentence analysis, and noun phrase (NP) analysis. At the top level, we train a text genre classifier to identify event narrative documents. At the middle level, we create two types of sentence classifiers. *Event sentence classifiers* identify sentences that are associated with relevant events, and *role-specific context classifiers* identify sentences that contain possible role fillers irrespective of whether an event is mentioned. At the lowest level, we use *role filler extractors* to label individual noun phrases as role fillers. As documents pass through the pipeline, they are analyzed at different levels of granularity. All documents pass through the event sentence classifier, and event sentences are given to the role filler extractors. Documents identified as event narratives additionally pass through role-specific sentence classifiers, and the role-specific sentences are also given to the role filler extractors. This multi-layered approach creates an event extraction system that can discover role fillers in a variety of different contexts, while maintaining good precision.

In the following sections, we position our research with respect to related work, present the details of our multi-layered event extraction model, and show experimental results for five event roles using the MUC-4 data set.

2 Related Work

Some event extraction data sets only include documents that describe relevant events (e.g., well-known data sets for the domains of corporate acquisitions (Freitag, 1998b; Freitag and McCallum, 2000; Finn and Kushmerick, 2004), job postings (Califf and Mooney, 2003; Freitag and McCallum, 2000), and seminar announcements (Freitag, 1998b; Ciravegna, 2001; Chieu and Ng, 2002; Finn and Kushmerick, 2004; Gu and Cercone, 2006). But many IE data sets present a more realistic task where the IE system must determine whether a relevant event is present in the document, and if so, extract its role fillers. Most of the Message Understanding Conference data sets represent this type of event extraction task, containing (roughly) a 50/50 mix of relevant and irrelevant documents (e.g., MUC-3, MUC-4, MUC-6, and MUC-7 (Hirschman, 1998)). Our research focuses on this setting where the event extraction system is not assured of getting only relevant documents to process.

Most event extraction models can be characterized as either pattern-based or classifier-based approaches. Early event extraction systems used handcrafted patterns (e.g., (Appelt et al., 1993; Lehnert et al., 1991)), but more recent systems generate patterns or rules automatically using supervised learning (e.g., (Kim and Moldovan, 1993; Riloff, 1993; Soderland et al., 1995; Huffman, 1996; Freitag, 1998b; Ciravegna, 2001; Califf and Mooney, 2003)), weakly supervised learning (e.g., (Riloff, 1996; Riloff and Jones, 1999; Yangarber et al., 2000; Sudo et al., 2003; Stevenson and Greenwood, 2005)), or unsupervised learning (e.g., (Shinyama and Sekine, 2006; Sekine, 2006)). In addition, many classifiers have been created to sequentially label event role fillers in a sentence (e.g., (Freitag, 1998a; Chieu and Ng, 2002; Finn and Kushmerick, 2004; Li et al., 2005; Yu et al., 2005)). Research has also been done on relation extraction (e.g., (Roth and Yih, 2001; Zelenko et al., 2003; Bunescu and Mooney, 2007)), but that task is different from event extraction because it focuses on isolated relations rather than template-based event analysis.

Most event extraction systems scan a text and search small context windows using patterns or a classifier. However, recent work has begun to ex-

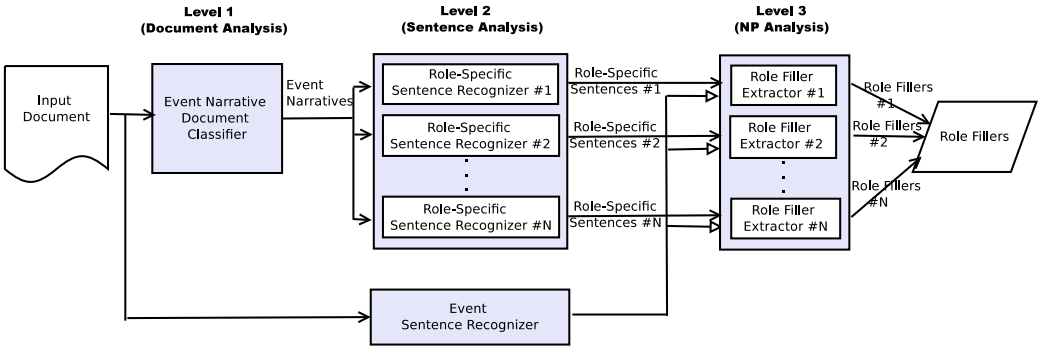


Figure 1: TIER: A Multi-Layered Architecture for Event Extraction

plore more global approaches. (Maslennikov and Chua, 2007) use discourse trees and local syntactic dependencies in a pattern-based framework to incorporate wider context. Ji and Grishman (2008) enforce event role consistency across different documents. (Liao and Grishman, 2010) use cross-event inference to help with the extraction of role fillers shared across events. And there have been several recent IE models that explore the idea of identifying relevant sentences to gain a wider contextual view and then extracting role fillers. (Gu and Cercone, 2006) created HMMs to first identify relevant sentences, but their research focused on eliminating redundant extractions and worked with seminar announcements, where the system was only given relevant documents. (Patwardhan and Riloff, 2007) developed a system that learns to recognize event sentences and uses patterns that have a *semantic affinity* for an event role to extract role fillers. GLACIER (Patwardhan and Riloff, 2009) jointly considers sentential evidence and phrasal evidence in a unified probabilistic framework. Our research follows in the same spirit as these approaches by performing multiple levels of text analysis. But our event extraction model includes two novel contributions: (1) we develop a set of *role-specific* sentence classifiers to learn to recognize *secondary contexts* associated with each type of event role , and (2) we exploit text genre to incorporate a third level of analysis that enables the system to aggressively hunt for role fillers in documents that are event narratives. In Section 5, we compare the performance of our model with both the GLACIER system and Patwardhan & Riloff's semantic affinity model.

3 A Multi-Layered Approach to Event Extraction

The main idea behind our approach is to analyze documents at multiple levels of granularity in order to identify role fillers that occur in different types of contexts. Our event extraction model progressively "zooms in" on relevant information by first identifying the document type, then identifying sentences that are likely to contain relevant information, and finally analyzing individual noun phrases to identify role fillers. The key advantage of this architecture is that it allows us to search for information using two different principles: (1) we look for contexts that directly refer to the event, as per most traditional event extraction systems, and (2) we look for secondary contexts that are often associated with a specific type of role filler. Identifying these *role-specific contexts* can root out important facts would have been otherwise missed. Figure 1 shows the multi-layered pipeline of our event extraction system.

An important aspect of our model is that two different strategies are employed to handle documents of different types. The event extraction task is to find any description of a relevant event, even if the event is not the topic of the article.[1] Consequently, all documents are given to the event sentence recognizers and their mission is to identify any sentence that mentions a relevant event. This path through the pipeline is conservative because information is extracted only from event sentences, but all documents are processed, including stories that contain only a fleeting reference to a relevant event.

[1]Per the MUC-4 task definition (MUC-4 Proceedings, 1992).

The second path through the pipeline performs additional processing for documents that belong to the event narrative text genre. For event narratives, we assume that most of the document discusses a relevant event so we can more aggressively hunt for event-related information in secondary contexts.

In this section, we explain how we create the two types of sentence classifiers and the role filler extractors. We will return to the issue of document genre and the event narrative classifier in Section 4.

3.1 Sentence Classification

We have argued that event role fillers commonly occur in two types of contexts: event contexts and role-specific secondary contexts. For the purposes of this research, we use sentences as our definition of a "context", although there are obviously many other possible definitions. An *event context* is a sentence that describes the actual event. A *secondary context* is a sentence that provides information related to an event but in the context of other activities that precede or follow the event.

For both types of classifiers, we use exactly the same feature set, but we train them in different ways. The MUC-4 corpus used in our experiments includes a training set consisting of documents and answer keys. Each document that describes a relevant event has answer key templates with the role fillers (*answer key strings*) for each event. To train the event sentence recognizer, we consider a sentence to be a positive training instance if it contains one or more answer key strings from any of the event roles. This produced 3,092 positive training sentences. All remaining sentences that do not contain any answer key strings are used as negative instances. This produced 19,313 negative training sentences, yielding a roughly 6:1 ratio of negative to positive instances.

There is no guarantee that a classifier trained in this way will identify event sentences, but our hypothesis was that training across all of the event roles together would produce a classifier that learns to recognize general event contexts. This approach was also used to train GLACIER's sentential event recognizer (Patwardhan and Riloff, 2009), and they demonstrated that this approach worked reasonably well when compared to training with event sentences labelled by human judges.

The main contribution of our work is introducing additional *role-specific sentence classifiers* to seek out role fillers that appear in less obvious secondary contexts. We train a set of role-specific sentence classifiers, one for each type of event role. Every sentence that contains a role filler of the appropriate type is used as a positive training instance. Sentences that do not contain any answer key strings are negative instances.[2] In this way, we force each classifier to focus on the contexts specific to its particular event role. We expect the role-specific sentence classifiers to find some secondary contexts that the event sentence classifier will miss, although some sentences may be classified as both.

Using all possible negative instances would produce an extremely skewed ratio of negative to positive instances. To control the skew and keep the training set-up consistent with the event sentence classifier, we randomly choose from the negative instances to produce a 6:1 ratio of negative to positive instances.

Both types of classifiers use an SVM model created with SVMlin (Keerthi and DeCoste, 2005), and exactly the same features. The feature set consists of the unigrams and bigrams that appear in the training texts, the semantic class of each noun phrase[3], plus a few additional features to represent the tense of the main verb phrase in the sentence and whether the document is long (> 35 words) or short (< 5 words). All of the feature values are binary.

3.2 Role Filler Extractors

Our extraction model also includes a set of role filler extractors, one per event role. Each extractor receives a sentence as input and determines which noun phrases (NPs) in the sentence are fillers for the event role. To train an SVM classifier, noun phrases corresponding to answer key strings for the event role are positive instances. We randomly choose among all noun phrases that are not in the answer keys to create a 10:1 ratio of negative to positive instances.

[2] We intentionally do not use sentences that contain fillers for competing event roles as negative instances because sentences often contain multiple role fillers of different types (e.g., a weapon may be found near a body). Sentences without any role fillers are certain to be irrelevant contexts.

[3] We used the Sundance parser (Riloff and Phillips, 2004) to identify noun phrases and assign semantic class labels.

The feature set for the role filler extractors is much richer than that of the sentence classifiers because they must carefully consider the local context surrounding a noun phrase. We will refer to the noun phrase being labelled as the *targeted NP*. The role filler extractors use three types of features:

Lexical features: we represent four words to the left and four words to the right of the targeted NP, as well as the head noun and modifiers (adjectives and noun modifiers) of the targeted NP itself.

Lexico-syntactic patterns: we use the AutoSlog pattern generator (Riloff, 1993) to automatically create lexico-syntactic patterns around each noun phrase in the sentence. These patterns are similar to dependency relations in that they typically represent the syntactic role of the NP with respect to other constituents (e.g., subject-of, object-of, and noun arguments).

Semantic features: we use the Stanford NER tagger (Finkel et al., 2005) to determine if the targeted NP is a named entity, and we use the Sundance parser (Riloff and Phillips, 2004) to assign semantic class labels to each NP's head noun.

4 Event Narrative Document Classification

One of our goals was to explore the use of *document genre* to permit more aggressive strategies for extracting role fillers. In this section, we first present an analysis of the MUC-4 data set which reveals the distribution of event narratives in the corpus, and then explain how we train a classifier to automatically identify event narrative stories.

4.1 Manual Analysis

We define an *event narrative* as an article whose main focus is on reporting the details of an event. For the purposes of this research, we are only concerned with events that are relevant to the event extraction task (i.e., terrorism). An *irrelevant document* is an article that does not mention any relevant events. In between these extremes is another category of documents that briefly mention a relevant event, but the event is not the focus of the article. We will refer to these documents as *fleeting reference* documents. Many of the fleeting reference documents in the MUC-4 corpus are transcripts of interviews, speeches, or terrorist propaganda communiques that refer to a terrorist event and mention at least one role filler, but within a discussion about a different topic (e.g., the political ramifications of a terrorist incident).

To gain a better understanding of how we might create a system to automatically distinguish event narrative documents from fleeting reference documents, we manually labelled the 116 relevant documents in our tuning set. This was an informal study solely to help us understand the nature of these texts.

	# of Event Narratives	# of Fleeting Ref. Docs	Acc
Gold Standard	54	62	
Heuristics	40	55	.82

Table 1: Manual Analysis of Document Types

The first row of Table 1 shows the distribution of event narratives and fleeting references based on our "gold standard" manual annotations. We see that more than half of the relevant documents (62/116) are *not* focused on reporting a terrorist event, even though they contain information about a terrorist event somewhere in the document.

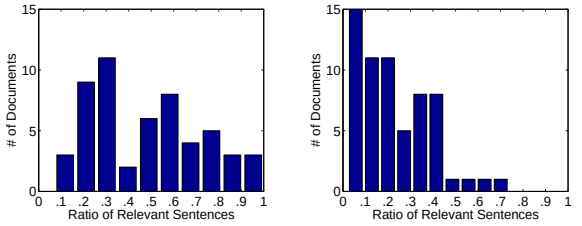
4.2 Heuristics for Event Narrative Identification

Our goal is to train a document classifier to automatically identify event narratives. The MUC-4 answer keys reveal which documents are relevant and irrelevant with respect to the terrorism domain, but they do not tell us which relevant documents are event narratives and which are fleeting reference stories. Based on our manual analysis of the tuning set, we developed several heuristics to help separate them.

We observed two types of clues: the location of the relevant information, and the density of relevant information. First, we noticed that event narratives tend to mention relevant information within the first several sentences, whereas fleeting reference texts usually mention relevant information only in the middle or end of the document. Therefore our first heuristic requires that an event narrative mention a role filler within the first 7 sentences.

Second, event narratives generally have a higher density of relevant information. We use several criteria to estimate information density because a single criterion was inadequate to cover different sce-

narios. For example, some documents mention role fillers throughout the document. Other documents contain a high concentration of role fillers in some parts of the document but no role fillers in other parts. We developed three density heuristics to account for different situations. All of these heuristics count distinct role fillers. The first density heuristic requires that more than 50% of the sentences contain at least one role filler ($\frac{|RelSents|}{|AllSents|} > 0.5$) . Figure 2 shows histograms for different values of this ratio in the event narrative (a) vs. the fleeting reference documents (b). The histograms clearly show that documents with a high ($> 50\%$) ratio are almost always event narratives.



(a) (b)

Figure 2: Histograms of Density Heuristic #1 in Event Narratives (a) vs. Fleeting References (b).

A second density heuristic requires that the ratio of different *types* of roles filled to sentences be $>$ 50% ($\frac{|Roles|}{|AllSents|} > 0.5$). A third density heuristic requires that the ratio of distinct role *fillers* to sentences be $> 70\%$ ($\frac{|RoleFillers|}{|AllSents|} > 0.7$). If any of these three criteria are satisfied, then the document is considered to have a high density of relevant information.[4]

We use these heuristics to label a document as an event narrative if: (1) it has a high density of relevant information, and (2) it mentions a role filler within the first 7 sentences.

The second row of Table 1 shows the performance of these heuristics on the tuning set. The heuristics correctly identify $\frac{40}{54}$ event narratives and $\frac{55}{62}$ fleeting reference stories, to achieve an overall accuracy of 82%. These results are undoubtedly optimistic because the heuristics were derived from analysis of the tuning set. But we felt confident enough to move forward with using these heuristics to generate training data for an event narrative classifier.

4.3 Event Narrative Classifier

The heuristics above use the answer keys to help determine whether a story belongs to the event narrative genre, but our goal is to create a classifier that can identify event narrative documents without the benefit of answer keys. So we used the heuristics to automatically create training data for a classifier by labelling each relevant document in the training set as an event narrative or a fleeting reference document. Of the 700 relevant documents, 292 were labeled as event narratives. We then trained a document classifier using the 292 event narrative documents as positive instances and all irrelevent training documents as negative instances. The 308 relevant documents that were not identified as event narratives were discarded to minimize noise (i.e., we estimate that our heuristics fail to identify 25% of the event narratives). We then trained an SVM classifier using bag-of-words (unigram) features.

Table 2 shows the performance of the event narrative classifier on the manually labeled tuning set. The classifier identified 69% of the event narratives with 63% precision. Overall accuracy was 81%.

Recall	Precision	Accuracy
.69	.63	.81

Table 2: Event Narrative Classifier Results

At first glance, the performance of this classifier is mediocre. However, these results should be interpreted loosely because there is not always a clear dividing line between event narratives and other documents. For example, some documents begin with a specific event description in the first few paragraphs but then digress to discuss other topics. Fortunately, it is not essential for TIER to have a perfect event narrative classifier since all documents will be processed by the event sentence recognizer anyway. The recall of the event narrative classifier means that nearly 70% of the event narratives will get additional scrutiny, which should help to find additional role fillers. Its precision of 63% means that some documents that are not event narratives will also get additional scrutiny, but information will be extracted only if both the role-specific sentence recognizer and NP extractors believe they have found

[4]Heuristic #1 covers most of the event narratives.

Method	PerpInd	PerpOrg	Target	Victim	Weapon	Average
Baselines						
AutoSlog-TS	33/49/40	52/33/41	54/59/56	49/54/51	38/44/41	45/48/46
Semantic Affinity	48/39/43	36/58/45	56/46/50	46/44/45	53/46/50	48/47/47
GLACIER	51/58/**54**	34/45/38	43/72/53	55/58/56	57/53/55	48/57/52
New Results without document classification						
AllSent	25/67/36	26/78/39	34/83/49	32/72/45	30/75/43	30/75/42
EventSent	52/54/53	50/44/47	52/67/59	55/51/53	56/57/56	53/54/54
RoleSent	37/54/44	37/58/45	49/75/59	52/60/55	38/66/48	43/63/51
EventSent+RoleSent	38/60/46	36/63/46	47/78/59	52/64/57	36/66/47	42/66/51
New Results with document classification						
DomDoc/EventSent+DomDoc/RoleSent	45/54/49	42/51/46	51/68/58	54/56/55	46/63/53	48/58/52
EventSent+DomDoc/RoleSent	43/59/50	45/61/**52**	51/77/**61**	52/61/56	44/66/53	47/65/54
EventSent+ENarrDoc/RoleSent	48/57/52	46/53/50	51/73/60	56/60/**58**	53/64/**58**	51/62/**56**

Table 3: Experimental results, reported as Precision/Recall/F-score

something relevant.

4.4 Domain-relevant Document Classifier

For comparison's sake, we also created a document classifier to identify *domain-relevant* documents. That is, we trained a classifier to determine whether a document is relevant to the domain of terrorism, irrespective of the style of the document. We trained an SVM classifier with the same bag-of-words feature set, using all relevant documents in the training set as positive instances and all irrelevant documents as negative instances. We use this classifier for several experiments described in the next section.

5 Evaluation

5.1 Data Set and Metrics

We evaluated our approach on a standard benchmark collection for event extraction systems, the MUC-4 data set (MUC-4 Proceedings, 1992). The MUC-4 corpus consists of 1700 documents with associated answer key templates. To be consistent with previously reported results on this data set, we use the 1300 DEV documents for training, 200 documents (TST1+TST2) as a tuning set and 200 documents (TST3+TST4) as the test set. Roughly half of the documents are relevant (i.e., they mention at least 1 terrorist event) and the rest are irrelevant.

We evaluate our system on the five MUC-4 "string-fill" event roles: *perpetrator individuals*, *perpetrator organizations*, *physical targets*, *victims* and *weapons*. The complete IE task involves template generation, which is complex because many documents have multiple templates (i.e., they discuss multiple events). Our work focuses on extracting individual facts and not on template generation per se (e.g., we do not perform coreference resolution or event tracking). Consequently, our evaluation follows that of other recent work and evaluates the accuracy of the extractions themselves by matching the head nouns of extracted NPs with the head nouns of answer key strings (e.g., "armed guerrillas" is considered to match "guerrillas")[5]. Our results are reported as Precision/Recall/F(1)-score for each event role separately. We also show an overall average for all event roles combined.[6]

5.2 Baselines

As baselines, we compare the performance of our IE system with three other event extraction systems. The first baseline is AutoSlog-TS (Riloff, 1996), which uses domain-specific extraction patterns. AutoSlog-TS applies its patterns to every sentence in every document, so does not attempt to explicitly identify relevant sentences or documents. The next two baselines are more recent systems: the (Patwardhan and Riloff, 2007) *semantic affinity* model and the (Patwardhan and Riloff, 2009) GLACIER system. The *semantic affinity* approach

[5]Pronouns were discarded since we do not perform coreference resolution. Duplicate extractions with the same head noun were counted as one hit or one miss.

[6]We generated the Average scores ourselves by macro-averaging over the scores reported for the individual event roles.

explicitly identifies event sentences and uses patterns that have a semantic affinity for an event role to extract role fillers. GLACIER is a probabilistic model that incorporates both phrasal and sentential evidence jointly to label role fillers.

The first 3 rows in Table 3 show the results for each of these systems on the MUC-4 data set. They all used the same evaluation criteria as our results.

5.3 Experimental Results

The lower portion of Table 3 shows the results of a variety of event extraction models that we created using different components of our system. The **AllSent** row shows the performance of our Role Filler Extractors when applied to every sentence in every document. This system produced high recall, but precision was consistently low.

The **EventSent** row shows the performance of our Role Filler Extractors applied only to the *event sentences* identified by our event sentence classifier. This boosts precision across all event roles, but with a sharp reduction in recall. We see a roughly 20 point swing from recall to precision. These results are similar to GLACIER's results on most event roles, which isn't surprising because GLACIER also incorporates event sentence identification.

The **RoleSent** row shows the results of our Role Filler Extractors applied only to the *role-specific sentences* identified by our classifiers. We see a 12-13 point swing from recall to precision compared to the **AllSent** row. This result is consistent with our hypothesis that many role fillers exist in role-specific contexts that are not event sentences. As expected, extracting facts from role-specific contexts that do not necessarily refer to an event is less reliable. The **EventSent+RoleSent** row shows the results when information is extracted from both types of sentences. We see slightly higher recall, which confirms that one set of extractions is not a strict subset of the other, but precision is still relatively low.

The next set of experiments incorporates document classification as the third layer of text analysis. The **DomDoc/EventSent+DomDoc/RoleSent** row shows the results of applying both types of sentence classifiers only to documents identified as domain-relevant by the Domain-relevant Document (**DomDoc**) Classifier described in Section 4.4. Extracting information only from domain-relevant documents improves precision by $+6$, but also sacrifices 8 points of recall.

The **EventSent** row reveals that information found in event sentences has the highest precision, even without relying on document classification. We concluded that evidence of an event sentence is probably sufficient to warrant role filler extraction irrespective of the style of the document. As we discussed in Section 4, many documents contain only a fleeting reference to an event, so it is important to be able to extract information from those isolated event descriptions as well. Consequently, we created a system, **EventSent+DomDoc/RoleSent**, that extracts information from event sentences in *all* documents, but extracts information from role-specific sentences only if they appear in a domain-relevant document. This architecture captured the best of both worlds: recall improved from 58% to 65% with only a one point drop in precision.

Finally, we evaluated the idea of using document *genre* as a filter instead of domain relevance. The last row, **EventSent+ENarrDoc/RoleSent**, shows the results of our final architecture which extracts information from event sentences in all documents, but extracts information from role-specific sentences only in Event Narrative documents. This architecture produced the best F1 score of 56. This model increases precision by an additional 4 points and produces the best balance of recall and precision.

Overall, TIER's multi-layered extraction architecture produced higher F1 scores than previous systems on four of the five event roles. The improved recall is due to the additional extractions from secondary contexts. The improved precision comes from our two-pronged strategy of treating event narratives differently from other documents. TIER aggressively searches for extractions in event narrative stories but is conservative and extracts information only from event sentences in all other documents.

5.4 Analysis

We looked through some examples of TIER's output to try to gain insight about its strengths and limitations. TIER's role-specific sentence classifiers did correctly identify some sentences containing role fillers that were not classified as event sentences. Several examples are shown below, with the role

fillers in italics:

(1) "The victims were identified as *David Lecky*, director of the Columbus school, and *James Arthur Donnelly*."

(2) "There were *seven children*, including *four of the Vice President's children*, in the home at the time."

(3) "*The woman* fled and sought refuge inside the facilities of the Salvadoran Alberto Masferrer University, where she took a group of *students* as hostages, threatening them with *hand grenades*."

(4) "The FMLN stated that *several homes* were damaged and that animals were killed in the surrounding hamlets and villages."

The first two sentences identify victims, but the terrorist event itself was mentioned earlier in the document. The third sentence contains a perpetrator (*the woman*), victims (*students*), and weapons (*hand grenades*) in the context of a hostage situation after the main event (a bus attack), when the perpetrator escaped. The fourth sentence describes incidental damage to civilian homes following clashes between government forces and guerrillas.

However there is substantial room for improvement in each of TIER's subcomponents, and many role fillers are still overlooked. One reason is that it can be difficult to recognize acts of terrorism. Many sentences refer to a potentially relevant subevent (e.g., injury or physical damage) but recognizing that the event is part of a terrorist incident depends on the larger discourse. For example, consider the examples below that TIER did not recognize as relevant sentences:

(5) "Later, *two individuals* in a Chevrolet Opala automobile pointed AK rifles at the students, fired some shots, and quickly drove away."

(6) "Meanwhile, national police members who were dressed in civilian clothes seized university students *Hugo Martinez* and *Raul Ramirez*, who are still missing."

(7) "*All labor union offices* in San Salvador were looted."

In the first sentence, the event is described as someone pointing rifles at people and the perpetrators are referred to simply as individuals. There are no strong keywords in this sentence that reveal this is a terrorist attack. In the second sentence, police are being accused of state-sponsored terrorism when they seize civilians. The verb "seize" is common in this corpus, but usually refers to the seizing of weapons or drug stashes, not people. The third sentence describes a looting subevent. Acts of looting and vandalism are not usually considered to be terrorism, but in this article it is in the context of accusations of terrorist acts by government officials.

6 Conclusions

We have presented a new approach to event extraction that uses three levels of analysis: document genre classification to identify event narrative stories, two types of sentence classifiers, and noun phrase classifiers. A key contribution of our work is the creation of role-specific sentence classifiers that can detect role fillers in secondary contexts that do not directly refer to the event. Another important aspect of our approach is a two-pronged strategy that handles event narratives differently from other documents. TIER aggressively hunts for role fillers in event narratives, but is conservative about extracting information from other documents. This strategy produced improvements in both recall and precision over previous state-of-the-art systems.

This work just scratches the surface of using document genre identification to improve information extraction accuracy. In future work, we hope to identify additional types of document genre styles and incorporate genre directly into the extraction model. Coreference resolution and discourse analysis will also be important to further improve event extraction performance.

7 Acknowledgments

We gratefully acknowledge the support of the National Science Foundation under grant IIS-1018314 and the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0172. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the U.S. government.

References

D. Appelt, J. Hobbs, J. Bear, D. Israel, and M. Tyson. 1993. FASTUS: a finite-state processor for information extraction from real-world text. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*.

R. Bunescu and R. Mooney. 2007. Learning to Extract Relations from the Web using Minimal Supervision. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

M.E. Califf and R. Mooney. 2003. Bottom-up Relational Learning of Pattern Matching rules for Information Extraction. *Journal of Machine Learning Research*, 4:177–210.

H.L. Chieu and H.T. Ng. 2002. A Maximum Entropy Approach to Information Extraction from Semi-Structured and Free Text. In *Proceedings of the 18th National Conference on Artificial Intelligence*.

F. Ciravegna. 2001. Adaptive Information Extraction from Text by Rule Induction and Generalisation. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*.

J. Finkel, T. Grenager, and C. Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 363–370, Ann Arbor, MI, June.

A. Finn and N. Kushmerick. 2004. Multi-level Boundary Classification for Information Extraction. In *In Proceedings of the 15th European Conference on Machine Learning*, pages 111–122, Pisa, Italy, September.

D. Freitag and A. McCallum. 2000. Information Extraction with HMM Structures Learned by Stochastic Optimization. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 584–589, Austin, TX, August.

Dayne Freitag. 1998a. Multistrategy Learning for Information Extraction. In *Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers.

Dayne Freitag. 1998b. Toward General-Purpose Learning for Information Extraction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*.

Z. Gu and N. Cercone. 2006. Segment-Based Hidden Markov Models for Information Extraction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 481–488, Sydney, Australia, July.

L. Hirschman. 1998. "The Evolution of Evaluation: Lessons from the Message Understanding Conferences. *Computer Speech and Language*, 12.

S. Huffman. 1996. Learning Information Extraction Patterns from Examples. In Stefan Wermter, Ellen Riloff, and Gabriele Scheler, editors, *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, pages 246–260. Springer-Verlag, Berlin.

H. Ji and R. Grishman. 2008. Refining Event Extraction through Cross-Document Inference. In *Proceedings of ACL-08: HLT*, pages 254–262, Columbus, OH, June.

S. Keerthi and D. DeCoste. 2005. A Modified Finite Newton Method for Fast Solution of Large Scale Linear SVMs. *Journal of Machine Learning Research*.

J. Kim and D. Moldovan. 1993. Acquisition of Semantic Patterns for Information Extraction from Corpora. In *Proceedings of the Ninth IEEE Conference on Artificial Intelligence for Applications*, pages 171–176, Los Alamitos, CA. IEEE Computer Society Press.

W. Lehnert, C. Cardie, D. Fisher, E. Riloff, and R. Williams. 1991. University of Massachusetts: Description of the CIRCUS System as Used for MUC-3. In *Proceedings of the Third Message Understanding Conference (MUC-3)*, pages 223–233, San Mateo, CA. Morgan Kaufmann.

Y. Li, K. Bontcheva, and H. Cunningham. 2005. Using Uneven Margins SVM and Perceptron for Information Extraction. In *Proceedings of Ninth Conference on Computational Natural Language Learning*, pages 72–79, Ann Arbor, MI, June.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48st Annual Meeting on Association for Computational Linguistics (ACL-10)*.

M. Maslennikov and T. Chua. 2007. A Multi-Resolution Framework for Information Extraction from Free Text. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

MUC-4 Proceedings. 1992. *Proceedings of the Fourth Message Understanding Conference (MUC-4)*. Morgan Kaufmann.

S. Patwardhan and E. Riloff. 2007. Effective Information Extraction with Semantic Affinity Patterns and Relevant Regions. In *Proceedings of 2007 the Conference on Empirical Methods in Natural Language Processing (EMNLP-2007)*.

S. Patwardhan and E. Riloff. 2009. A Unified Model of Phrasal and Sentential Evidence for Information Extraction. In *Proceedings of 2009 the Conference on Empirical Methods in Natural Language Processing (EMNLP-2009)*.

E. Riloff and R. Jones. 1999. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*.

E. Riloff and W. Phillips. 2004. An Introduction to the Sundance and AutoSlog Systems. Technical Report UUCS-04-015, School of Computing, University of Utah.

E. Riloff. 1993. Automatically Constructing a Dictionary for Information Extraction Tasks. In *Proceedings of the 11th National Conference on Artificial Intelligence*.

E. Riloff. 1996. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044–1049. The AAAI Press/MIT Press.

D. Roth and W. Yih. 2001. Relational Learning via Propositional Algorithms: An Information Extraction Case Study. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 1257–1263, Seattle, WA, August.

Satoshi Sekine. 2006. On-demand information extraction. In *Proceedings of Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL-06*.

Y. Shinyama and S. Sekine. 2006. Preemptive Information Extraction using Unrestricted Relation Discovery. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 304–311, New York City, NY, June.

S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert. 1995. CRYSTAL: Inducing a conceptual dictionary. In *Proc. of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1314–1319.

M. Stevenson and M. Greenwood. 2005. A Semantic Approach to IE Pattern Induction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 379–386, Ann Arbor, MI, June.

K. Sudo, S. Sekine, and R. Grishman. 2003. An Improved Extraction Pattern Representation Model for Automatic IE Pattern Acquisition. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*.

R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. 2000. Automatic Acquisition of Domain Knowledge for Information Extraction. In *Proceedings of the Eighteenth International Conference on Computational Linguistics (COLING 2000)*.

K. Yu, G. Guan, and M. Zhou. 2005. Resumé Information Extraction with Cascaded Hybrid Model. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 499–506, Ann Arbor, MI, June.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel Methods for Relation Extraction. *Journal of Machine Learning Research*, 3.

Knowledge Base Population: Successful Approaches and Challenges

Heng Ji
Computer Science Department
Queens College and Graduate Center
City University of New York
New York, NY 11367, USA
hengji@cs.qc.cuny.edu

Ralph Grishman
Computer Science Department
New York University

New York, NY 10003, USA
grishman@cs.nyu.edu

Abstract

In this paper we give an overview of the Knowledge Base Population (KBP) track at the 2010 Text Analysis Conference. The main goal of KBP is to promote research in discovering facts about entities and augmenting a knowledge base (KB) with these facts. This is done through two tasks, *Entity Linking* – linking names in context to entities in the KB – and *Slot Filling* – adding information about an entity to the KB. A large source collection of newswire and web documents is provided from which systems are to discover information. Attributes ("slots") derived from Wikipedia infoboxes are used to create the reference KB. In this paper we provide an overview of the techniques which can serve as a basis for a good KBP system, lay out the remaining challenges by comparison with traditional Information Extraction (IE) and Question Answering (QA) tasks, and provide some suggestions to address these challenges.

1 Introduction

Traditional information extraction (IE) evaluations, such as the Message Understanding Conferences (MUC) and Automatic Content Extraction (ACE), assess the ability to extract information from individual documents in isolation. In practice, however, we may need to gather information about a person or organization that is scattered among the documents of a large collection. This requires the ability to identify the relevant documents and to integrate facts, possibly redundant, possibly complementary, possibly in conflict, coming from these documents. Furthermore, we may want to use the extracted information to *augment* an existing data base. This requires the ability to link individuals mentioned in a document, and information about these individuals, to entries in the data base. On the other hand, traditional Question Answering (QA) evaluations made limited efforts at disambiguating entities in queries (e.g. Pizzato et al., 2006), and limited use of relation/event extraction in answer search (e.g. McNamee et al., 2008).

The Knowledge Base Population (KBP) shared task, conducted as part of the NIST Text Analysis Conference, aims to address and evaluate these capabilities, and bridge the IE and QA communities to promote research in discovering facts about entities and expanding a knowledge base with these facts. KBP is done through two separate subtasks, Entity Linking and Slot Filling; in 2010, 23 teams submitted results for one or both sub-tasks. A variety of approaches have been proposed to address both tasks with considerable success; nevertheless, there are many aspects of the task that remain unclear. What are the fundamental techniques used to achieve reasonable performance? What is the impact of each novel method? What types of problems are represented in the current KBP paradigm compared to traditional IE and QA? In which way have the current testbeds and evaluation methodology affected our perception of the task difficulty? Have we reached a performance ceiling with current state of the art techniques? What are the remaining challenges and what are the possible ways to address these challenges? In this paper we aim to answer some of these questions based on our detailed analysis of evaluation results.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1148–1158,
Portland, Oregon, June 19-24, 2011.

2 Task Definition and Evaluation Metrics

This section will summarize the tasks conducted at KBP 2010. The overall goal of KBP is to automatically identify salient and novel entities, link them to corresponding Knowledge Base (KB) entries (if the linkage exists), then discover attributes about the entities, and finally expand the KB with any new attributes.

In the Entity Linking task, given a person (PER), organization (ORG) or geo-political entity (GPE, a location with a government) query that consists of a name string and a background document containing that name string, the system is required to provide the ID of the KB entry to which the name refers; or NIL if there is no such KB entry. The background document, drawn from the KBP corpus, serves to disambiguate ambiguous name strings.

In selecting among the KB entries, a system could make use of the Wikipedia text associated with each entry as well as the structured fields of each entry. In addition, there was an optional task where the system could only make use of the structured fields; this was intended to be representative of applications where no backing text was available. Each site could submit up to three runs with different parameters.

The goal of Slot Filling is to collect from the corpus information regarding certain attributes of an entity, which may be a person or some type of organization. Each query in the Slot Filling task consists of the name of the entity, its type (person or organization), a background document containing the name (again, to disambiguate the query in case there are multiple entities with the same name), its node ID (if the entity appears in the knowledge base), and the attributes which need not be filled. Attributes are excluded if they are already filled in the reference data base and can only take on a single value. Along with each slot fill, the system must provide the ID of a document which supports the correctness of this fill. If the corpus does not provide any information for a given attribute, the system should generate a NIL response (and no document ID). KBP2010 defined 26 types of attributes for persons (such as the age, birthplace, spouse, children, job title, and employing organization) and 16 types of attributes for organizations (such as the top employees, the founder, the year founded, the headquarters location, and subsidiaries). Some of these attributes are specified as only taking a single value (e.g., birthplace), while some can take multiple values (e.g., top employees).

The reference KB includes hundreds of thousands of entities based on articles from an October 2008 dump of English Wikipedia which includes 818,741 nodes. The source collection includes 1,286,609 newswire documents, 490,596 web documents and hundreds of transcribed spoken documents.

To score Entity Linking, we take each query and check whether the KB node ID (or NIL) returned by a system is correct or not. Then we compute the Micro-averaged Accuracy, computed across all queries.

To score Slot Filling, we first pool all the system responses (as is done for information retrieval evaluations) together with a set of manually-prepared slot fills. These responses are then assessed by hand. Equivalent answers (such as "Bill Clinton" and "William Jefferson Clinton") are grouped into equivalence classes. Each system response is rated as correct, wrong, or redundant (a response which is equivalent to another response for the same slot or an entry already in the knowledge base). Given these judgments, we count

Correct = total number of non-NIL system output slots judged correct

System = total number of non-NIL system output slots

Reference = number of single-valued slots with a correct non-NIL response + number of equivalence classes for all list-valued slots

Recall = Correct / Reference

Precision = Correct / System

F-Measure = (2 × Recall × Precision) / (Recall + Precision)

3 Entity Linking: What Works

In Entity Linking, we saw a general improvement in performance over last year's results – the top system achieved 85.78% micro-averaged accuracy. When measured against a benchmark based on inter-annotator agreement, two systems' performance approached and one system exceeded the benchmark on person entities.

3.1 A General Architecture

A typical entity linking system architecture is depicted in Figure 1.

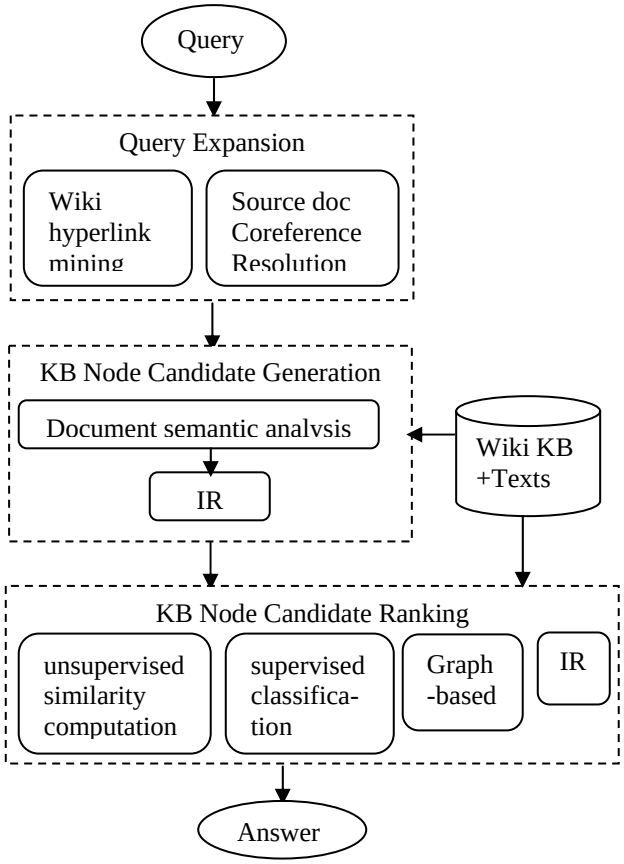


Figure 1. General Entity Linking System Architecture

It includes three steps: (1) query expansion – expand the query into a richer set of forms using Wikipedia structure mining or coreference resolution in the background document. (2) candidate generation – finding all possible KB entries that a query might link to; (3) candidate ranking – rank the probabilities of all candidates and NIL answer.

Table 1 summarizes the systems which exploited different approaches at each step. In the following subsections we will highlight the new and effective techniques used in entity linking.

3.2 Wikipedia Structure Mining

Wikipedia articles are peppered with structured information and hyperlinks to other (on average 25) articles (Medelyan et al., 2009). Such information provides additional sources for entity linking: (1). Query Expansion: For example, WebTLab (Fernandez et al., 2010) used Wikipedia link structure (source, anchors, redirects and disambiguation) to extend the KB and compute entity co-occurrence estimates. Many other teams including CUNY and Siel used redirect pages and disambiguation pages for query expansion. The Siel team also exploited bold texts from first paragraphs because they often contain nicknames, alias names and full names.

Methods		System Examples	System Ranking Range
Query Expansion	Wikipedia Hyperlink Mining	CUNY (Chen et al., 2010), NUSchime (Zhang et al., 2010), Siel (Bysani et al., 2010), SMU-SIS (Gottipati et al., 2010), USFD (Yu et al., 2010), WebTLab team (Fernandez et al., 2010)	[2, 15]
	Source document coreference resolution	CUNY (Chen et al., 2010)	9
Candidate Generation	Document semantic analysis and context modeling	ARPANI (Thomas et al., 2010), CUNY (Chen et al., 2010), LCC (Lehmann et al., 2010)	[1,14]
	IR	CUNY (Chen et al., 2010), Budapestacad (Nemeskey et al., 2010), USFD (Yu et al., 2010)	[9, 16]
Candidate Ranking	Unsupervised Similarity Computation (e.g. VSM)	CUNY (Chen et al., 2010), SMU-SIS (Gottipati et al., 2010), USFD (Yu et al., 2010)	[9, 14]
	Supervised Classification	LCC (Lehmann et al., 2010), NUSchime (Zhang et al., 2010), Stanford-UBC (Chang et al., 2010), HLTCOE (McNamee, 2010), UC3M (Pablo-Sanchez et al., 2010)	[1, 10]
	Rule-based	LCC (Lehmann et al., 2010), BuptPris (Gao et al., 2010)	[1, 8]
	Global Graph-based Ranking	CMCRC (Radford et al., 2010)	3
	IR	Budapestacad (Nemeskey et al., 2010)	16

Table 1. Entity Linking Method Comparison

(2). Candidate Ranking: Stanford-UBC used Wikipedia hyperlinks (clarification, disambiguation, title) for query re-mapping, and encoded lexical and part-of-speech features from Wikipedia articles containing hyperlinks to the queries to train a supervised classifier; they reported a significant improvement on micro-averaged accuracy, from 74.85% to 82.15%. In fact, when the mined attributes become rich enough, they can be used as an expanded query and sent into an information retrieval engine in order to obtain the relevant source documents. Budapestacad team (Nemeskey et al., 2010) adopted this strategy.

3.3 Ranking Approach Comparison

The ranking approaches exploited in the KBP2010 entity linking systems can be generally categorized into four types:

(1). Unsupervised or weakly-supervised learning, in which annotated data is minimally used to tune thresholds and parameters. The similarity measure is largely based on the unlabeled contexts.

(2). Supervised learning, in which a pair of entity and KB node is modeled as an instance for classification. Such a classifier can be learned from the annotated training data based on many different features.

(3). Graph-based ranking, in which context entities are taken into account in order to reach a global optimized solution together with the query entity.

(4). IR (Information Retrieval) approach, in which the entire background source document is considered as a single query to retrieve the most relevant Wikipedia article.

The first question we will investigate is how much higher performance can be achieved by using supervised learning? Among the 16 entity linking systems which participated in the regular evaluation, LCC (Lehmann et al., 2010), HLTCOE (McNamee, 2010), Stanford-UBC (Chang et al., 2010), NUSchime (Zhang et al., 2010) and UC3M (Pablo-Sanchez et al., 2010) have explicitly used supervised classification based on many lexical and name tagging features, and most of them are ranked in top 6 in the evaluation. Therefore we can conclude that supervised learning normally leads to a reasonably good performance. However, a high-performing entity linking system can also be implemented in an unsupervised fashion by exploiting effective characteristics and algorithms, as we will discuss in the next sections.

3.4 Semantic Relation Features

Almost all entity linking systems have used semantic relations as features (e.g. BuptPris (Gao et al., 2010), CUNY (Chen et al., 2010) and HLTCOE). The semantic features used in the BuptPris system include name tagging, infoboxes, synonyms, variants and abbreviations. In the CUNY system, the semantic features are automatically extracted from their slot filling system. The results are summarized in Table 2, showing the gains over a baseline system (using only Wikipedia title features in the case of BuptPris, using tf-idf weighted word features for CUNY). As we can see, except for person entities in the BuptPris system, all types of entities have obtained significant improvement by using semantic features in entity linking.

System	Using Semantic Features	PER	ORG	GPE	Overall
BuptPris	No	83.89	59.47	33.38	58.93
	Yes	79.09	74.13	66.62	73.29
CUNY	No	84.55	63.07	57.54	59.91
	Yes	92.81	65.73	84.10	69.29

Table 2. Impact of Semantic Features on Entity Linking (Micro-Averaged Accuracy %)

3.5 Context Inference

In the current setting of KBP, a set of target entities is provided to each system in order to simplify the task and its evaluation, because it's not feasible to require a system to generate answers for all possible entities in the entire source collection. However, ideally a fully-automatic KBP system should be able to automatically discover novel entities ("queries") which have no KB entry or few slot fills in the KB, extract their attributes, and conduct global reasoning over these attributes in order to generate the final output. At the very least, due to the semantic coherence principle (McNamara, 2001), the information of an entity depends on the information of other entities. For example, the WebTLab team and the CMCRC team extracted all entities in the context of a given query, and disambiguated all entities at the same time using a PageRank-like algorithm (Page et al., 1998) or a Graph-based Re-ranking algorithm. The SMU-SIS team (Gottipati and Jiang, 2010) re-formulated queries using contexts. The LCC team modeled

contexts using Wikipedia page concepts, and computed linkability scores iteratively. Consistent improvements were reported by the WebTLab system (from 63.64% to 66.58%).

4 Entity Linking: Remaining Challenges

4.1 Comparison with Traditional Cross-document Coreference Resolution

Part of the entity linking task can be modeled as a cross-document entity resolution problem which includes two principal challenges: the same entity can be referred to by more than one name string and the same name string can refer to more than one entity. The research on cross-document entity coreference resolution can be traced back to the Web People Search task (Artiles et al., 2007) and ACE2008 (e.g. Baron and Freedman, 2008). Compared to WePS and ACE, KBP requires linking an entity mention in a source document to a knowledge base with or without Wikipedia articles. Therefore sometimes the linking decisions heavily rely on entity profile comparison with Wikipedia infoboxes. In addition, KBP introduced GPE entity disambiguation. In source documents, especially in web data, usually few explicit attributes about GPE entities are provided, so an entity linking system also needs to conduct external knowledge discovery from background related documents or hyperlink mining.

4.2 Analysis of Difficult Queries

There are 2250 queries in the Entity Linking evaluation; for 58 of them at most 5 (out of the 46) system runs produced correct answers. Most of these queries have corresponding KB entries. For 19 queries all 46 systems produced different results from the answer key. Interestingly, the systems which perform well on the difficult queries are not necessarily those achieved top overall performance – they were ranked 13rd, 6th, 5th, 12nd, 10th, and 16th respectively for overall queries. 11 queries are highly ambiguous city names which can exist in many states or countries (e.g. "Chester"), or refer to person or organization entities. From these most difficult queries we observed the following challenges and possible solutions.

- **Require deep understanding of context entities for GPE queries**

In a document where the query entity is not a central topic, the author often assumes that the readers have enough background knowledge ('anchor' location from the news release information, world knowledge or related documents) about these entities. For 6 queries, a system would need to interpret or extract attributes for their context entities. For example, in the following passage:

> *…There are also photos of **Jake** on **IHJ** in **Brentwood**, still looking somber…*

in order to identify that the query "*Brentwood*" is located in California, a system will need to understand that "*IHJ*" is "*I heart Jake community*" and that the "*Jake*" referred to lives in Los Angeles, of which *Brentwood* is a part.

In the following example, a system is required to capture the knowledge that "*Chinese Christian man*" normally appears in "*China*" or there is a "*Mission School*" in "*Canton, China*" in order to link the query "*Canton*" to the correct KB entry. This is a very difficult query also because the more common way of spelling "*Canton*" in China is "*Guangdong*".

> *…and was from a **Mission School** in **Canton**, … but for the energetic efforts of this **Chinese Christian man** and the Refuge Matron…*

- **Require external hyperlink analysis**

Some queries require a system to conduct detailed analysis on the hyperlinks in the source document or the Wikipedia document. For example, in the source document "*…Filed under: Falcons <http://sports.aol.com/fanhouse/category/atlanta-falcons/>*", a system will need to analyze the document which this hyperlink refers to. Such cases might require new query reformulation and cross-document aggregation techniques, which are both beyond traditional entity disambiguation paradigms.

- **Require Entity Salience Ranking**

Some of these queries represent salient entities and so using web popularity rank (e.g. ranking/hit counts of Wikipedia pages from search engine) can yield correct answers in most cases (Bysani et al., 2010; Dredze et al., 2010). In fact we found that a naïve candidate ranking approach based on web popularity alone can achieve 71% micro-averaged accuracy, which is better than 24 system runs in KBP2010.

Since the web information is used as a black box (including query expansion and query log analysis) which changes over time, it's more difficult to duplicate research results. However, gazetteers with entities ranked by salience or major entities marked are worth encoding as additional features. For example, in the following passages:

> *... Tritschler brothers competed in gymnastics at the 1904 Games in **St Louis** 104 years ago*" and "*A chartered airliner carrying Democratic White House hopeful Barack Obama was forced to make an unscheduled landing on Monday in **St. Louis** after its flight crew detected mechanical problems…*

although there is little background information to decide where the query "*St Louis*" is located, a system can rely on such a major city list to generate the correct linking. Similarly, if a system knows that "*Georgia Institute of Technology*" has higher salience than "*Georgian Technical University*", it can correctly link a query "*Georgia Tech*" in most cases.

5 Slot Filling: What Works

5.1 A General Architecture

The slot-filling task is a hybrid of traditional IE (a fixed set of relations) and QA (responding to a query, generating a unified response from a large collection). Most participants met this challenge through a hybrid system which combined aspects of QA (passage retrieval) and IE (answer extraction). A few used off-the-shelf QA, either bypassing question analysis or (if QA was used as a "black box") creating a set of questions corresponding to each slot.

The basic system structure (Figure 2) involved three phases: document/passage retrieval (retrieving passages involving the queried entity), answer extraction (getting specific answers from the retrieved passages), and answer combination (merging and selecting among the answers extracted).

The solutions adopted for answer extraction reflected the range of current IE methods as well as QA answer extraction techniques (see Table 3). Most systems used one main pipeline, while CUNY and BuptPris adopted a hybrid approach of combining multiple approaches.

One particular challenge for KBP, in comparison with earlier IE tasks, was the paucity of training data. The official training data, linked to specific text from specific documents, consisted of responses to 100 queries; the participants jointly prepared responses to another 50. So traditional supervised learning, based directly on the training data, would provide limited coverage. Coverage could be improved by using the training data as seeds for a bootstrapping procedure.

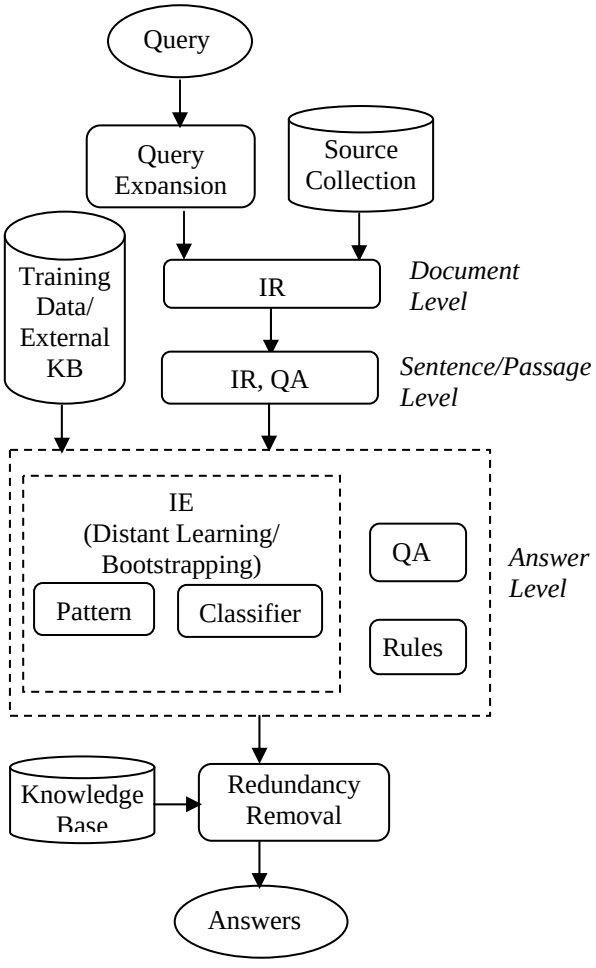


Figure 2. General Slot Filling System Architecture

Methods		System Examples	
Trained IE	Pattern Learning	Distant Learning (large seed, one iteration)	CUNY (Chen et al., 2010)
		Bootstrapping (small seed, multiple iterations)	NYU (Grishman and Min, 2010)
	Supervised Classifier	Distant Supervision	Budapestacad (Nemeskey et al., 2010), lsv (Chrupala et al., 2010), Stanford (Surdeanu et al., 2010), UBC (Intxaurrondo et al., 2010)
		Trained from KBP training data and other related tasks	BuptPris (Gao et al., 2010), CUNY (Chen et al., 2010), IBM (Castelli et al., 2010), ICL (Song et al., 2010), LCC (Lehmann et al., 2010), lsv (Chrupala et al., 2010), Siel (Bysani et al., 2010)
QA			CUNY (Chen et al., 2010), iirg (Byrne and Dunnion, 2010)
Hand-coded Heuristic Rules			BuptPris (Gao et al., 2010), USFD (Yu et al., 2010)

Table 3. Slot Filling Answer Extraction Method Comparison

On the other hand, there were a lot of 'facts' available – pairs of entities bearing a relationship corresponding closely to the KBP relations – in the form of filled Wikipedia infoboxes. These could be used for various forms of indirect or distant learning, where instances in a large corpus of such pairs are taken as (positive) training instances. However, such instances are noisy – if a pair of entities participates in more than one relation, the found instance may not be an example of the intended relation – and so some filtering of the instances or resulting patterns may be needed. Several sites used such distant supervision to acquire patterns or train classifiers, in some cases combined with direct supervision using the training data (Chrupala et al., 2010).

Several groups used and extended existing relation extraction systems, and then mapped the results into KBP slots. Mapping the ACE relations and events by themselves provided limited coverage (34% of slot fills in the training data), but was helpful when combined with other sources (e.g. CUNY). Groups with more extensive existing extraction systems could primarily build on these (e.g. LCC, IBM).

For example, IBM (Castelli et al., 2010) extended their mention detection component to cover 36 entity types which include many non-ACE types; and added new relation types between entities and event anchors. LCC and CUNY applied active learning techniques to cover non-ACE types of entities, such as “origin”, “religion”, “title”, “charge”, “web-site” and “cause-of-death”, and effectively develop lexicons to filter spurious answers.

Top systems also benefited from customizing and tightly integrating their recently enhanced extraction techniques into KBP. For example, IBM, NYU (Grishman and Min, 2010) and CUNY exploited entity coreference in pattern learning and reasoning. It is also notable that traditional extraction components trained from newswire data suffer from noise in web data. In order to address this problem, IBM applied their new robust mention detection techniques for noisy inputs (Florian et al., 2010); CUNY developed a component to recover structured forms such as tables in web data automatically and filter spurious answers.

5.2 Use of External Knowledge Base

Many instance-centered knowledge bases that have harvested Wikipedia are proliferating on the semantic web. The most well known are probably the Wikipedia derived resources, including DBpedia (Auer 2007), Freebase (Bollacker 2008) and YAGO (Suchanek et al., 2007) and Linked Open Data (http://data.nytimes.com/). The main motivation of the KBP program is to automatically distill information from news and web unstructured data instead of manually constructed knowledge bases, but these existing knowledge bases can provide a large number of seed tuples to bootstrap slot filling or guide distant learning.

Such resources can also be used in a more direct way. For example, CUNY exploited Freebase and LCC exploited DBpedia as fact validation in slot filling. However, most of these resources are manually created from single data modalities and only cover well-known entities. For example, while Freebase contains 116 million instances of

7,300 relations for 9 million entities, it only covers 48% of the slot types and 5% of the slot answers in KBP2010 evaluation data. Therefore, both CUNY and LCC observed limited gains from the answer validation approach from Freebase. Both systems gained about 1% improvement in recall with a slight loss in precision.

5.3 Cross-Slot and Cross-Query Reasoning

Slot Filling can also benefit from extracting revertible queries from the context of any target query, and conducting global ranking or reasoning to refine the results. CUNY and IBM developed recursive reasoning components to refine extraction results. For a given query, if there are no other related answer candidates available, they built "revertible" queries in the contexts, similar to (Prager et al., 2006), to enrich the inference process iteratively. For example, if *a* is extracted as the answer for org:subsidiaries of the query *q*, we can consider *a* as a new revertible query and verify that a org:parents answer of *a* is *q*. Both systems significantly benefited from recursive reasoning (CUNY F-measure on training data was enhanced from 33.57% to 35.29% and IBM F-measure was enhanced from 26% to 34.83%).

6 Slot Filling: Remaining Challenges

Slot filling remains a very challenging task; only one system exceeded 30% F-measure on the 2010 evaluation. During the 2010 evaluation data annotation/adjudication process, an initial answer key annotation was created by a manual search of the corpus (resulting in 797 instances), and then an independent adjudication pass was applied to assess these annotations together with pooled system responses. The Precision, Recall and F-measure for the initial human annotation are only about 70%, 54% and 61% respectively. While we believe the annotation consistency can be improved, in part by refinement of the annotation guidelines, this does place a limit on system performance.

Most of the shortfall in system performance reflects inadequacies in the answer extraction stage, reflecting limitations in the current state-of-the-art in information extraction. An analysis of the 2010 training data shows that cross-sentence coreference and some types of inference are critical to slot filling. In only 60.4% of the cases do the entity name and slot fill appear together in the same sentence, so a system which processes sentences in isolation is severely limited in its performance. 22.8% of the cases require cross-sentence (identity) coreference; 15% require some cross-sentence inference and 1.8% require cross-slot inference. The inferences include:

- Non-identity coreference: in the following passage: "***Lahoud*** is married to an Armenian and the couple have *three children*. Eldest *son* ***Emile Emile Lahoud*** was a member of parliament between 2000 and 2005." the semantic relation between "*children*" and "*son*" needs to be exploited in order to generate "*Emile Emile Lahoud*" as the *per:children* of the query entity "*Lahoud*";

- Cross-slot inference based on revertible queries, propagation links or even world knowledge to capture some of the most challenging cases. In the KBP slot filling task, slots are often dependent on each other, so we can improve the results by improving the "coherence" of the story (i.e. consistency among all generated answers (query profiles)). In the following example:

  "People Magazine has confirmed that actress Julia Roberts has given birth to her third child a boy named ***Henry Daniel Moder****. Henry was born Monday in Los Angeles and weighed 8? lbs. Roberts, 39, and husband* ***Danny Moder****, 38, are already parents to twins Hazel and Phinnaeus who were born in November 2006."*

  the following reasoning rules are needed to generate the answer "Henry Daniel Moder" as *per:children* of "Danny Moder":

  ChildOf ("Henry Daniel Moder", "Julia Roberts*")*
  ∧ *Coreferential (*"Julia Roberts", "Roberts*")*
  ∧ *SpouseOf (*"Roberts", "Danny Moder*")* → *ChildOf (*"Henry Daniel Moder", "Danny Moder*")*

KBP Slot Filling is similar to ACE Relation Extraction, which has been extensively studied for the past 7 years. However, the amount of training data is much smaller, forcing sites to adjust their training strategies. Also, some of the constraints of ACE relation mention extraction – notably, that both arguments are present in the same sentence – are not present, making the role of coreference and cross-sentence inference more critical.

The role of coreference and inference as limiting factors, while generally recognized, is emphasized

by examining the 163 slot values that the human annotators filled but that none of the systems were able to get correct. Many of these difficult cases involve a combination of problems, but we estimate that at least 25% of the examples involve coreference which is beyond current system capabilities, such as nominal anaphors:

"Alexandra Burke is out with the video for her second single … taken from the British artist's debut album"
"a woman charged with running a prostitution ring … her business, Pamela Martin and Associates"
(underlined phrases are coreferential).

While the types of inferences which may be required is open-ended, certain types come up repeatedly, reflecting the types of slots to be filled: systems would benefit from specialists which are able to reason about times, locations, family relationships, and employment relationships.

7 Toward System Combination

The increasing number of diverse approaches based on different resources provide new opportunities for both entity linking and slot filling tasks to benefit from system combination.

The NUSchime entity linking system trained a SVM based re-scoring model to combine two individual pipelines. Only one feature based on confidence values from the pipelines was used for re-scoring. The micro-averaged accuracy was enhanced from 79.29%/79.07% to 79.38% after combination. We also applied a voting approach on the top 9 entity linking systems and found that all combination orders achieved significant gains, with the highest absolute improvement of 4.7% in micro-averaged accuracy over the top entity linking system.

The CUNY slot filling system trained a maximum-entropy-based re-ranking model to combine three individual pipelines, based on various global features including voting and dependency relations. Significant gain in F-measure was achieved: from 17.9%, 27.7% and 21.0% (on training data) to 34.3% after combination. When we applied the same re-ranking approach to the slot filling systems which were ranked from the 2nd to 14th, we achieved 4.3% higher F-score than the best of these systems.

8 Conclusion

Compared to traditional IE and QA tasks, KBP has raised some interesting and important research issues: It places more emphasis on cross-document entity resolution which received limited effort in ACE; it forces systems to deal with redundant and conflicting answers across large corpora; it links the facts in text to a knowledge base so that NLP and data mining/database communities have a better chance to collaborate; it provides opportunities to develop novel training methods such as distant (and noisy) supervision through Infoboxes (Surdeanu et al., 2010; Chen et al., 2010).

In this paper, we provided detailed analysis of the reasons which have made KBP a more challenging task, shared our observations and lessons learned from the evaluation, and suggested some possible research directions to address these challenges which may be helpful for current and new participants, or IE and QA researchers in general.

Acknowledgements

The first author was supported by the U.S. Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053, the U.S. NSF CAREER Award under Grant IIS-0953149 and PSC-CUNY Research Program. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

Javier Artiles, Julio Gonzalo and Satoshi Sekine. 2007. The SemEval-2007 WePS Evaluation: Establishing a benchmark for the Web People Search Task. *Proc. the 4th International Workshop on Semantic Evaluations (Semeval-2007).*

S. Auer, C. Bizer, G. Kobilarov, J. Lehmann and Z. Ives. 2007. DBpedia: A nucleus for a web of open data. *Proc. 6th International Semantic Web Conference.*

K. Balog, L. Azzopardi, M. de Rijke. 2008. Personal Name Resolution of Web People Search. *Proc. WWW2008 Workshop: NLP Challenges in the Information Explosion Era (NLPIX 2008).*

Alex Baron and Marjorie Freedman. 2008. Who is Who and What is What: Experiments in Cross-Document Co-Reference. *Proc. EMNLP 2008*.

K. Bollacker, R. Cook, and P. Tufts. 2007. Freebase: A Shared Database of Structured General Human Knowledge. *Proc. National Conference on Artificial Intelligence* (Volume 2).

Lorna Byrne and John Dunnion. 2010. UCD IIRG at TAC 2010. *Proc. TAC 2010 Workshop.*

Praveen Bysani, Kranthi Reddy, Vijay Bharath Reddy, Sudheer Kovelamudi, Prasad Pingali and Vasudeva Varma. 2010. IIIT Hyderabad in Guided Summarization and Knowledge Base Population. *Proc. TAC 2010 Workshop*.

Vittorio Castelli, Radu Florian and Ding-jung Han. 2010. Slot Filling through Statistical Processing and Inference Rules. *Proc. TAC 2010 Workshop*.

Angel X. Chang, Valentin I. Spitkovsky, Eric Yeh, Eneko Agirre and Christopher D. Manning. 2010. Stanford-UBC Entity Linking at TAC-KBP. *Proc. TAC 2010 Workshop*.

Zheng Chen, Suzanne Tamang, Adam Lee, Xiang Li, Wen-Pin Lin, Matthew Snover, Javier Artiles, Marissa Passantino and Heng Ji. 2010. CUNY-BLENDER TAC-KBP2010 Entity Linking and Slot Filling System Description. *Proc. TAC 2010 Workshop*.

Grzegorz Chrupala, Saeedeh Momtazi, Michael Wiegand, Stefan Kazalski, Fang Xu, Benjamin Roth, Alexandra Balahur, Dietrick Klakow. Saarland University Spoken Language Systems at the Slot Filling Task of TAC KBP 2010. *Proc. TAC 2010 Workshop*.

Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber and Tim Finin. 2010. Entity Disambiguation for Knowledge Base Population. *Proc. COLING 2010*.

Norberto Fernandez, Jesus A. Fisteus, Luis Sanchez and Eduardo Martin. 2010. WebTLab: A Cooccurence-based Approach to KBP 2010 Entity-Linking Task. *Proc. TAC 2010 Workshop*.

Radu Florian, John F. Pitrelli, Salim Roukos and Imed Zitouni. 2010. Improving Mention Detection Robustness to Noisy Input. *Proc. EMNLP2010*.

Sanyuan Gao, Yichao Cai, Si Li, Zongyu Zhang, Jingyi Guan, Yan Li, Hao Zhang, Weiran Xu and Jun Guo. 2010. PRIS at TAC2010 KBP Track. *Proc. TAC 2010 Workshop*.

Swapna Gottipati and Jing Jiang. 2010. SMU-SIS at TAC 2010 – KBP Track Entity Linking. *Proc. TAC 2010 Workshop*.

Ralph Grishman and Bonan Min. 2010. New York University KBP 2010 Slot-Filling System. *Proc. TAC 2010 Workshop*.

Ander Intxaurrondo, Oier Lopez de Lacalle and Eneko Agirre. 2010. UBC at Slot Filling TAC-KBP2010. *Proc. TAC 2010 Workshop*.

John Lehmann, Sean Monahan, Luke Nezda, Arnold Jung and Ying Shi. 2010. LCC Approaches to Knowledge Base Population at TAC 2010. *Proc. TAC 2010 Workshop*.

Paul McNamee and Hoa Dang. 2009. Overview of the TAC 2009 Knowledge Base Population Track. *Proc. TAC 2009 Workshop*.

Paul McNamee, Hoa Trang Dang, Heather Simpson, Patrick Schone and Stephanie M. Strassel. 2010. An Evaluation of Technologies for Knowledge Base Population. *Proc. LREC2010.*

Paul McNamee, Rion Snow, Patrick Schone and James Mayfield. 2008. Learning Named Entity Hyponyms for Question Answering. *Proc. IJCNLP2008.*

Paul McNamee. 2010. HLTCOE Efforts in Entity Linking at TAC KBP 2010. *Proc. TAC 2010 Workshop.*

Danielle S McNamara. 2001. Reading both High-coherence and Low-coherence Texts: Effects of Text Sequence and Prior Knowledge. *Canadian Journal of Experimental Psychology.*

Olena Medelyan, Catherine Legg, David Milne and Ian H. Witten. 2009. Mining Meaning from Wikipedia. *International Journal of Human-Computer Studies archive.* Volume 67 , Issue 9.

David Nemeskey, Gabor Recski, Attila Zseder and Andras Kornai. 2010. BUDAPESTACAD at TAC 2010. *Proc. TAC 2010 Workshop*.

Cesar de Pablo-Sanchez, Juan Perea and Paloma Martinez. 2010. Combining Similarities with Regression based Classifiers for Entity Linking at TAC 2010. *Proc. TAC 2010 Workshop*.

Lawrence Page, Sergey Brin, Rajeev Motwani and Terry Winograd. 1998. The PageRank Citation Ranking: Bringing Order to the Web. *Proc. the 7th International World Wide Web Conference.*

Luiz Augusto Pizzato, Diego Molla and Cecile Paris. 2006. Pseudo Relevance Feedback Using Named Entities for Question Answering. *Proc. the Australasian Language Technology Workshop 2006.*

J. Prager, P. Duboue, and J. Chu-Carroll. 2006. Improving QA Accuracy by Question Inversion. *Proc. ACL-COLING 2006.*

Will Radford, Ben Hachey, Joel Nothman, Matthew Honnibal and James R. Curran. 2010. CMCRC at TAC10: Document-level Entity Linking with Graph-based Re-ranking. *Proc. TAC 2010 Workshop*.

Yang Song, Zhengyan He and Houfeng Wang. 2010. ICL_KBP Approaches to Knowledge Base Population at TAC2010. *Proc. TAC 2010 Workshop*.

F. M. Suchanek, G. Kasneci, and G. Weikum. 2007. Yago: A Core of Semantic Knowledge. *Proc. 16th International World Wide Web Conference.*

Mihai Surdeanu, David McClosky, Julie Tibshirani, John Bauer, Angel X. Chang, Valentin I. Spitkovsky, Christopher D. Manning. 2010. A Simple Distant Supervision Approach for the TAC-KBP Slot Filling Task. *Proc. TAC 2010 Workshop*.

Ani Thomas, Arpana Rawai, M K Kowar, Sanjay Sharma, Sarang Pitale and Neeraj Kharya. 2010. Bhilai Institute of Technology Durg at TAC 2010: Knowledge Base Population Task Challenge. *Proc. TAC 2010 Workshop*.

Jingtao Yu, Omkar Mujgond and Rob Gaizauskas. 2010. The University of Sheffield System at TAC KBP 2010. *Proc. TAC 2010 Workshop*.

Wei Zhang, Yan Chuan Sim, Jian Su and Chew Lim Tan. 2010. NUS-I2R: Learning a Combined System for Entity Linking. *Proc. TAC 2010 Workshop*.

Nonlinear Evidence Fusion and Propagation for Hyponymy Relation Mining

Fan Zhang[2*] **Shuming Shi**[1] **Jing Liu**[2] **Shuqi Sun**[3*] **Chin-Yew Lin**[1]
[1]Microsoft Research Asia
[2]Nankai University, China
[3]Harbin Institute of Technology, China
{shumings, cyl}@microsoft.com

Abstract

This paper focuses on mining the hyponymy (or is-a) relation from large-scale, open-domain web documents. A nonlinear probabilistic model is exploited to model the correlation between sentences in the aggregation of pattern matching results. Based on the model, we design a set of evidence combination and propagation algorithms. These significantly improve the result quality of existing approaches. Experimental results conducted on 500 million web pages and hypernym labels for 300 terms show over 20% performance improvement in terms of P@5, MAP and R-Precision.

1 Introduction

An important task in text mining is the automatic extraction of entities and their lexical relations; this has wide applications in natural language processing and web search. This paper focuses on mining the hyponymy (or is-a) relation from large-scale, open-domain web documents. From the viewpoint of entity classification, the problem is to automatically assign *fine-grained* class labels to terms.

There have been a number of approaches (Hearst 1992; Pantel & Ravichandran 2004; Snow et al., 2005; Durme & Pasca, 2008; Talukdar et al., 2008) to address the problem. These methods typically exploited manually-designed or automatically-learned patterns (e.g., "*NP such as NP*", "*NP like NP*", "*NP is a NP*"). Although some degree of success has been achieved with these efforts, the results are still far from perfect, in terms of both recall and precision. As will be demonstrated in this paper, even by processing a large corpus of 500 million web pages with the most popular patterns, we are not able to extract correct labels for many (especially rare) entities. Even for popular terms, incorrect results often appear in their label lists.

The basic philosophy in existing hyponymy extraction approaches (and also many other text-mining methods) is *counting*: count the number of supporting sentences. Here a *supporting sentence* of a term-label pair is a sentence from which the pair can be extracted via an extraction pattern. We demonstrate that the specific way of counting has a great impact on result quality, and that the state-of-the-art counting methods are not optimal. Specifically, we examine the problem from the viewpoint of probabilistic evidence combination and find that the probabilistic assumption behind simple counting is the statistical *independence* between the observations of supporting sentences. By assuming a *positive correlation* between supporting sentence observations and adopting properly designed nonlinear combination functions, the results precision can be improved.

It is hard to extract correct labels for rare terms from a web corpus due to the data sparseness problem. To address this issue, we propose an evidence propagation algorithm motivated by the observation that similar terms tend to share common hypernyms. For example, if we already know that 1) Helsinki and Tampere are cities, and 2) Porvoo is similar to Helsinki and Tampere, then Porvoo is

[*] This work was performed when Fan Zhang and Shuqi Sun were interns at Microsoft Research Asia

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1159–1168,
Portland, Oregon, June 19-24, 2011.

very likely also a city. This intuition, however, does not mean that the labels of a term can always be transferred to its similar terms. For example, Mount Vesuvius and Kilimanjaro are volcanoes and Lhotse is similar to them, but Lhotse is not a volcano. Therefore we should be very conservative and careful in hypernym propagation. In our propagation algorithm, we first construct some *pseudo supporting sentences* for a term from the supporting sentences of its similar terms. Then we calculate label scores for terms by performing *nonlinear* evidence combination based on the (pseudo and real) supporting sentences. Such a nonlinear propagation algorithm is demonstrated to perform better than linear propagation.

Experimental results on a publicly available collection of 500 million web pages with hypernym labels annotated for 300 terms show that our nonlinear evidence fusion and propagation significantly improve the precision and coverage of the extracted hyponymy data. This is one of the technologies adopted in our semantic search and mining system *NeedleSeek*[2].

In the next section, we discuss major related efforts and how they differ from our work. Section 3 is a brief description of the baseline approach. The probabilistic evidence combination model that we exploited is introduced in Section 4. Our main approach is illustrated in Section 5. Section 6 shows our experimental settings and results. Finally, Section 7 concludes this paper.

2 Related Work

Existing efforts for hyponymy relation extraction have been conducted upon various types of data sources, including plain-text corpora (Hearst 1992; Pantel & Ravichandran, 2004; Snow et al., 2005; Snow et al., 2006; Banko, et al., 2007; Durme & Pasca, 2008; Talukdar et al., 2008), semi-structured web pages (Cafarella et al., 2008; Shinzato & Torisawa, 2004), web search results (Geraci et al., 2006; Kozareva et al., 2008; Wang & Cohen, 2009), and query logs (Pasca 2010). Our target for optimization in this paper is the approaches that use lexico-syntactic patterns to extract hyponymy relations from plain-text corpora. Our future work will study the application of the proposed algorithms on other types of approaches.

[2] http://research.microsoft.com/en-us/projects/needleseek/ or http://needleseek.msra.cn/

The probabilistic evidence combination model that we exploit here was first proposed in (Shi et al., 2009), for combining the page in-link evidence in building a nonlinear static-rank computation algorithm. We applied it to the hyponymy extraction problem because the model takes the dependency between supporting sentences into consideration and the resultant evidence fusion formulas are quite simple. In (Snow et al., 2006), a probabilistic model was adopted to combine evidence from heterogeneous relationships to jointly optimize the relationships. The independence of evidence was assumed in their model. In comparison, we show that better results will be obtained if the evidence correlation is modeled appropriately.

Our evidence propagation is basically about using term similarity information to help instance labeling. There have been several approaches which improve hyponymy extraction with instance clusters built by distributional similarity. In (Pantel & Ravichandran, 2004), labels were assigned to the committee (i.e., representative members) of a semantic class and used as the hypernyms of the whole class. Labels generated by their approach tend to be rather coarse-grained, excluding the possibility of a term having its private labels (considering the case that one meaning of a term is not covered by the input semantic classes). In contrast to their method, our label scoring and ranking approach is applied to every single term rather than a semantic class. In addition, we also compute label scores in a nonlinear way, which improves results quality. In Snow et al. (2005), a supervised approach was proposed to improve hypernym classification using coordinate terms. In comparison, our approach is unsupervised. Durme & Pasca (2008) cleaned the set of instance-label pairs with a TF*IDF like method, by exploiting clusters of semantically related phrases. The core idea is to keep a term-label pair (T, L) only if the number of terms having the label L in the term T's cluster is above a threshold and if L is not the label of too many clusters (otherwise the pair will be discarded). In contrast, we are able to add new (high-quality) labels for a term with our evidence propagation method. On the other hand, low quality labels get smaller score gains via propagation and are ranked lower.

Label propagation is performed in (Talukdar et al., 2008; Talukdar & Pereira, 2010) based on multiple instance-label graphs. Term similarity information was not used in their approach.

Most existing work tends to utilize small-scale or private corpora, whereas the corpus that we used is publicly available and much larger than most of the existing work. We published our term sets (refer to Section 6.1) and their corresponding user judgments so researchers working on similar topics can reproduce our results.

Type	Pattern
Hearst-I	NP_L {,} (such as) {NP,}* {and|or} NP
Hearst-II	NP_L {,} (include(s) | including) {NP,}* {and|or} NP
Hearst-III	NP_L {,} (e.g.|e.g) {NP,}* {and|or} NP
IsA-I	NP (is|are|was|were|being) (a|an) NP_L
IsA-II	NP (is|are|was|were|being) {the, those} NP_L
IsA-III	NP (is|are|was|were|being) {another, any} NP_L

Table 1. Patterns adopted in this paper (NP: named phrase representing an entity; NP_L: label)

3 Preliminaries

The problem addressed in this paper is corpus-based *is-a* relation mining: extracting hypernyms (as labels) for entities from a large-scale, open-domain document corpus. The desired output is a mapping from terms to their corresponding hypernyms, which can naturally be represented as a weighted bipartite graph (term-label graph). Typically we are only interested in top labels of a term in the graph.

Following existing efforts, we adopt pattern-matching as a basic way of extracting hypernymy/hyponymy relations. Two types of patterns (refer to Table 1) are employed, including the popular "Hearst patterns" (Hearst, 1992) and the IsA patterns which are exploited less frequently in existing hyponym mining efforts. One or more term-label pairs can be extracted if a pattern matches a sentence. In the baseline approach, the weight of an edge $T \rightarrow L$ (from term T to hypernym label L) in the term-label graph is computed as,

$$w(T \rightarrow L) = m \cdot \mathrm{IDF}(L) = m \cdot \log \frac{1+N}{1+\mathrm{DF}(L)} \qquad (3.1)$$

where m is the number of times the pair (T, L) is extracted from the corpus, $\mathrm{DF}(L)$ is the number of in-links of L in the graph, N is total number of terms in the graph, and IDF means the "inverse document frequency".

A term can only keep its top-k neighbors (according to the edge weight) in the graph as its final labels.

Our pattern matching algorithm implemented in this paper uses part-of-speech (POS) tagging information, without adopting a parser or a chunker. The noun phrase boundaries (for terms and labels) are determined by a manually designed POS tag list.

4 Probabilistic Label-Scoring Model

Here we model the hyponymy extraction problem from the probability theory point of view, aiming at estimating the score of a term-label pair (i.e., the score of a label w.r.t. a term) with probabilistic evidence combination. The model was studied in (Shi et al., 2009) to combine the page in-link evidence in building a nonlinear static-rank computation algorithm.

We represent the score of a term-label pair by the probability of the label being a correct hypernym of the term, and define the following events,

$A_{T,L}$: Label L is a hypernym of term T (the abbreviated form A is used in this paper unless it is ambiguous).

E_i: The observation that (T, L) is extracted from a sentence S_i via pattern matching (i.e., S_i is a supporting sentence of the pair).

Assuming that we already know m supporting sentences ($S_1 \sim S_m$), our problem is to compute $P(A|E_1,E_2,..,E_m)$, the *posterior* probability that L is a hypernym of term T, given evidence $E_1 \sim E_m$. Formally, we need to find a function f to satisfy,

$$P(A|E_1,\ldots,E_m) = f(P(A), P(A|E_1)\ldots, P(A|E_m)) \qquad (4.1)$$

For simplicity, we first consider the case of m=2. The case of m>2 is quite similar.

We start from the simple case of independent supporting sentences. That is,

$$P(E_1, E_2) = P(E_1) \cdot P(E_2) \qquad (4.2)$$

$$P(E_1, E_2|A) = P(E_1|A) \cdot P(E_2|A) \qquad (4.3)$$

By applying Bayes rule, we get,

$$\begin{aligned} P(A|E_1, E_2) &= \frac{P(E_1, E_2|A) \cdot P(A)}{P(E_1, E_2)} \\ &= \frac{P(E_1|A) \cdot P(A)}{P(E_1)} \cdot \frac{P(E_2|A) \cdot P(A)}{P(E_2)} \cdot \frac{1}{P(A)} \\ &= \frac{P(A|E_1) \cdot P(A|E_2)}{P(A)} \end{aligned} \qquad (4.4)$$

Then define

$$G(A|E) = \log \frac{P(A|E)}{P(A)} = \log(P(A|E)) - log(\mathrm{P}(A))$$

Here $G(A|E)$ represents the *log-probability-gain* of A given E, with the meaning of the *gain* in the log-probability value of A after the evidence E is observed (or known). It is a measure of the impact of evidence E to the probability of event A. With the definition of $G(A|E)$, Formula 4.4 can be transformed to,

$$G(A|E_1, E_2) = G(A|E_1) + G(A|E_2) \quad (4.5)$$

Therefore, if E_1 and E_2 are independent, the log-probability-gain of A given both pieces of evidence will exactly be the sum of the gains of A given every single piece of evidence respectively. It is easy to prove (by following a similar procedure) that the above Formula holds for the case of m>2, as long as the pieces of evidence are mutually independent.

Therefore for a term-label pair with m mutually independent supporting sentences, if we set every gain $G(A|E_i)$ to be a constant value g, the posterior gain score of the pair will be $\sum_{i=1}^{m} g = mg$. If the value g is the IDF of label L, the posterior gain will be,

$$G(A_{T,L}|E_1 \ldots, E_m) = \sum_{i=1}^{m} \mathrm{IDF}(L) = m \cdot \mathrm{IDF}(L) \quad (4.6)$$

This is exactly the Formula 3.1. By this way, we provide a probabilistic explanation of scoring the candidate labels for a term via simple counting.

	Hearst-I	**IsA-I**	**E_1: Hearst-I E_2: IsA-I**
R_A: $\frac{P(E_1,E_2\|A)}{P(E_1\|A)P(E_2\|A)}$	66.87	17.30	24.38
R: $\frac{P(E_1,E_2)}{P(E_1)P(E_2)}$	5997	1711	802.7
R_A/R	0.011	0.010	0.030

Table 2. Evidence dependency estimation for intra-pattern and inter-pattern supporting sentences

In the above analysis, we assume the statistical independence of the supporting sentence observations, which may not hold in reality. Intuitively, if we already know one supporting sentence S_1 for a term-label pair (T, L), then we have more chance to find another supporting sentence than if we do not know S_1. The reason is that, before we find S_1, we have to estimate the probability with the chance of discovering a supporting sentence for a *random* term-label pair. The probability is quite low because most term-label pairs do not have hyponymy relations. Once we have observed S_1, however, the chance of (T, L) having a hyponymy relation increases. Therefore the chance of observing another supporting sentence becomes larger than before.

Table 2 shows the rough estimation of $\frac{P(E_1,E_2|A)}{P(E_1|A)P(E_2|A)}$ (denoted as R_A), $\frac{P(E_1,E_2)}{P(E_1)P(E_2)}$ (denoted as R), and their ratios. The statistics are obtained by performing maximal likelihood estimation (MLE) upon our corpus and a random selection of term-label pairs from our term sets (see Section 6.1) together with their top labels[3]. The data verifies our analysis about the correlation between E_1 and E_2 (note that R=1 means independent). In addition, it can be seen that the conditional independence assumption of Formula 4.3 does not hold (because R_A>1). It is hence necessary to consider the correlation between supporting sentences in the model. The estimation of Table 2 also indicates that,

$$\frac{P(E_1, E_2)}{P(E_1)P(E_2)} > \frac{P(E_1, E_2|A)}{P(E_1|A)P(E_2|A)} \quad (4.7)$$

By following a similar procedure as above, with Formulas 4.2 and 4.3 replaced by 4.7, we have,

$$G(A|E_1, E_2) < G(A|E_1) + G(A|E_2) \quad (4.8)$$

This formula indicates that when the supporting sentences are positively correlated, the posterior score of label L w.r.t. term T (given both the sentences) is smaller than the sum of the gains caused by one sentence only. In the extreme case that sentence S_2 fully depends on E_1 (i.e. $P(E_2|E_1)$=1), it is easy to prove that

$$G(A|E_1, E_2) = G(A|E_1)$$

It is reasonable, since event E_2 does not bring in more information than E_1.

Formula 4.8 cannot be used directly for computing the posterior gain. What we really need is a function h satisfying

$$G(A|E_1, \ldots, E_m) = h\big(G(A|E_1), \ldots, G(A|E_m)\big) \quad (4.9)$$

and

$$h(x_1 \ldots, x_m) < \sum_{i=1}^{m} x_i \quad (4.10)$$

Shi et al. (2009) discussed other constraints to h and suggested the following nonlinear functions,

$$h_1(x_1 \ldots, x_m) = \ln(1 + \sum_{i=1}^{m} (\mathrm{e}^{x_i} - 1)) \quad (4.11)$$

[3] R_A is estimated from the labels judged as "Good"; whereas the estimation of R is from all judged labels.

$$h_2(x_1 \ldots, x_m) = \sqrt[p]{\sum_{i=1}^{m} x_i^p} \qquad (p>1) \qquad (4.12)$$

In the next section, we use the above two *h* functions as basic building blocks to compute label scores for terms.

5 Our Approach

Multiple types of patterns (Table 1) can be adopted to extract term-label pairs. For two supporting sentences the correlation between them may depend on whether they correspond to the same pattern. In Section 5.1, our nonlinear evidence fusion formulas are constructed by making specific assumptions about the correlation between intra-pattern supporting sentences and inter-pattern ones.

Then in Section 5.2, we introduce our evidence propagation technique in which the evidence of a (T, L) pair is propagated to the terms similar to *T*.

5.1 Nonlinear evidence fusion

For a term-label pair (*T*, *L*), assuming *K* patterns are used for hyponymy extraction and the supporting sentences discovered with pattern *i* are,

$$S_{i,1}, S_{i,2} \ldots, S_{i,m_i} \qquad (5.1)$$

where m_i is the number of supporting sentences corresponding to pattern *i*. Also assume the gain score of $S_{i,j}$ is $x_{i,j}$, i.e., $x_{i,j}=G(A|S_{i,j})$.

Generally speaking, supporting sentences corresponding to the same pattern typically have a higher correlation than the sentences corresponding to different patterns. This can be verified by the data in Table-2. By ignoring the inter-pattern correlations, we make the following simplified assumption:

Assumption: Supporting sentences corresponding to the same pattern are correlated, while those of different patterns are independent.

According to this assumption, our label-scoring function is,

$$score(T, L) = \sum_{i=1}^{K} h(x_{i,1}, x_{i,2} \ldots, x_{i,m_i}) \qquad (5.2)$$

In the simple case that $x_{i,j} = \mathrm{IDF}(L)$, if the *h* function of Formula 4.12 is adopted, then,

$$Score(T, L) = \left(\sum_{i=1}^{K} \sqrt[p]{m_i} \right) \cdot \mathrm{IDF}(L) \qquad (5.3)$$

We use an example to illustrate the above formula.

Example: For term *T* and label L_1, assume the numbers of the supporting sentences corresponding to the six pattern types in Table 1 are (4, 4, 4, 4, 4, 4), which means the number of supporting sentences discovered by each pattern type is 4. Also assume the supporting-sentence-count vector of label L_2 is (25, 0, 0, 0, 0, 0). If we use Formula 5.3 to compute the scores of L_1 and L_2, we can have the following (ignoring IDF for simplicity),

$Score(L_1) = 6 \cdot \sqrt{4} = 12$; $Score(L_2) = \sqrt{25} = 5$

One the other hand, if we simply count the total number of supporting sentences, the score of L_2 will be larger.

The rationale implied in the formula is: For a given term *T*, the labels supported by multiple types of patterns tend to be more reliable than those supported by a single pattern type, if they have the same number of supporting sentences.

5.2 Evidence propagation

According to the evidence fusion algorithm described above, in order to extract term labels reliably, it is desirable to have many supporting sentences of different types. This is a big challenge for rare terms, due to their low frequency in sentences (and even lower frequency in supporting sentences because not all occurrences can be covered by patterns). With evidence propagation, we aim at discovering more supporting sentences for terms (especially rare terms). Evidence propagation is motivated by the following two observations:

(I) Similar entities or coordinate terms tend to share some common hypernyms.

(II) Large term similarity graphs are able to be built efficiently with state-of-the-art techniques (Agirre et al., 2009; Pantel et al., 2009; Shi et al., 2010). With the graphs, we can obtain the similarity between two terms without their hypernyms being available.

The first observation motivates us to "*borrow*" the supporting sentences from other terms as auxiliary evidence of the term. The second observation means that *new* information is brought with the state-of-the-art term similarity graphs (in addition to the term-label information discovered with the patterns of Table 1).

Our evidence propagation algorithm contains two phases. In phase I, some *pseudo supporting sentences* are constructed for a term from the supporting sentences of its neighbors in the similarity graph. Then we calculate the label scores for terms based on their (pseudo and real) supporting sentences.

Phase I: For every supporting sentence S and every similar term T_1 of the term T, add a pseudo supporting sentence S_1 for T_1, with the gain score,

$$G(A_{T_1,L_1}|S_1) = \mu \cdot \mathrm{Sim}(T,T_1) \cdot G(A_{T,L}|S) \qquad (5.5)$$

where $\mu \in [0,1]$ is the propagation factor, and $\mathrm{Sim}(\cdot,\cdot)$ is the term similarity function taking values in [0, 1]. The formula reasonably assumes that the gain score of the pseudo supporting sentence depends on the gain score of the original real supporting sentence, the similarity between the two terms, and the propagation factor.

Phase II: The nonlinear evidence combination formulas in the previous subsection are adopted to combine the evidence of pseudo supporting sentences.

Term similarity graphs can be obtained by distributional similarity or patterns (Agirre et al., 2009; Pantel et al., 2009; Shi et al., 2010). We call the first type of graph *DS* and the second type *PB*. DS approaches are based on the distributional hypothesis (Harris, 1985), which says that terms appearing in analogous contexts tend to be similar. In a DS approach, a term is represented by a feature vector, with each feature corresponding to a context in which the term appears. The similarity between two terms is computed as the similarity between their corresponding feature vectors. In PB approaches, a list of carefully-designed (or automatically learned) patterns is exploited and applied to a text collection, with the hypothesis that the terms extracted by applying each of the patterns to a specific piece of text tend to be similar. Two categories of patterns have been studied in the literature (Heast 1992; Pasca 2004; Kozareva et al., 2008; Zhang et al., 2009): sentence lexical patterns, and HTML tag patterns. An example of sentence lexical patterns is "*T {, T}*{,} (and|or) T*". HTML tag patterns include HTML tables, drop-down lists, and other tag repeat patterns. In this paper, we generate the DS and PB graphs by adopting the best-performed methods studied in (Shi et al., 2010). We will compare, by experiments, the propagation performance of utilizing the two categories of graphs, and also investigate the performance of utilizing both graphs for evidence propagation.

6 Experiments

6.1 Experimental setup

Corpus We adopt a publicly available dataset in our experiments: ClueWeb09[4]. This is a very large dataset collected by Carnegie Mellon University in early 2009 and has been used by several tracks of the Text Retrieval Conference (TREC)[5]. The whole dataset consists of 1.04 billion web pages in ten languages while only those in English, about 500 million pages, are used in our experiments. The reason for selecting such a dataset is twofold: First, it is a corpus large enough for conducting web-scale experiments and getting meaningful results. Second, since it is publicly available, it is possible for other researchers to reproduce the experiments in this paper.

Term sets Approaches are evaluated by using two sets of selected terms: *Wiki200*, and *Ext100*. For every term in the term sets, each approach generates a list of hypernym labels, which are manually judged by human annotators. Wiki200 is constructed by first randomly selecting 400 Wikipedia[6] titles as our candidate terms, with the probability of a title T being selected to be $\log(1 + F(T))$, where $F(T)$ is the frequency of T in our data corpus. The reason of adopting such a probability formula is to balance popular terms and rare ones in our term set. Then 200 terms are manually selected from the 400 candidate terms, with the principle of maximizing the diversity of terms in terms of length (i.e., number of words) and type (person, location, organization, software, movie, song, animal, plant, etc.). Wiki200 is further divided into two subsets: *Wiki100H* and *Wiki100L*, containing respectively the 100 high-frequency and low-frequency terms. Ext100 is built by first selecting 200 *non*-Wikipedia-title terms at random from the term-label graph generated by the baseline approach (Formula 3.1), then manually selecting 100 terms.

Some sample terms in the term sets are listed in Table 3.

[4] http://boston.lti.cs.cmu.edu/Data/clueweb09/
[5] http://trec.nist.gov/
[6] http://www.wikipedia.org/

Term Set	Sample Terms
Wiki200	Canon EOS 400D, Disease management, El Salvador, Excellus Blue Cross Blue Shield, F33, Glasstron, Indium, Khandala, Kung Fu, Lake Greenwood, Le Gris, Liriope, Lionel Barrymore, Milk, Mount Alto, Northern Wei, Pink Lady, Shawshank, The Dog Island, White flight, World War II…
Ext100	A2B, Antique gold, GPTEngine, Jinjiang Inn, Moyea SWF to Apple TV Converter, Nanny service, Outdoor living, Plasmid DNA, Popon, Spam detection, Taylor Ho Bynum, Villa Michelle…

Table 3. Sample terms in our term sets

Annotation For each term in the term set, the top-5 results (i.e., hypernym labels) of various methods are mixed and judged by human annotators. Each annotator assigns each result item a judgment of "Good", "Fair" or "Bad". The annotators do not know the method by which a result item is generated. Six annotators participated in the labeling with a rough speed of 15 minutes per term. We also encourage the annotators to add new good results which are not discovered by any method.

The term sets and their corresponding user annotations are available for download at the following links (dataset ID=data.queryset.semcat01):

http://research.microsoft.com/en-us/projects/needleseek/
http://needleseek.msra.cn/datasets/

Evaluation We adopt the following metrics to evaluate the hypernym list of a term generated by each method. The evaluation score on a term set is the average over all the terms.

Precision@*k*: The percentage of relevant (good or fair) labels in the top-*k* results (labels judged as "Fair" are counted as 0.5)

Recall@k: The ratio of relevant labels in the top-*k* results to the total number of relevant labels

R-Precision: Precision@R where *R* is the total number of labels judged as "Good"

Mean average precision (*MAP*): The average of precision values at the positions of all good or fair results

Before annotation and evaluation, the hypernym list generated by each method for each term is preprocessed to remove *duplicate* items. Two hypernyms are called *duplicate* items if they share the same head word (e.g., "military conflict" and "conflict"). For duplicate hypernyms, only the first (i.e., the highest ranked one) in the list is kept. The goal with such a preprocessing step is to partially consider results diversity in evaluation and to make a more meaningful comparison among different methods. Consider two hypernym lists for "subway":

List-1: restaurant; chain restaurant; worldwide chain restaurant; franchise; restaurant franchise…

List-2: restaurant; franchise; transportation; company; fast food…

There are more detailed hypernyms in the first list about "subway" as a restaurant or a franchise; while the second list covers a broader range of meanings for the term. It is hard to say which is better (without considering the upper-layer applications). With this preprocessing step, we keep our focus on short hypernyms rather than detailed ones.

Term Set	Method	MAP	R-Prec	P@1	P@5
Wiki200	Linear	0.357	0.376	0.783	0.547
	Log	0.371 ↑3.92%	0.384 ↑2.13%	0.803 ↑2.55%	0.561 ↑2.56%
	PNorm	0.372 ↑4.20%	0.384 ↑2.13%	0.800 ↑2.17%	0.562 ↑2.74%
Wiki100H	Linear	0.363	0.382	0.805	0.627
	Log	0.393 ↑8.26%	0.402 ↑5.24%	0.845 ↑4.97%	0.660 ↑5.26%
	PNorm	0.395 ↑8.82%	0.403 ↑5.50%	0.840 ↑4.35%	0.662 ↑5.28%

Table 4. Performance comparison among various evidence fusion methods (Term sets: Wiki200 and Wiki100H; p=2 for PNorm)

6.2 Experimental results

We first compare the evaluation results of different evidence fusion methods mentioned in Section 4.1. In Table 4, *Linear* means that Formula 3.1 is used to calculate label scores, whereas *Log* and *PNorm* represent our nonlinear approach with Formulas 4.11 and 4.12 being utilized. The performance improvement numbers shown in the table are based on the linear version; and the upward pointing arrows indicate relative percentage improvement over the baseline. From the table, we can see that the nonlinear methods outperform the linear ones on the Wiki200 term set. It is interesting to note that the performance improvement is more significant on Wiki100H, the set of high frequency terms. By examining the labels and supporting sentences for the terms in each term set, we find that for many low-frequency terms (in Wiki100L), there are only a few supporting sentences (corresponding

to one or two patterns). So the scores computed by various fusion algorithms tend to be similar. In contrast, more supporting sentences can be discovered for high-frequency terms. Much information is contained in the sentences about the hypernyms of the high-frequency terms, but the linear function of Formula 3.1 fails to make effective use of it. The two nonlinear methods achieve better performance by appropriately modeling the dependency between supporting sentences and computing the log-probability gain in a better way.

The comparison of the linear and nonlinear methods on the Ext100 term set is shown in Table 5. Please note that the terms in Ext100 do not appear in Wikipedia titles. Thanks to the scale of the data corpus we are using, even the baseline approach achieves reasonably good performance. Please note that the terms (refer to Table 3) we are using are "harder" than those adopted for evaluation in many existing papers. Again, the results quality is improved with the nonlinear methods, although the performance improvement is not big due to the reason that most terms in Ext100 are rare. Please note that the recall (R@1, R@5) in this paper is pseudo-recall, i.e., we treat the number of *known* relevant (Good or Fair) results as the total number of relevant ones.

Method	MAP	R-Prec	P@1	P@5	R@1	R@5
Linear	0.384	0.429	0.665	0.472	0.116	0.385
Log	0.395	0.429	0.715	0.472	0.125	0.385
	↑2.86%	↑0%	↑7.52%	↑0%	↑7.76%	↑0%
PNorm	0.390	0.429	0.700	0.472	0.120	0.385
	↑1.56%	↑0%	↑5.26%	↑0%	↑3.45%	↑0%

Table 5. Performance comparison among various evidence fusion methods (Term set: Ext100; p=2 for PNorm)

The parameter p in the PNorm method is related to the degree of correlations among supporting sentences. The linear method of Formula 3.1 corresponds to the special case of p=1; while p=∞ represents the case that other supporting sentences are fully correlated to the supporting sentence with the maximal log-probability gain. Figure 1 shows that, for most of the term sets, the best performance is obtained for $p \in$[2.0, 4.0]. The reason may be that the sentence correlations are better estimated with p values in this range.

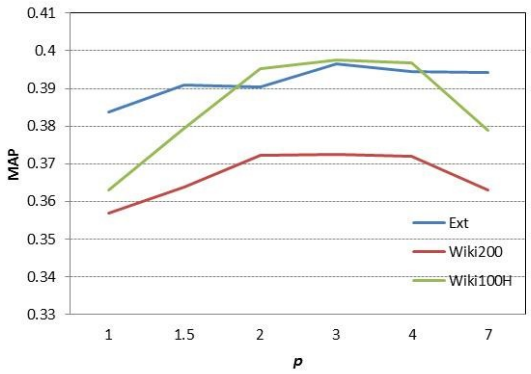


Figure 1. Performance curves of PNorm with different parameter values (Measure: MAP)

The experimental results of evidence propagation are shown in Table 6. The methods for comparison are,

Base: The linear function without propagation.

NL: Nonlinear evidence fusion (PNorm with p=2) without propagation.

LP: Linear propagation, i.e., the linear function is used to combine the evidence of pseudo supporting sentences.

NLP: Nonlinear propagation where PNorm (p=2) is used to combine the pseudo supporting sentences.

NL+NLP: The nonlinear method is used to combine both supporting sentences and pseudo supporting sentences.

Method	MAP	R-Prec	P@1	P@5	R@5
Base	0.357	0.376	0.783	0.547	0.317
NL	0.372	0.384	0.800	0.562	0.325
	↑4.20%	↑2.13%	↑2.17%	↑2.74%	↑2.52%
LP	0.357	0.376	0.783	0.547	0.317
	↑0%	↑0%	↑0%	↑0%	↑0%
NLP	0.396	0.418	0.785	0.605	0.357
	↑10.9%	↑11.2%	↑0.26%	↑10.6%	↑12.6%
NL+NLP	0.447	0.461	0.840	0.667	0.404
	↑25.2%	↑22.6%	↑7.28%	↑21.9%	↑27.4%

Table 6. Evidence propagation results (Term set: Wiki200; Similarity graph: PB; Nonlinear formula: PNorm)

In this paper, we generate the DS (distributional similarity) and PB (pattern-based) graphs by adopting the best-performed methods studied in (Shi et al., 2010). The performance improvement numbers (indicated by the upward pointing arrows) shown in tables 6~9 are relative percentage improvement

over the *base* approach (i.e., linear function without propagation). The values of parameter μ are set to maximize the MAP values.

Several observations can be made from Table 6. First, no performance improvement can be obtained with the linear propagation method (LP), while the nonlinear propagation algorithm (NLP) works quite well in improving both precision and recall. The results demonstrate the high correlation between pseudo supporting sentences and the great potential of using term similarity to improve hypernymy extraction. The second observation is that the NL+NLP approach achieves a much larger performance improvement than NL and NLP. Similar results (omitted due to space limitation) can be observed on the Ext100 term set.

Method	MAP	R-Prec	P@1	P@5	R@5
Base	0.357	0.376	0.783	0.547	0.317
NL+NLP (PB)	0.415 ↑16.2%	0.439 ↑16.8%	0.830 ↑6.00%	0.633 ↑15.7%	0.379 ↑19.6%
NL+NLP (DS)	0.456 ↑27.7%	0.469 ↑24.7%	0.843 ↑7.66%	0.673 ↑23.0%	0.406 ↑28.1%
NL+NLP (PB+DS)	0.473 ↑32.5%	0.487 ↑29.5%	0.860 ↑9.83%	0.700 ↑28.0%	0.434 ↑36.9%

Table 7. Combination of PB and DS graphs for evidence propagation (Term set: Wiki200; Nonlinear formula: Log)

Method	MAP	R-Prec	P@1	P@5	R@5
Base	0.351	0.370	0.760	0.467	0.317
NL+NLP (PB)	0.411 ↑17.1%	0.448 ↑21.1%	0.770 ↑1.32%	0.564 ↑20.8%	0.401 ↑26.5%
NL+NLP (DS)	0.469 ↑33.6%	0.490 ↑32.4%	0.815 ↑7.24%	0.622 ↑33.2%	0.438 ↑38.2%
NL+NLP (PB+DS)	0.491 ↑39.9%	0.513 ↑38.6%	0.860 ↑13.2%	0.654 ↑40.0%	0.479 ↑51.1%

Table 8. Combination of PB and DS graphs for evidence propagation (Term set: Wiki100L)

Now let us study whether it is possible to combine the PB and DS graphs to obtain better results. As shown in Tables 7, 8, and 9 (for term sets Wiki200, Wiki100L, and Ext100 respectively, using the *Log* formula for fusion and propagation), utilizing both graphs really yields additional performance gains. We explain this by the fact that the information in the two term similarity graphs tends to be complimentary. The performance improvement over Wiki100L is especially remarkable. This is reasonable because rare terms do not have adequate information in their supporting sentences due to data sparseness. As a result, they benefit the most from the pseudo supporting sentences propagated with the similarity graphs.

Method	MAP	R-Prec	P@1	P@5	R@5
Base	0.384	0.429	0.665	0.472	0.385
NL+NLP (PB)	0.454 ↑18.3%	0.479 ↑11.7%	0.745 ↑12.0%	0.550 ↑16.5%	0.456 ↑18.4%
NL+NLP (DS)	0.404 ↑5.18%	0.441 ↑2.66%	0.720 ↑8.27%	0.486 ↑2.97%	0.402 ↑4.37%
NL+NLP(PB+DS)	0.483 ↑26.0%	0.518 ↑20.6%	0.760 ↑14.3%	0.586 ↑24.2%	0.492 ↑27.6%

Table 9. Combination of PB and DS graphs for evidence propagation (Term set: Ext100)

7 Conclusion

We demonstrated that the way of aggregating supporting sentences has considerable impact on results quality of the hyponym extraction task using lexico-syntactic patterns, and the widely-used counting method is not optimal. We applied a series of nonlinear evidence fusion formulas to the problem and saw noticeable performance improvement. The data quality is improved further with the combination of nonlinear evidence fusion and evidence propagation. We also introduced a new evaluation corpus with annotated hypernym labels for 300 terms, which were shared with the research community.

Acknowledgments

We would like to thank Matt Callcut for reading through the paper. Thanks to the annotators for their efforts in judging the hypernym labels. Thanks to Yueguo Chen, Siyu Lei, and the anonymous reviewers for their helpful comments and suggestions. The first author is partially supported by the NSF of China (60903028,61070014), and Key Projects in the Tianjin Science and Technology Pillar Program.

References

E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Pasca, and A. Soroa. 2009. A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In Proc. of NAACL-HLT'2009.

M. Banko, M.J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open Information Extraction from the Web. In Proc. of IJCAI'2007.

M. Cafarella, A. Halevy, D. Wang, E. Wu, and Y. Zhang. 2008. WebTables: Exploring the Power of Tables on the Web. In Proceedings of the 34th Conference on Very Large Data Bases (VLDB'2008), pages 538–549, Auckland, New Zealand.

B. Van Durme and M. Pasca. 2008. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. Twenty-Third AAAI Conference on Artificial Intelligence.

F. Geraci, M. Pellegrini, M. Maggini, and F. Sebastiani. 2006. Cluster Generation and Cluster Labelling for Web Snippets: A Fast and Accurate Hierarchical Solution. In Proceedings of the 13th Conference on String Processing and Information Retrieval (SPIRE'2006), pages 25–36, Glasgow, Scotland.

Z. S. Harris. 1985. Distributional Structure. The Philosophy of Linguistics. New York: Oxford University Press.

M. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In Fourteenth International Conference on Computational Linguistics, Nantes, France.

Z. Kozareva, E. Riloff, E.H. Hovy. 2008. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs. In Proc. of ACL'2008.

P. Pantel, E. Crestan, A. Borkovsky, A.-M. Popescu and V. Vyas. 2009. Web-Scale Distributional Similarity and Entity Set Expansion. EMNLP'2009. Singapore.

P. Pantel and D. Ravichandran. 2004. Automatically Labeling Semantic Classes. In Proc. of the 2004 Human Language Technology Conference (HLT-NAACL'2004), 321–328.

M. Pasca. 2004. Acquisition of Categorized Named Entities for Web Search. In Proc. of CIKM'2004.

M. Pasca. 2010. The Role of Queries in Ranking Labeled Instances Extracted from Text. In Proc. of COLING'2010, Beijing, China.

S. Shi, B. Lu, Y. Ma, and J.-R. Wen. 2009. Nonlinear Static-Rank Computation. In Proc. of CIKM'2009, Kong Kong.

S. Shi, H. Zhang, X. Yuan, J.-R. Wen. 2010. Corpus-based Semantic Class Mining: Distributional vs. Pattern-Based Approaches. In Proc. of COLING'2010, Beijing, China.

K. Shinzato and K. Torisawa. 2004. Acquiring Hyponymy Relations from Web Documents. In Proc. of the 2004 Human Language Technology Conference (HLT-NAACL'2004).

R. Snow, D. Jurafsky, and A. Y. Ng. 2005. Learning Syntactic Patterns for Automatic Hypernym Discovery. In Proceedings of the 19th Conference on Neural Information Processing Systems.

R. Snow, D. Jurafsky, and A. Y. Ng. 2006. Semantic Taxonomy Induction from Heterogenous Evidence. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06), 801–808.

P. P. Talukdar and F. Pereira. 2010. Experiments in Graph-based Semi-Supervised Learning Methods for Class-Instance Acquisition. In 48th Annual Meeting of the Association for Computational Linguistics (ACL'2010).

P. P. Talukdar, J. Reisinger, M. Pasca, D. Ravichandran, R. Bhagat, and F. Pereira. 2008. Weakly-Supervised Acquisition of Labeled Class Instances using Graph Random Walks. In Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP'2008), pages 581–589.

R.C. Wang. W.W. Cohen. Automatic Set Instance Extraction using the Web. In Proc. of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-IJCNLP'2009), pages 441–449, Singapore.

H. Zhang, M. Zhu, S. Shi, and J.-R. Wen. 2009. Employing Topic Models for Pattern-based Semantic Class Discovery. In Proc. of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-IJCNLP'2009), pages 441–449, Singapore.

A Pronoun Anaphora Resolution System based on Factorial Hidden Markov Models

Dingcheng Li
University of Minnesota,
Twin Cities, Minnesota
lixxx345@umn.edu

Tim Miller
University of Wisconsin
Milwaukee, Wisconsin
tmill@cs.umn.edu

William Schuler
The Ohio State University
Columbus, Ohio
schuler@ling.osu.edu

Abstract

This paper presents a supervised pronoun anaphora resolution system based on factorial hidden Markov models (FHMMs). The basic idea is that the hidden states of FHMMs are an explicit short-term memory with an antecedent buffer containing recently described referents. Thus an observed pronoun can find its antecedent from the hidden buffer, or in terms of a generative model, the entries in the hidden buffer generate the corresponding pronouns. A system implementing this model is evaluated on the ACE corpus with promising performance.

1 Introduction

Pronoun anaphora resolution is the task of finding the correct antecedent for a given pronominal anaphor in a document. It is a subtask of coreference resolution, which is the process of determining whether two or more linguistic expressions in a document refer to the same entity. Adopting terminology used in the Automatic Context Extraction (ACE) program (NIST, 2003), these expressions are called mentions. Each mention is a reference to some entity in the domain of discourse. Mentions usually fall into three categories – proper mentions (proper names), nominal mentions (descriptions), and pronominal mentions (pronouns). There is a great deal of related work on this subject, so the descriptions of other systems below are those which are most related or which the current model has drawn insight from.

Pairwise models (Yang et al., 2004; Qiu et al., 2004) and graph-partitioning methods (McCallum and Wellner, 2003) decompose the task into a collection of pairwise or mention set coreference decisions. Decisions for each pair or each group of mentions are based on probabilities of features extracted by discriminative learning models. The aforementioned approaches have proven to be fruitful; however, there are some notable problems. Pairwise modeling may fail to produce coherent partitions. That is, if we link results of pairwise decisions to each other, there may be conflicting coreferences. Graph-partitioning methods attempt to reconcile pairwise scores into a final coherent clustering, but they are combinatorially harder to work with in discriminative approaches.

One line of research aiming at overcoming the limitation of pairwise models is to learn a mention-ranking model to rank preceding mentions for a given anaphor (Denis and Baldridge, 2007) This approach results in more coherent coreference chains.

Recent years have also seen the revival of interest in generative models in both machine learning and natural language processing. Haghighi and Klein (2007), proposed an unsupervised non-parametric Bayesian model for coreference resolution. In contrast to pairwise models, this fully generative model produces each mention from a combination of global entity properties and local attentional state. Ng (2008) did similar work using the same unsupervised generative model, but relaxed head generation as head-index generation, enforced agreement constraints at the global level, and assigned salience only to pronouns.

Another unsupervised generative model was recently presented to tackle only pronoun anaphora

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1169–1178,
Portland, Oregon, June 19-24, 2011.

resolution (Charniak and Elsner, 2009). The expectation-maximization algorithm (EM) was applied to learn parameters automatically from the parsed version of the North American News Corpus (McClosky et al., 2008). This model generates a pronoun's person, number and gender features along with the governor of the pronoun and the syntactic relation between the pronoun and the governor. This inference process allows the system to keep track of multiple hypotheses through time, including multiple different possible histories of the discourse.

Haghighi and Klein (2010) improved their nonparametric model by sharing lexical statistics at the level of abstract entity types. Consequently, their model substantially reduces semantic compatibility errors. They report the best results to date on the complete end-to-end coreference task. Further, this model functions in an online setting at mention level. Namely, the system identifies mentions from a parse tree and resolves resolution with a left-to-right sequential beam search. This is similar to Luo (2005) where a Bell tree is used to score and store the searching path.

In this paper, we present a supervised pronoun resolution system based on Factorial Hidden Markov Models (FHMMs). This system is motivated by human processing concerns, by operating incrementally and maintaining a limited short term memory for holding recently mentioned referents. According to Clark and Sengul (1979), anaphoric definite NPs are much faster retrieved if the antecedent of a pronoun is in immediately previous sentence. Therefore, a limited short term memory should be good enough for resolving the majority of pronouns. In order to construct an operable model, we also measured the average distance between pronouns and their antecedents as discussed in next sections and used distances as important salience features in the model.

Second, like Morton (2000), the current system essentially uses prior information as a discourse model with a time-series manner, using a dynamic programming inference algorithm. Third, the FHMM described here is an integrated system, in contrast with (Haghighi and Klein, 2010). The model generates part of speech tags as simple structural information, as well as related semantic information at each time step or word-by-word step. While the framework described here can be extended to deeper structural information, POS tags alone are valuable as they can be used to incorporate the binding features (described below).

Although the system described here is evaluated for pronoun resolution, the framework we describe can be extended to more general coreference resolution in a fairly straightforward manner. Further, as in other HMM-based systems, the system can be either supervised or unsupervised. But extensions to unsupervised learning are left for future work.

The final results are compared with a few supervised systems as the mention-ranking model (Denis and Baldridge, 2007) and systems compared in their paper, and Charniak and Elsner's (2009) unsupervised system, emPronouns. The FHMM-based pronoun resolution system does a better job than the global ranking technique and other approaches. This is a promising start for this novel FHMM-based pronoun resolution system.

2 Model Description

This work is based on a graphical model framework called Factorial Hidden Markov Models (FHMMs). Unlike the more commonly known Hidden Markov Model (HMM), in an FHMM the hidden state at each time step is expanded to contain more than one random variable (as shown in Figure 1). This allows for the use of more complex hidden states by taking advantage of conditional independence between substates. This conditional independence allows complex hidden states to be learned with limited training data.

2.1 Factorial Hidden Markov Model

Factorial Hidden Markov Models are an extension of HMMs (Ghahramani and Jordan, 1997). HMMs represent sequential data as a sequence of hidden states generating observation states (words in this case) at corresponding time steps t. A most likely sequence of hidden states can then be hypothesized given any sequence of observed states, using Bayes Law (Equation 2) and Markov independence assumptions (Equation 3) to define a full probability as the product of a Transition Model (Θ_T) prior probability and an Observation Model (Θ_O) likelihood

probability.

$$\hat{h}_{1..T} \stackrel{\text{def}}{=} \underset{h_{1..T}}{\text{argmax}}\ \mathsf{P}(h_{1..T} \mid o_{1..T}) \quad (1)$$

$$\stackrel{\text{def}}{=} \underset{h_{1..T}}{\text{argmax}}\ \mathsf{P}(h_{1..T}) \cdot \mathsf{P}(o_{1..T} \mid h_{1..T}) \quad (2)$$

$$\stackrel{\text{def}}{=} \underset{h_{1..T}}{\text{argmax}} \prod_{t=1}^{T} \mathsf{P}_{\Theta_T}(h_t \mid h_{t-1}) \cdot \mathsf{P}_{\Theta_O}(o_t \mid h_t) \quad (3)$$

For a simple HMM, the hidden state corresponding to each observation state only involves one variable. An FHMM contains more than one hidden variable in the hidden state. These hidden substates are usually layered processes that jointly generate the evidence. In the model described here, the substates are also coupled to allow interaction between the separate processes. As Figure 1 shows, the hidden states include three sub-states, op, cr and pos which are short forms of *operation*, *coreference feature* and *part-of-speech*. Then, the transition model expands the left term in (3) to (4).

$$\begin{aligned}\mathsf{P}_{\Theta_T}(h_t \mid h_{t-1}) \stackrel{\text{def}}{=} & \mathsf{P}(op_t \mid op_{t-1}, pos_{t-1}) \\ & \cdot \mathsf{P}(cr_t \mid cr_{t-1}, op_{t-1}) \\ & \cdot \mathsf{P}(pos_t \mid op_t, pos_{t-1})\end{aligned} \quad (4)$$

The observation model expands from the right term in (3) to (5).

$$\mathsf{P}_{\Theta_O}(o_t \mid h_t) \stackrel{\text{def}}{=} \mathsf{P}(o_t \mid pos_t, cr_t) \quad (5)$$

The observation state depends on more than one hidden state at each time step in an FHMM. Each hidden variable can be further split into smaller variables. What these terms stand for and the motivations behind the above equations will be explained in the next section.

2.2 Modeling a Coreference Resolver with FHMMs

FHMMs in our model, like standard HMMs, cannot represent the hierarchical structure of a syntactic phrase. In order to partially represent this information, the head word is used to represent the whole noun phrase. After coreference is resolved, the coreferring chain can then be expanded to the whole phrase with NP chunker tools.

In this system, hidden states are composed of three main variables: a referent operation (*OP*), coreference features (*CR*) and part of speech tags (*POS*) as displayed in Figure 1. The transition model is defined as Equation 4.

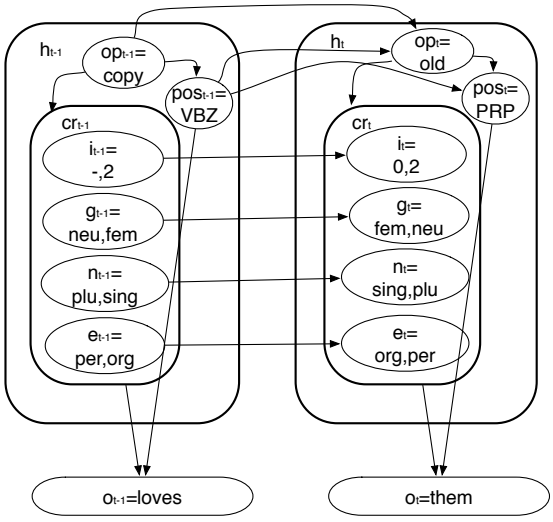


Figure 1: Factorial HMM CR Model

The starting point for the hidden state at each time step is the *OP* variable, which determines which kind of referent operations will occur at the current word. Its domain has three possible states: *none*, *new* and *old*.

The *none* state indicates that the present state will not generate a mention. All previous hidden state values (the list of previous mentions) will be passed deterministically (with probability 1) to the current time step without any changes. The *new* state signifies that there is a new mention in the present time step. In this event, a new mention will be added to the entity set, as represented by its set of feature values and position in the coreference table. The *old* state indicates that there is a mention in the present time state and that this mention refers back to some antecedent mention. In such a case, the list of entities in the buffer will be reordered deterministically, moving the currently mentioned entity to the top of the list.

Notice that op_t is defined to depend on op_{t-1} and pos_{t-1}. This is sometimes called a *switching* FHMM (Duh, 2005). This dependency can be useful, for example, if op_{t-1} is *new*, in which case op_t has a higher probability of being *none* or *old*. If

pos_{t-1} is a verb or preposition, op_t has more probability of being *old* or *new*.

One may wonder why op_t generates pos_t, and not the other way around. This model only roughly models the process of (new and old) entity generation, and either direction of causality might be consistent with a model of human entity generation, but this direction of causality is chosen to represent the effect of semantics (referents) generating syntax (POS tags). In addition, this is a joint model in which POS tagging and coreference resolution are integrated together, so the best combination of those hidden states will be computed in either case.

2.3 Coreference Features

Coreference features for this model refer to features that may help to identify co-referring entities.

In this paper, they mainly include index (*I*), named entity type (*E*), number (*N*) and gender (*G*). The index feature represents the order that a mention was encountered relative to the other mentions in the buffer. The latter three features are well known and described elsewhere, and are not themselves intended as the contribution of this work. The novel aspect of this part of the model is the fact that the features are carried forward, updated after every word, and essentially act as a discourse model. The features are just a shorthand way of representing some well known essential aspects of a referent (as pertains to anaphora resolution) in a discourse model.

Features	Values
I	positive integers from 1. . .n
G	male, female, neutral, unknown
N	singular, plural, unknown
E	person, location, organization, GPE, vehicle, company, facility

Table 1: Coreference features stored with each mention.

Unlike discriminative approaches, generative models like the FHMM described here do not have access to all observations at once. This model must then have a mechanism for jointly considering pronouns in tandem with previous mentions, as well as the features of those mentions that might be used to find matches between pronouns and antecedents.

Further, higher order HMMs may contain more accurate information about observation states. This is especially true for coreference resolution because pronouns often refer back to mentions that are far away from the present state. In this case, we would need to know information about mentions which are at least two mentions before the present one. In this sense, a higher order HMM may seem ideal for coreference resolution. However, higher order HMMs will quickly become intractable as the order increases.

In order to overcome these limitations, two strategies which have been discussed in the last section are taken: First, a switching variable called *OP* is designed (as discussed in last section); second, a memory of recently mentioned entities is maintained to store features of mentions and pass them forward incrementally.

OP is intended to model the decision to use the current word to introduce a new referent (*new*), refer to an antecedent (*old*), or neither (*none*). The entity buffer is intended to model the set of 'activated' entities in the discourse – those which could plausibly be referred to with a pronoun. These designs allow similar benefits as longer dependencies of higher-order HMMs but avoid the problem of intractability. The number of mentions maintained must be limited in order for the model to be tractable. Fortunately, human short term memory faces effectively similar limitations and thus pronouns usually refer back to mentions not very far away.

Even so, the impact of the size of the buffer on decoding time may be a concern. Since the buffer of our system will carry forward a few previous groups of coreference features plus *op* and *pos*, the computational complexity will be exorbitantly high if we keep high beam size and meanwhile if each feature interacts with others. Luckily, we have successfully reduced the intractability to a workable system in both speed and space with following methods. First, we estimate the size of buffer with a simple count of average distances between pronouns and their antecedents in the corpus. It is found that about six is enough for covering 99.2% of all pronouns.

Secondly, the coreference features we have used have the nice property of being independent from one another. One might expect English non-person entities to almost always have neutral gender, and

thus be modeled as follows:

$$\mathsf{P}(e_t, g_t \mid e_{t-1}, g_{t-1}) = \mathsf{P}(g_t \mid g_{t-1}, e_t) \cdot \mathsf{P}(e_t \mid e_{t-1}) \quad (6)$$

However, a few considerations made us reconsider. First, exceptions are found in the corpus. Personal pronouns such as *she* or *he* are used to refer to country, regions, states or organizations. Second, existing model files made by Bergsma (2005) include a large number of non-neutral gender information for non-person words. We employ these files for acquiring gender information of unknown words. If we use Equation 6, sparsity and complexity will increase. Further, preliminary experiments have shown models using an independence assumption between gender and personhood work better. Thus, we treat each coreference feature as an independent event. Hence, we can safely split coreference features into separate parts. This way dramatically reduces the model complexity. Thirdly, our HMM decoding uses the Viterbi algorithm with A-star beam search.

The probability of the new state of the coreference table $\mathsf{P}(cr_t \mid cr_{t-1}, op_t)$ is defined to be the product of probabilities of the individual feature transitions.

$$\begin{aligned} \mathsf{P}(cr_t \mid cr_{t-1}, op_t) = \mathsf{P}(i_t \mid i_{t-1}, op_t) \cdot \\ \mathsf{P}(e_t \mid e_{t-1}, i_t, op_t) \cdot \\ \mathsf{P}(g_t \mid g_{t-1}, i_t, op_t) \cdot \\ \mathsf{P}(n_t \mid n_{t-1}, i_t, op_t) \end{aligned} \quad (7)$$

This supposes that the features are conditionally independent of each other given the index variable, the operator and previous instance. Each feature only depends on the operator and the corresponding feature at the previous state, with that set of features re-ordered as specified by the index model.

2.4 Feature Passing

Equation 7 is correct and complete, but in fact the switching variable for operation type results in three different cases which simplifies the calculation of the transition probabilities for the coreference feature table.

Note the following observations about coreference features: i_t only needs a probabilistic model when op_t is *old* – in other words, only when the model must choose between several antecedents to re-refer to. g_t, e_t and n_t are deterministic except when op_t is *new*, when gender, entity type, and number information must be generated for the new entity being introduced.

When op_t is *none*, all coreference variables (entity features) will be copied over from the previous time step to the current time step, and the probability of this transition is 1.0. When op_t is *new*, i_t is changed deterministically by adding the new entity to the first position in the list and moving every other entity down one position. If the list of entities is full, the least recently mentioned entity will be discarded. The values for the top of the feature lists g_t, e_t, and n_t will then be generated from feature-specific probability distributions estimated from the training data. When op_t is *old*, i_t will probabilistically select a value $1 \ldots n$, for an entity list containing n items. The selected value will deterministically order the g_t, n_t and e_t lists. This distribution is also estimated from training data, and takes into account recency of mention. The shape of this distribution varies slightly depending on list size and noise in the training data, but in general the probability of a mention being selected is directly correlated to how recently it was mentioned.

With this understanding, coreference table transition probabilities can be written in terms of only their non-deterministic substate distributions:

$$\begin{aligned} \mathsf{P}(cr_t \mid cr_{t-1}, old) = \; & \mathsf{P}_{old}(i_t \mid i_{t-1}) \cdot \\ & \mathsf{P}_{reorder}(e_t \mid e_{t-1}, i_t) \cdot \\ & \mathsf{P}_{reorder}(g_t \mid g_{t-1}, i_t) \cdot \\ & \mathsf{P}_{reorder}(n_t \mid n_{t-1}, i_t) \end{aligned} \quad (8)$$

where the *old* model probabilistically selects the antecedent and moves it to the top of the list as described above, thus deciding how the reordering will take place. The *reorder* model actually implements the list reordering for each independent feature by moving the feature value corresponding to the selected entity in the index model to the top of that feature's list. The overall effect is simply the probabilistic reordering of entities in a list, where each entity is defined as a label and a set of features.

$$\begin{aligned} \mathsf{P}(cr_t \mid cr_{t-1}, new) = \; & \mathsf{P}_{new}(i_t \mid i_{t-1}) \cdot \\ & \mathsf{P}_{new}(g_t \mid g_{t-1}) \cdot \\ & \mathsf{P}_{new}(n_t \mid n_{t-1}) \cdot \\ & \mathsf{P}_{new}(e_t \mid e_{t-1}) \end{aligned} \quad (9)$$

where the *new* model probabilistically generates a

feature value based on the training data and puts it at the top of the list, moves every other entity down one position in the list, and removes the final item if the list is already full. Each entity in *i* takes a value from 1 to n for a list of size n. Each *g* can be one of four values – *male*, *female*, *neuter* and *unknown*; *n* one of three values – *plural*, *singular* and *unknown* and *e* around eight values.

Note that pos_t is used in both hidden states and observation states. While it is not considered a coreference feature as such, it can still play an important role in the resolving process. Basically, the system tags parts of speech incrementally while simultaneously resolving pronoun anaphora. Meanwhile, pos_{t-1} and op_{t-1} will jointly generate op_t. This point has been discussed in Section 2.2.

Importantly, the pos model can help to implement binding principles (Chomsky, 1981). It is applied when op_t is old. In training, pronouns are sub-categorised into personal pronouns, reflexive and other-pronoun. We then define a variable loc_t whose value is how far back in the list of antecedents the current hypothesis must have gone to arrive at the current value of i_t. If we have the syntax annotations or parsed trees, then, the part of speech model can be defined when op_t is *old* as $\mathsf{P}_{binding}(pos_t \,|\, loc_t, s_{loc_t})$. For example, if $pos_t \in reflexive$, $\mathsf{P}(pos_t \,|\, loc_t, s_{loc_t})$ where loc_t has smaller values (implying closer mentions to pos_t) and $s_{loc_t} = subject$ should have higher values since reflexive pronouns always refer back to subjects within its governing domains. This was what (Haghighi and Klein, 2009) did and we did this in training with the REUTERS corpus (Hasler et al., 2006) in which syntactic roles are annotated. We finally switched to the ACE corpus for the purpose of comparison with other work. In the ACE corpus, no syntactic roles are annotated. We did use the Stanford parser to extract syntactic roles from the ACE corpus. But the result is largely affected by the parsing accuracy. Again, for a fair comparison, we extract similar features to Denis and Baldridge (2007), which is the model we mainly compare with. They approximate syntactic contexts with POS tags surrounding the pronoun. Inspired by this idea, we successfully represent binding features with POS tags before anaphors. Instead of using $\mathsf{P}(pos_t \,|\, loc_t, s_{loc_t})$, we train $\mathsf{P}(pos_t \,|\, loc_t, pos_{loc_t})$ which can play the role of binding. For example, suppose the buffer size is 6 and $loc_t = 5$, $pos_{loc_t} = noun$. Then, $\mathsf{P}(pos_t = reflexive \,|\, loc_t, pos_{loc_t})$ is usually higher than $\mathsf{P}(pos_t = pronoun \,|\, loc_t, pos_{loc_t})$, since the reflexive has a higher probability of referring back to the noun located in position 5 than the pronoun.

In future work expanding to coreference resolution between any noun phrases we intend to integrate syntax into this framework as a joint model of coreference resolution and parsing.

3 Observation Model

The observation model that generates an observed state is defined as Equation 5. To expand that equation in detail, the observation state, the word, depends on its part of speech and its coreference features as well. Since FHMMs are generative, we can say part of speech and coreference features generate the word.

In actual implementation, the observed model will be very sparse, since cr_t will be split into more variables according to how many coreference features it is composed of. In order to avoid the sparsity, we transform the equation with Bayes' law as follows.

$$\mathsf{P}_{\Theta_O}(o_t \,|\, h_t) = \frac{P(o_t) \cdot \mathsf{P}(h_t \,|\, o_t)}{\sum_{o'} P(o')\mathsf{P}(h_t \,|\, o')} \tag{10}$$

$$= \frac{P(o_t) \cdot \mathsf{P}(pos_t, cr_t \,|\, o_t)}{\sum_{o'} P(o')\mathsf{P}(pos_t, cr_t \,|\, o')} \tag{11}$$

We define *pos* and *cr* to be independent of each other, so we can further split the above equation as:

$$\mathsf{P}_{\Theta_O}(o_t \,|\, h_t) \stackrel{\text{def}}{=} \frac{P(o_t) \cdot \mathsf{P}(pos_t \,|\, o_t) \cdot \mathsf{P}(cr_t \,|\, o_t)}{\sum_{o'} P(o') \cdot \mathsf{P}(pos_t \,|\, o') \cdot \mathsf{P}(cr_t \,|\, o')} \tag{12}$$

where $\mathsf{P}(cr_t \,|\, o_t) = \mathsf{P}(g_t \,|\, o_t)\mathsf{P}(n_t \,|\, o_t)\mathsf{P}(e_t \,|\, o_t)$ and $\mathsf{P}(cr_t \,|\, o') = \mathsf{P}(g_t \,|\, o')\mathsf{P}(n_t \,|\, o')\mathsf{P}(e_t \,|\, o')$.

This change transforms the FHMM to a hybrid FHMM since the observation model no longer generates the data. Instead, the observation model generates hidden states, which is more a combination of discriminative and generative approaches. This way facilitates building likelihood model files of features for given mentions from the training data. The

hidden state transition model represents prior probabilities of coreference features associated with each while this observation model factors in the probability given a pronoun.

3.1 Unknown Words Processing

If an observed word was not seen in training, the distribution of its part of speech, gender, number and entity type will be unknown. In this case, a special unknown words model is used.

The part of speech of unknown words $\mathsf{P}(pos_t \mid w_t = unkword)$ is estimated using a decision tree model. This decision tree is built by splitting letters in words from the end of the word backward to its beginning. A POS tag is assigned to the word after comparisons between the morphological features of words trained from the corpus and the strings concatenated from the tree leaves are made. This method is about as accurate as the approach described by Klein and Manning (2003).

Next, a similar model is set up for estimating $\mathsf{P}(n_t \mid w_t = unkword)$. Most English words have regular plural forms, and even irregular words have their patterns. Therefore, the morphological features of English words can often be used to determine whether a word is singular or plural.

Gender is irregular in English, so model-based predictions are problematic. Instead, we follow Bergsma and Lin (2005) to get the distribution of gender from their gender/number data and then predict the gender for unknown words.

4 Evaluation and Discussion

4.1 Experimental Setup

In this research, we used the ACE corpus (Phase 2) [1] for evaluation. The development of this corpus involved two stages. The first stage is called EDT (entity detection and tracking) while the second stage is called RDC (relation detection and characterization). All markables have named entity types such as FACILITY, GPE (geopolitical entity), PERSON, LOCATION, ORGANIZATION, PERSON, VEHICLE and WEAPONS, which were annotated in the first stage. In the second stage, relations between named entities were annotated. This corpus include three parts, composed of different genres: newspaper texts (NPAPER), newswire texts (NWIRE) and broadcasted news (BNEWS). Each of these is split into a *train* part and a *devtest* part. For the train part, there are 76, 130 and 217 articles in NPAPER, NWIRE and BNEWS respectively while for the test part, there are 17, 29 and 51 articles respectively. Though the number of articles are quite different for three genres, the total number of words are almost the same. Namely, the length of NPAPER is much longer than BNEWS (about 1200 words, 800 word and 500 words respectively for three genres). The longer articles involve longer coreference chains. Following the common practice, we used the *devtest* material only for testing. Progress during the development phase was estimated only by using cross-validation on the training set for the BNEWS section. In order to make comparisons with publications which used the same corpus, we make efforts to set up identical conditions for our experiments.

The main point of comparison is Denis and Baldridge (2007), which was similar in that it described a new type of coreference resolver using simple features.

Therefore, similar to their practice, we use all forms of personal and possessive pronouns that were annotated as ACE "markables". Namely, pronouns associated with named entity types could be used in this system. In experiments, we also used *true* ACE mentions as they did. This means that pleonastics and references to eventualities or to non-ACE entities are not included in our experiments either. In all, 7263 referential pronouns in training data set and 1866 in testing data set are found in all three genres. They have results of three different systems: SCC (single candidate classifier), TCC (twin candidate classifier) and RK (ranking). Besides the three and our own system, we also report results of emPronouns, which is an unsupervised system based on a recently published paper (Charniak and Elsner, 2009). We select this unsupervised system for two reasons. Firstly, emPronouns is a publicly available system with high accuracy in pronoun resolution. Secondly, it is necessary for us to demonstrate our system has strong empirical superiority over unsupervised ones. In testing, we also used the OPNLP Named Entity Recognizer to tag the test corpus.

[1] See `http://projects.ldc.upenn.edu/ace/annotation/previous/` for details on the corpus.

During training, besides coreference annotation itself, the part of speech, dependencies between words and named entities, gender, number and index are extracted using relative frequency estimation to train models for the coreference resolution system.

Inputs for testing are the plain text and the trained model files. The entity buffer used in these experiments kept track of only the six most recent mentions. The result of this process is an annotation of the headword of every noun phrase denoting it as a mention. In addition, this system does not do anaphoricity detection, so the antecedent operation for non-anaphora pronoun *it* is set to be *none*. Finally, the system does not yet model cataphora, about 10 cataphoric pronouns in the testing data which are all counted as wrong.

4.2 Results

The performance was evaluated using the ratio of the number of correctly resolved anaphors over the number of all anaphors as a success metrics. All the standards are consistent with those defined in Charniak and Elsner (2009).

During development, several preliminary experiments explored the effects of starting from a simple baseline and adding more features. The BNEWS corpus was employed in these development experiments. The baseline only includes part of speech tags, the index feature and and syntactic roles. Syntactic roles are extracted from the parsing results with Stanford parser. The success rate of this baseline configuration is 0.48. This low accuracy is partially due to the errors of automatic parsing. With gender and number features added, the performance jumped to 0.65. This shows that number and gender agreements play an important role in pronoun anaphora resolution. For a more standard comparison to other work, subsequent tests were performed on the gold standard ACE corpus (using the model as described with named entity features instead of syntactic role features). As shown in Denis and Baldridge (2007), they employ all features we use except syntactic roles. In these experiments, the system got better results as shown in Table 2.

The result of the first one is obtained by running the publicly available system emPronouns[2]. It is a high-accuracy unsupervised system which reported the best result in Charniak and Elsner (2009).

System	BNEWS	NPAPER	NWIRE
emPronouns	58.5	64.5	60.6
SCC	62.2	70.7	68.3
TCC	68.6	74.7	71.1
RK	72.9	76.4	72.4
FHMM	**74.9**	**79.4**	**74.5**

Table 2: Accuracy scores for emPronouns, the single-candidate classifier (SCC), the twin-candidate classifier (TCC), the ranker and FHMM

The results of the other three systems are those reported by Denis and Baldridge (2007). As Table 2 shows, the FHMM system gets the highest average results.

The emPronouns system got the lowest results partially due to the reason that we only directly run the existing system with its existing model files without retraining. But the gap between its results and results of our system is large. Thus, we may still say that our system probably can do a better job even if we train new models files for emPronouns with ACE corpus.

With almost exactly identical settings, why does our FHMM system get the highest average results? The convincing reason is that FHMM is strongly influenced by the sequential dependencies. The ranking approach ranks a set of mentions using a set of features, and it also maintains the discourse model, but it is not processing sequentially. The FHMM system always maintain a set of mentions as well as a first-order dependencies between part of speech and operator. Therefore, context can be more fully taken into consideration. This is the main reason that the FHMM approach achieved better results than the ranking approach.

From the result, one point we may notice is that NPAPER usually obtains higher results than both BNEWS and NWIRE for all systems while BNEWS lower than other two genres. In last section, we mention that articles in NPAPER are longer than other genres and also have denser coreference chains while articles in BENEWS are shorter and have sparer chains. Then, it is not hard to understand why results of NPAPER are better while those of

[2]the available system in fact only includes the testing part. Thus, it may be unfair to compare emPronouns this way with other systems.

BNEWS are poorer.

In Denis and Baldridge (2007), they also reported new results with a window of 10 sentences for RK model. All three genres obtained higher results than those when with shorter ones. They are 73.0, 77.6 and 75.0 for BNEWS, NPAPER and NWIRE respectively. We can see that except the one for NWIRE, the results are still poorer than our system. For NWIRE, the RK model got 0.5 higher. The average of the RK is 75.2 while that of the FHMM system is 76.3, which is still the best.

Since the emPronoun system can output sample-level results, it is possible to do a paired Student's t-test. That test shows that the improvement of our system on all three genres is statistically significant ($p < 0.001$). Unfortunately, the other systems only report overall results so the same comparison was not so straightforward.

4.3 Error Analysis

After running the system on these documents, we checked which pronouns fail to catch their antecedents. There are a few general reasons for errors.

First, pronouns which have antecedents very far away cannot be caught. Long-distance anaphora resolution may pose a problem since the buffer size cannot be too long considering the complexity of tracking a large number of mentions through time. During development, estimation of an acceptable size was attempted using the training data. It was found that a mention distance of fourteen would account for every case found in this corpus, though most cases fall well short of that distance. Future work will explore optimizations that will allow for larger or variable buffer sizes so that longer distance anaphora can be detected.

A second source of error is simple misjudgments when more than one candidate is waiting for selection. A simple case is that the system fails to distinguish plural personal nouns and non-personal nouns if both candidates are plural. This is not a problem for singular pronouns since gender features can tell whether pronouns are personal or not. Plural nouns in English do not have such distinctions, however. Consequently, *demands* and *Israelis* have the same probability of being selected as the antecedents for *they*, all else being equal. If *demands* is closer to *they*, *demands* will be selected as the antecedent. This may lead to the wrong choice if *they* in fact refers to *Israelis*. This may require better measures of referent salience than the "least recently used" heuristic currently implemented.

Third, these results also show difficulty resolving coordinate noun phrases due to the simplistic representation of noun phrases in the input. Consider this sentence: *President Barack Obama and his wife Michelle Obama visited China last week. They had a meeting with President Hu in Beijing*. In this example, the pronoun *they* corefers with the noun phrase *President Barack Obama and his wife Michelle Obama*. The present model cannot represent both the larger noun phrase and its contained noun phrases. Since the noun phrase is a coordinate one that includes both noun phrases, the model cannot find a head word to represent it.

Finally, while the coreference feature annotations of the ACE are valuable for learning feature models, the model training may still give some misleading results. This is brought about by missing features in the training corpus and by the data sparsity. We solved the problem with add-one smoothing and deleted interpolation in training models besides the transformation in the generation order of the observation model.

5 Conclusion and Future Work

This paper has presented a pronoun anaphora resolution system based on FHMMs. This generative system incrementally resolves pronoun anaphora with an entity buffer carrying forward mention features. The system performs well and outperforms other available models. This shows that FHMMs and other time-series models may be a valuable model to resolve anaphora.

Acknowledgments

We would like to thank the authors and maintainers of ranker models and emPronouns. We also would like to thank the three anonymous reviewers. The final version is revised based on their valuable comments. Thanks are extended to Shane Bergsma, who provided us the gender and number data distribution. In addition, Professor Jeanette Gundel and our labmate Stephen Wu also gave us support in paper editing and in theoretical discussion.

References

S Bergsma. 2005. Automatic acquisition of gender information for anaphora resolution. page 342353. Springer.

Eugene Charniak and Micha Elsner. 2009. Em works for pronoun anaphora resolution. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL-09)*, Athens, Greece.

Noam Chomsky. 1981. *Lectures on government and binding*. Foris, Dordercht.

H.H. Clark and CJ Sengul. 1979. In search of referents for nouns and pronouns. *Memory & Cognition*, 7(1):35–41.

P. Denis and J. Baldridge. 2007. A ranking approach to pronoun resolution. In *Proc. IJCAI*.

Kevin Duh. 2005. Jointly labeling multiple sequences: a factorial HMM approach. In *ACL '05: Proceedings of the ACL Student Research Workshop*, pages 19–24, Ann Arbor, Michigan.

Zoubin Ghahramani and Michael I. Jordan. 1997. Factorial hidden markov models. *Machine Learning*, 29:1–31.

A. Haghighi and D. Klein. 2007. Unsupervised coreference resolution in a nonparametric bayesian model. In *Proceedings of the 45th annual meeting on Association for Computational Linguistics*, page 848.

A. Haghighi and D. Klein. 2009. Simple coreference resolution with rich syntactic and semantic features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1152–1161. Association for Computational Linguistics.

A. Haghighi and D. Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 385–393. Association for Computational Linguistics.

L. Hasler, C. Orasan, and K. Naumann. 2006. NPs for events: Experiments in coreference annotation. In *Proceedings of the 5th edition of the International Conference on Language Resources and Evaluation (LREC2006)*, pages 1167–1172. Citeseer.

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

X Luo. 2005. On coreference resolution performance metrics. pages 25–32. Association for Computational Linguistics Morristown, NJ, USA.

A. McCallum and B. Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *IJCAI Workshop on Information Integration on the Web*. Citeseer.

David McClosky, Eugene Charniak, and Mark Johnson. 2008. BLLIP North American News Text, Complete. *Linguistic Data Consortium. LDC2008T13*.

T.S. Morton. 2000. Coreference for NLP applications. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

V. Ng. 2008. Unsupervised models for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 640–649. Association for Computational Linguistics.

US NIST. 2003. The ACE 2003 Evaluation Plan. *US National Institute for Standards and Technology (NIST), Gaithersburg, MD.[online*, pages 2003–08.

L. Qiu, M.Y. Kan, and T.S. Chua. 2004. A public reference implementation of the rap anaphora resolution algorithm. *Arxiv preprint cs/0406031*.

X. Yang, J. Su, G. Zhou, and C.L. Tan. 2004. Improving pronoun resolution by incorporating coreferential information of candidates. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 127. Association for Computational Linguistics.

Disentangling Chat with Local Coherence Models

Micha Elsner
School of Informatics
University of Edinburgh
melsner0@gmail.com

Eugene Charniak
Department of Computer Science
Brown University, Providence, RI 02912
ec@cs.brown.edu

Abstract

We evaluate several popular models of local discourse coherence for domain and task generality by applying them to chat disentanglement. Using experiments on synthetic multiparty conversations, we show that most models transfer well from text to dialogue. Coherence models improve results overall when good parses and topic models are available, and on a constrained task for real chat data.

1 Introduction

One property of a well-written document is *coherence*, the way each sentence fits into its context– sentences should be interpretable in light of what has come before, and in turn make it possible to interpret what comes after. Models of coherence have primarily been used for text-based generation tasks: ordering units of text for multidocument summarization or inserting new text into an existing article. In general, the corpora used consist of informative writing, and the tasks used for evaluation consider different ways of reordering the same set of textual units. But the theoretical concept of coherence goes beyond both this domain and this task setting– and so should coherence models.

This paper evaluates a variety of local coherence models on the task of chat disentanglement or "threading": separating a transcript of a multiparty interaction into independent conversations[1]. Such simultaneous conversations occur in internet chat rooms, and on shared voice channels such as push-to-talk radio. In these situations, a single, correctly disentangled, conversational thread will be coherent, since the speakers involved understand the normal rules of discourse, but the transcript as a whole will not be. Thus, a good model of coherence should be able to disentangle sentences as well as order them.

There are several differences between disentanglement and the newswire sentence-ordering tasks typically used to evaluate coherence models. Internet chat comes from a different domain, one where topics vary widely and no reliable syntactic annotations are available. The disentanglement task measures different capabilities of a model, since it compares documents that are not permuted versions of one another. Finally, full disentanglement requires a large-scale search, which is computationally difficult. We move toward disentanglement in stages, carrying out a series of experiments to measure the contribution of each of these factors.

As an intermediary between newswire and internet chat, we adopt the SWITCHBOARD (SWBD) corpus. SWBD contains recorded telephone conversations with known topics and hand-annotated parse trees; this allows us to control for the performance of our parser and other informational resources. To compare the two algorithmic settings, we use SWBD for ordering experiments, and also artificially entangle pairs of telephone dialogues to create synthetic transcripts which we can disentangle. Finally, we present results on actual internet chat corpora.

On synthetic SWBD transcripts, local coherence models improve performance considerably over our baseline model, Elsner and Charniak (2008b). On

[1] A public implementation is available via https://bitbucket.org/melsner/browncoherence.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1179–1189,
Portland, Oregon, June 19-24, 2011.

internet chat, we continue to do better on a constrained disentanglement task, though so far, we are unable to apply these improvements to the full task. We suspect that, with better low-level annotation tools for the chat domain and a good way of integrating prior information, our improvements on SWBD could transfer fully to IRC chat.

2 Related work

There is extensive previous work on coherence models for text ordering; we describe several specific models below, in section 2. This study focuses on models of local coherence, which relate text to its immediate context. There has also been work on global coherence, the structure of a document as a whole (Chen et al., 2009; Eisenstein and Barzilay, 2008; Barzilay and Lee, 2004), typically modeled in terms of sequential topics. We avoid using them here, because we do not believe topic sequences are predictable in conversation and because such models tend to be algorithmically cumbersome.

In addition to text ordering, local coherence models have also been used to score the fluency of texts written by humans or produced by machine (Pitler and Nenkova, 2008; Lapata, 2006; Miltsakaki and Kukich, 2004). Like disentanglement, these tasks provide an algorithmic setting that differs from ordering, and so can demonstrate previously unknown weaknesses in models. However, the target genre is still informative writing, so they reveal little about cross-domain flexibility.

The task of disentanglement or "threading" for internet chat was introduced by Shen et al. (2006). Elsner and Charniak (2008b) created the publicly available #LINUX corpus; the best published results on this corpus are those of Wang and Oard (2009). These two studies use overlapping unigrams to measure similarity between two sentences; Wang and Oard (2009) use a message expansion technique to incorporate context beyond a single sentence. Unigram overlaps are used to model coherence, but more sophisticated methods using syntax (Lapata and Barzilay, 2005) or lexical features (Lapata, 2003) often outperform them on ordering tasks. This study compares several of these methods with Elsner and Charniak (2008b), which we use as a baseline because there is a publicly available implementation[2].

Adams (2008) also created and released a disentanglement corpus. They use LDA (Blei et al., 2001) to discover latent topics in their corpus, then measuring similarity by looking for shared topics. These features fail to improve their performance, which is puzzling in light of the success of topic modeling for other coherence and segmentation problems (Eisenstein and Barzilay, 2008; Foltz et al., 1998). The results of this study suggest that topic models can help with disentanglement, but that it is difficult to find useful topics for IRC chat.

A few studies have attempted to disentangle conversational speech (Aoki et al., 2003; Aoki et al., 2006), mostly using temporal features. For the most part, however, this research has focused on auditory processing in the context of the *cocktail party* problem, the task of attending to a specific speaker in a noisy room (Haykin and Chen, 2005). Utterance content has some influence on what the listener perceives, but only for extremely salient cues such as the listener's name (Moray, 1959), so cocktail party research does not typically use lexical models.

3 Models

In this section, we briefly describe the models we intend to evaluate. Most of them are drawn from previous work; one, the topical entity grid, is a novel extension of the entity grid. For the experiments below, we train the models on SWBD, sometimes augmented with a larger set of automatically parsed conversations from the FISHER corpus. Since the two corpora are quite similar, FISHER is a useful source for extra data; McClosky et al. (2010) uses it for this purpose in parsing experiments. (We continue to use SWBD/FISHER even for experiments on IRC, because we do not have enough disentangled training data to learn lexical relationships.)

3.1 Entity grid

The entity grid (Lapata and Barzilay, 2005; Barzilay and Lapata, 2005) is an attempt to model some principles of Centering Theory (Grosz et al., 1995) in a statistical manner. It represents a document in terms of entities and their syntactic roles: subject (S), object (O), other (X) and not present (-). In each new

[2] cs.brown.edu/~melsner

utterance, the grid predicts the role in which each entity will appear, given its history of roles in the previous sentences, plus a *salience* feature counting the total number of times the entity occurs. For instance, for an entity which is the subject of sentence 1, the object of sentence 2, and occurs four times in total, the grid predicts its role in sentence 3 according to the conditional $P(\cdot|S, O, sal = 4)$.

As in previous work, we treat each noun in a document as denoting a single entity, rather than using a coreference technique to attempt to resolve them. In our development experiments, we noticed that coreferent nouns often occur farther apart in conversation than in newswire, since they are frequently referred to by pronouns and deictics in the interim. Therefore, we extend the history to six previous utterances. For robustness with this long history, we model the conditional probabilities using multilabel logistic regression rather than maximum likelihood. This requires the assumption of a linear model, but makes the estimator less vulnerable to overfitting due to sparsity, increasing performance by about 2% in development experiments.

3.2 Topical entity grid

This model is a variant of the generative entity grid, intended to take into account topical information. To create the topical entity grid, we learn a set of topic-to-word distributions for our corpus using LDA (Blei et al., 2001)[3] with 200 latent topics. This model embeds our vocabulary in a low-dimensional space: we represent each word w as the vector of topic probabilities $p(t_i|w)$. We experimented with several ways to measure relationships between words in this space, starting with the standard cosine. However, the cosine can depend on small variations in probability (for instance, if w has most of its mass in dimension 1, then it is sensitive to the exact weight of v for topic 1, even if this essentially never happens).

To control for this tendency, we instead use the magnitude of the dimension of greatest similarity:

$$sim(w, v) = max_i \; min(w_i, v_i)$$

To model coherence, we generalize the binary history features of the standard entity grid, which detect, for example, whether entity e is the subject of the previous sentence. In the topical entity grid, we instead compute a real-valued feature which sums up the similarity between entity e and the subject(s) of the previous sentence.

These features can detect a transition like: "The House voted yesterday. The Senate will consider the bill today.". If "House" and "Senate" have a high similarity, then the feature will have a high value, predicting that "Senate" is a good subject for the current sentence. As in the previous section, we learn the conditional probabilities with logistic regression; we train in parallel by splitting the data and averaging (Mann et al., 2009). The topics are trained on FISHER, and on NANC for news.

3.3 IBM-1

The IBM translation model was first considered for coherence by Soricut and Marcu (2006), although a less probabilistically elegant version was proposed earlier (Lapata, 2003). This model attempts to generate the content words of the next sentence by translating them from the words of the previous sentence, plus a null word; thus, it will learn alignments between pairs of words that tend to occur in adjacent sentences. We learn parameters on the FISHER corpus, and on NANC for news.

3.4 Pronouns

The use of a generative pronoun resolver for coherence modeling originates in Elsner and Charniak (2008a). That paper used a supervised model (Ge et al., 1998), but we adapt a newer, unsupervised model which they also make publicly available (Charniak and Elsner, 2009)[4]. They model each pronoun as generated by an antecedent somewhere in the previous two sentences. If a good antecedent is found, the probability of the pronoun's occurrence will be high; otherwise, the probability is low, signaling that the text is less coherent because the pronoun is hard to interpret correctly.

We use the model as distributed for news text. For conversation, we adapt it by running a few iterations of their EM training algorithm on the FISHER data.

[3] www.cs.princeton.edu/~blei/topicmodeling.html

[4] bllip.cs.brown.edu/resources.shtml\#software

3.5 Discourse-newness

Building on work from summarization (Nenkova and McKeown, 2003) and coreference resolution (Poesio et al., 2005), Elsner and Charniak (2008a) use a model which recognizes discourse-new versus old NPs as a coherence model. For instance, the model can learn that "President Barack Obama" is a more likely first reference than "Obama". Following their work, we score discourse-newness with a maximum-entropy classifier using syntactic features counting different types of NP modifiers, and we use NP head identity as a proxy for coreference.

3.6 Chat-specific features

Most disentanglement models use non-linguistic information alongside lexical features; in fact, timestamps and speaker identities are usually *better* cues than words are. We capture three essential non-linguistic features using simple generative models.

The first feature is the **time** gap between one utterance and the next within the same thread. Consistent short gaps are a sign of normal turn-taking behavior; long pauses do occur, but much more rarely (Aoki et al., 2003). We round all time gaps to the nearest second and model the distribution of time gaps using a histogram, choosing bucket sizes adaptively so that each bucket contains at least four datapoints.

The second feature is **speaker** identity; conversations usually involve a small subset of the total number of speakers, and a few core speakers make most of the utterances. We model the distribution of speakers in each conversation using a Chinese Restaurant Process (CRP) (Aldous, 1985) (tuning the dispersion α to maximize development performance). The CRP's "rich-get-richer" dynamics capture our intuitions, favoring conversations dominated by a few vociferous speakers.

Finally, we model name **mentioning**. Speakers in IRC chat often use their addressee's names to coordinate the chat (O'Neill and Martin, 2003), and this is a powerful source of information (Elsner and Charniak, 2008b). Our model classifies each utterance into either the start or continuation of a conversational turn, by checking if the previous utterance had the same speaker. Given this status, it computes probabilities for three outcomes: no name mention, a mention of someone who has previously spoken in the conversation, or a mention of someone else. (The third option is extremely rare; this accounts for most of the model's predictive power). We learn these probabilities from IRC training data.

3.7 Model combination

To combine these different models, we adopt the log-linear framework of Soricut and Marcu (2006). Here, each model P_i is assigned a weight λ_i, and the combined score $P(d)$ is proportional to:

$$\sum_i \lambda_i log(P_i(d))$$

The weights λ can be learned discriminatively, maximizing the probability of d relative to a task-specific contrast set. For ordering experiments, the contrast set is a single random permutation of d; we explain the training regime for disentanglement below, in subsection 4.1.

4 Comparing orderings of SWBD

To measure the differences in performance caused by moving from news to a conversational domain, we first compare our models on an ordering task, discrimination (Barzilay and Lapata, 2005; Karamanis et al., 2004). In this task, we take an original document and randomly permute its sentences, creating an artificial incoherent document. We then test to see if our model prefers the coherent original.

For SWBD, rather than compare permutations of the individual utterances, we permute conversational turns (sets of consecutive utterances by each speaker), since turns are natural discourse units in conversation. We take documents numbered 2000–3999 as training/development and the remainder as test, yielding 505 training and 153 test documents; we evaluate 20 permutations per document. As a comparison, we also show results for the same models on WSJ, using the train-test split from Elsner and Charniak (2008a); the test set is sections 14-24, totalling 1004 documents.

Purandare and Litman (2008) carry out similar experiments on distinguishing permuted SWBD documents, using lexical and WordNet features in a model similar to Lapata (2003). Their accuracy for this task (which they call "switch-hard") is roughly 68%.

	WSJ	SWBD
EGrid	76.4‡	86.0
Topical EGrid	71.8‡	70.9‡
IBM-1	77.2‡	84.9†
Pronouns	69.6‡	71.7‡
Disc-new	72.3‡	55.0‡
Combined	81.9	88.4
-EGrid	81.0	87.5
-Topical EGrid	**82.2**	**90.5**
-IBM-1	79.0‡	88.9
-Pronouns	81.3	88.5
-Disc-new	**82.2**	88.4

Table 1: Discrimination F scores on news and dialogue. ‡ indicates a significant difference from the combined model at p=.01 and † at p=.05.

In Table 1, we show the results for individual models, for the combined model, and ablation results for mixtures without each component. WSJ is more difficult than SWBD overall because, on average, news articles are shorter than SWBD conversations. Short documents are harder, because permuting disrupts them less. The best SWBD result is 91%; the best WSJ result is 82% (both for mixtures without the topical entity grid). The WSJ result is state-of-the-art for the dataset, improving slightly on Elsner and Charniak (2008a) at 81%. We test results for significance using the non-parametric Mann-Whitney U test.

Controlling for the fact that discrimination is easier on SWBD, most of the individual models perform similarly in both corpora. The strongest models in both cases are the entity grid and IBM-1 (at about 77% for news, 85% for dialogue). Pronouns and the topical entity grid are weaker. The major outlier is the discourse-new model, whose performance drops from 72% for news to only 55%, just above chance, for conversation.

The model combination results show that all the models are quite closely correlated, since leaving out any single model does not degrade the combination very much (only one of the ablations is significantly worse than the combination). The most critical in news is IBM-1 (decreasing performance by 3% when removed); in conversation, it is the entity grid (decreasing by about 1%). The topical entity grid actually has a (nonsignificant) *negative* impact on combined performance, implying that its predictive power in this setting comes mainly from information that other models also capture, but that it is noisier and less reliable. In each domain, the combined models outperform the best single model, showing the information provided by the weaker models is not completely redundant.

Overall, these results suggest that most previously proposed local coherence models are domain-general; they work on conversation as well as news. The exception is the discourse-newness model, which benefits most from the specific conventions of a written style. Full names with titles (like "President Barack Obama") are more common in news, while conversation tends to involve fewer completely unfamiliar entities and more cases of bridging reference, in which grounding information is given implicitly (Nissim, 2006). Due to its poor performance, we omit the discourse-newness model in our remaining experiments.

5 Disentangling SWBD

We now turn to the task of disentanglement, testing whether models that are good at ordering also do well in this new setting. We would like to hold the domain constant, but we do not have any disentanglement data recorded from naturally occurring speech, so we create synthetic instances by merging pairs of SWBD dialogues. Doing so creates an artificial transcript in which two pairs of people appear to be talking simultaneously over a shared channel.

The situation is somewhat contrived in that each pair of speakers converses only with each other, never breaking into the other pair's dialogue and rarely using devices like name mentioning to make it clear who they are addressing. Since this makes speaker identity a perfect cue for disentanglement, we do not use it in this section. The only chat-specific model we use is **time**.

Because we are not using speaker information, we remove all utterances which do not contain a noun before constructing synthetic transcripts– these are mostly backchannels like "Yeah". Such utterances cannot be correctly assigned by our coherence models, which deal with content; we suspect most of them could be dealt with by associating them with the nearest utterance from the same speaker.

Once the backchannels are stripped, we can create a synthetic transcript. For each dialogue, we first simulate timestamps by sampling the number of seconds between each utterance and the next from a discretized Gaussian: $\lfloor N(0, 2.5) \rfloor$. The interleaving of the conversations is dictated by the timestamps. We truncate the longer conversation at the length of the shorter; this ensures a baseline score of 50% for the degenerate model that assigns all utterances to the same conversation.

We create synthetic instances of two types– those where the two entangled conversations had different topical prompts and those where they were the same. (Each dialogue in SWBD focuses on a preselected topic, such as *fishing* or *movies*.) We entangle dialogues from our ordering development set to use for mixture training and validation; for testing, we use 100 instances of each type, constructed from dialogues in our test set.

When disentangling, we treat each thread as independent of the others. In other words, the probability of the entire transcript is the product of the probabilities of the component threads. Our objective is to find the set of threads maximizing this. As a comparison, we use the model of Elsner and Charniak (2008b) as a baseline. To make their implementation comparable to ours, in this section we constrain it to find only two threads.

5.1 Disentangling a single utterance

Our first disentanglement task is to correctly assign a single utterance, given the true structure of the rest of the transcript. For each utterance, we compare two versions of the transcript, the original, and a version where it is swapped into the other thread. Our accuracy measures how often our models prefer the original. Unlike full-scale disentanglement, this task does not require a computationally demanding search, so it is possible to run experiments quickly. We also use it to train our mixture models for disentanglement, by construct a training example for each utterance i in our training transcripts. Since the Elsner and Charniak (2008b) model maximizes a correlation clustering objective which sums up independent edge weights, we can also use it to disentangle a single sentence efficiently.

Our results are shown in Table 2. Again, results for individual models are above the line, then our combined model, and finally ablation results for mixtures omitting a single model. The results show that, for a pair of dialogues that differ in topic, our best model can assign a single sentence with 87% accuracy. For the same topic, the accuracy is 80%. In each case, these results improve on (Elsner and Charniak, 2008b), which scores 78% and 74%.

	Different	Same	Avg.
EGrid	80.2	72.9	76.6
Topical EGrid	81.7	73.3	77.5
IBM-1	70.4	66.7	68.5
Pronouns	53.1	50.1	51.6
Time	58.5	57.4	57.9
Combined	**86.8**	**79.6**	**83.2**
-EGrid	86.0	79.1	82.6
-Topical EGrid	85.2	78.7	81.9
-IBM-1	86.2	78.7	82.4
-Pronouns	**86.8**	79.4	83.1
-Time	84.5	76.7	80.6
E+C '08	78.2	73.5	75.8

Table 2: Average accuracy for disentanglement of a single utterance on 200 synthetic multiparty conversations from SWBD test.

Changing to this new task has a substantial impact on performance. The topical model, which performed poorly for ordering, is actually stronger than the entity grid in this setting. IBM-1 underperforms either grid model (69% to 77%); on ordering, it was nearly as good (85% to 86%).

Despite their ordering performance of 72%, pronouns are essentially useless for this task, at 52%. This decline is due partly to domain, and partly to task setting. Although SWBD contains more pronominals than WSJ, many of them are first and second-person pronouns or deictics, which our model does not attempt to resolve. Since the disentanglement task involves moving only a single sentence, if moving this sentence does not sever a resolvable pronoun from its antecedent, the model will be unable to make a good decision.

As before, the ablation results show that all the models are quite correlated, since removing any single model from the mixture causes only a small decrease in performance. The largest drop (83% to 81%) is caused by removing time; though time is a weak model on its own, it is completely orthogo-

nal to the other models, since unlike them, it does not depend on the words in the sentences.

Comparing results between "different topic" and "same topic" instances shows that "same topic" is harder– by about 7% for the combined model. The IBM model has a relatively small gap of 3.7%, and in the ablation results, removing it causes a larger drop in performance for "same" than "different"; this suggests it is somewhat more robust to similarity in topic than entity grids.

Disentanglement accuracy is hard to predict given ordering performance; the two tasks plainly make different demands on models. One difference is that the models which use longer histories (the two entity grids) remain strong, while the models considering only one or two previous sentences (IBM and pronouns) do not do as well. Since the changes being considered here affect only a single sentence, while permutation affects the entire transcript, more history may help by making the model more sensitive to small changes.

5.2 Disentangling an entire transcript

We now turn to the task of disentangling an entire transcript at once. This is a practical task, motivated by applications such as search and information retrieval. However, it is more difficult than assigning only a single utterance, because decisions are interrelated– an error on one utterance may cause a cascade of poor decisions further down. It is also computationally harder.

We use tabu search (Glover and Laguna, 1997) to find a good solution. The search repeatedly finds and moves the utterance which would most improve the model score if swapped from one thread to the other. Unlike greedy search, tabu search is constrained not to repeat a solution that it has recently visited; this forces it to keep exploring when it reaches a local maximum. We run 500 iterations of tabu search (usually finding the first local maximum after about 100) and return the best solution found.

We measure performance with one-to-one overlap, which maps the two clusters to the two gold dialogues, then measures percent correct[5]. Our results (Table 3) show that, for transcripts with different topics, our disentanglement has 68% overlap with truth, extracting about two thirds of the structure correctly; this is substantially better than Elsner and Charniak (2008b), which scores 59%. Where the entangled conversations have the same topic, performance is lower, about 60%, but still better than the comparison model with 57%. Since correlations with the previous section are fairly reliable, and the disentanglement procedure is computationally intensive, we omit ablation experiments.

	Different	Same	Avg.
EGrid	60.3	57.1	58.7
Topical EGrid	62.3	56.8	59.6
IBM-1	56.5	55.2	55.9
Pronouns	54.5	54.4	54.4
Time	55.4	53.8	54.6
Combined	**67.9**	**59.8**	**63.9**
E+C '08	59.1	57.4	58.3

Table 3: One-to-one overlap between disentanglement results and truth on 200 synthetic multiparty conversations from SWBD test.

As we expect, full disentanglement is more difficult than single-sentence disentanglement (combined scores drop by about 20%), but the single-sentence task is a good predictor of relative performance. Entity grid models do best, the IBM model remains useful, but less so than for discrimination, and pronouns are very weak. The IBM model performs similarly under both metrics (56% and 57%), while other models perform worse on loc_3. This supports our suggestion that IBM's decline in performance from ordering is indeed due to its using a single sentence history; it is still capable of getting local structures right, but misses global ones.

6 IRC data

In this section, we move from synthetic data to real multiparty discourse recorded from internet chat rooms. We use two datasets: the #LINUX corpus (Elsner and Charniak, 2008b), and three larger corpora, #IPHONE, #PHYSICS and #PYTHON (Adams, 2008). We use the 1000-line "development" section of #LINUX for tuning our mixture models and the 800-line "test" section for development experiments. We reserve the Adams (2008) corpora for testing; together, they consist of 19581 lines of chat, with each section containing 500 to 1000 lines.

[5]The other popular metrics, F and loc_3, are correlated.

Chat-specific	74.0
+EGrid	**79.3**
+Topical EGrid	76.8
+IBM-1	76.3
+Pronouns	73.9
+EGrid/Topic/IBM-1	78.3
E+C '08b	76.4

Table 4: Accuracy for single utterance disentanglement, averaged over annotations of 800 lines of #LINUX data.

In order to use syntactic models like the entity grid, we parse the transcripts using (McClosky et al., 2006). Performance is bad, although the parser does identify most of the NPs; poor results are typical for a standard parser on chat (Foster, 2010). We postprocess the parse trees to retag "lol", "haha" and "yes" as UH (rather than NN, NNP and JJ).

In this section, we use all three of our chat-specific models (sec. 2.0.6; **time**, **speaker** and **mention**) as a baseline. This baseline is relatively strong, so we evaluate our other models in combination with it.

6.1 Disentangling a single sentence

As before, we show results on correctly disentangling a single sentence, given the correct structure of the rest of the transcript. We average performance on each transcript over the different annotations, then average the transcripts, weighing them by length to give each utterance equal weight.

Table 4 gives results on our development corpus, #LINUX. Our best result, for the chat-specific features plus entity grid, is 79%, improving on the comparison model, Elsner and Charniak (2008b), which gets 76%. (Although the table only presents an average over all annotations of the dataset, this model is also more accurate for each individual annotator than the comparison model.) We then ran the same model, chat-specific features plus entity grid, on the test corpora from Adams (2008). These results (Table 5) are also better than Elsner and Charniak (2008b), at an average of 93% over 89%.

As pointed out in Elsner and Charniak (2008b), the chat-specific features are quite powerful in this domain, and it is hard to improve over them. Elsner and Charniak (2008b), which has simple lexical features, mostly based on unigram overlap, increases

	#IPHONE	#PHYSICS	#PYTHON
+EGrid	**92.3**	**96.6**	**91.1**
E+C '08b	89.0	90.2	88.4

Table 5: Average accuracy for disentanglement of a single utterance for 19581 total lines from Adams (2008).

performance over baseline by 2%. Both IBM and the topical entity grid achieve similar gains. The entity grid does better, increasing performance to 79%. Pronouns, as before for SWBD, are useless.

We believe that the entity grid's good performance here is due mostly to two factors: its use of a long history, and its lack of lexicalization. The grid looks at the previous six sentences, which differentiates it from the IBM model and from Elsner and Charniak (2008b), which treats each pair of sentences independently. Using this long history helps to distinguish important nouns from unimportant ones better than frequency alone. We suspect that our lexicalized models, IBM and the topical entity grid, are hampered by poor parameter settings, since their parameters were learned on FISHER rather than IRC chat. In particular, we believe this explains why the topical entity grid, which slightly outperformed the entity grid on SWBD, is much worse here.

6.2 Full disentanglement

Running our tabu search algorithm on the full disentanglement task yields disappointing results. Accuracies on the #LINUX dataset are not only worse than previous work, but also worse than simple baselines like creating one thread for each speaker. The model finds far too many threads– it detects over 300, when the true number is about 81 (averaging over annotations). This appears to be related to biases in our chat-specific models as well as in the entity grid; the time model (which generates gaps between adjacent sentences) and the speaker model (which uses a CRP) both assign probability 1 to single-utterance conversations. The entity grid also has a bias toward short conversations, because unseen entities are empirically more likely to occur toward the beginning of a conversation than in the middle.

A major weakness in our model is that we aim only to maximize coherence of the individual conversations, with no prior on the likely length or number of conversations that will appear in the tran-

script. This allows the model to create far too many conversations. Integrating a prior into our framework is not straightforward because we currently train our mixture to maximize single-utterance disentanglement performance, and the prior is not useful for this task.

We experimented with fixing parts of the transcript to the solution obtained by Elsner and Charniak (2008b), then using tabu search to fill in the gaps. This constrains the number of conversations and their approximate positions. With this structure in place, we were able to obtain scores comparable to Elsner and Charniak (2008b), but not improvements. It appears that our performance increase on single-sentence disentanglement does not transfer to this task because of cascading errors and the necessity of using external constraints.

7 Conclusions

We demonstrate that several popular models of local coherence transfer well to the conversational domain, suggesting that they do indeed capture coherence in general rather than specific conventions of newswire text. However, their performance across tasks is not as stable; in particular, models which use less history information are worse for disentanglement.

Our results study suggest that while sophisticated coherence models can potentially contribute to disentanglement, they would benefit greatly from improved low-level resources for internet chat. Better parsing, or at least NP chunking, would help for models like the entity grid which rely on syntactic role information. Larger training sets, or some kind of transfer learning, could improve the learning of topics and other lexical parameters. In particular, our results on SWBD data confirm the conjecture of (Adams, 2008) that LDA topic modeling is in principle a useful tool for disentanglement– we believe a topic-based model could also work on IRC chat, but would require a better set of extracted topics. With better parameters for these models and the integration of a prior, we believe that our good performance on SWBD and single-utterance disentanglement for IRC can be extended to full-scale disentanglement of IRC.

Acknowledgements

We are extremely grateful to Regina Barzilay, Mark Johnson, Rebecca Mason, Ben Swanson and Neal Fox for their comments, to Craig Martell for the NPS chat datasets and to three anonymous reviewers. This work was funded by a Google Fellowship for Natural Language Processing.

References

Paige H. Adams. 2008. *Conversation Thread Extraction and Topic Detection in Text-based Chat.* Ph.D. thesis, Naval Postgraduate School.

David Aldous. 1985. Exchangeability and related topics. In *Ecole d'Ete de Probabilities de Saint-Flour XIII 1983*, pages 1–198. Springer.

Paul M. Aoki, Matthew Romaine, Margaret H. Szymanski, James D. Thornton, Daniel Wilson, and Allison Woodruff. 2003. The mad hatter's cocktail party: a social mobile audio space supporting multiple simultaneous conversations. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 425–432, New York, NY, USA. ACM Press.

Paul M. Aoki, Margaret H. Szymanski, Luke D. Plurkowski, James D. Thornton, Allison Woodruff, and Weilie Yi. 2006. Where's the "party" in "multiparty"?: analyzing the structure of small-group sociable talk. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 393–402, New York, NY, USA. ACM Press.

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: an entity-based approach. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL 2004: Proceedings of the Main Conference*, pages 113–120.

David Blei, Andrew Y. Ng, and Michael I. Jordan. 2001. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:2003.

Eugene Charniak and Micha Elsner. 2009. EM works for pronoun anaphora resolution. In *Proceedings of EACL*, Athens, Greece.

Harr Chen, S.R.K. Branavan, Regina Barzilay, and David R. Karger. 2009. Global models of document structure using latent permutations. In *Proceedings of Human Language Technologies: The 2009 Annual*

Conference of the North American Chapter of the Association for Computational Linguistics, pages 371–379, Boulder, Colorado, June. Association for Computational Linguistics.

Jacob Eisenstein and Regina Barzilay. 2008. Bayesian unsupervised topic segmentation. In *EMNLP*, pages 334–343.

Micha Elsner and Eugene Charniak. 2008a. Coreference-inspired coherence modeling. In *Proceedings of ACL-08: HLT, Short Papers*, pages 41–44, Columbus, Ohio, June. Association for Computational Linguistics.

Micha Elsner and Eugene Charniak. 2008b. You talking to me? a corpus and algorithm for conversation disentanglement. In *Proceedings of ACL-08: HLT*, pages 834–842, Columbus, Ohio, June. Association for Computational Linguistics.

Peter Foltz, Walter Kintsch, and Thomas Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse Processes*, 25(2&3):285–307.

Jennifer Foster. 2010. "cba to check the spelling": Investigating parser performance on discussion forum posts. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 381–384, Los Angeles, California, June. Association for Computational Linguistics.

Niyu Ge, John Hale, and Eugene Charniak. 1998. A statistical approach to anaphora resolution. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 161–171, Orlando, Florida. Harcourt Brace.

Fred Glover and Manuel Laguna. 1997. *Tabu Search.* University of Colorado at Boulder.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Simon Haykin and Zhe Chen. 2005. The Cocktail Party Problem. *Neural Computation*, 17(9):1875–1902.

Nikiforos Karamanis, Massimo Poesio, Chris Mellish, and Jon Oberlander. 2004. Evaluating centering-based metrics of coherence. In *ACL*, pages 391–398.

Mirella Lapata and Regina Barzilay. 2005. Automatic evaluation of text coherence: Models and representations. In *IJCAI*, pages 1085–1090.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the annual meeting of ACL, 2003.*

Mirella Lapata. 2006. Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics*, 32(4):1–14.

Gideon Mann, Ryan McDonald, Mehryar Mohri, Nathan Silberman, and Dan Walker. 2009. Efficient large-scale distributed training of conditional maximum entropy models. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1231–1239.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 28–36, Los Angeles, California, June. Association for Computational Linguistics.

Eleni Miltsakaki and K. Kukich. 2004. Evaluation of text coherence for electronic essay scoring systems. *Nat. Lang. Eng.*, 10(1):25–55.

Neville Moray. 1959. Attention in dichotic listening: Affective cues and the influence of instructions. *Quarterly Journal of Experimental Psychology*, 11(1):56–60.

Ani Nenkova and Kathleen McKeown. 2003. References to named entities: a corpus study. In *NAACL '03*, pages 70–72.

Malvina Nissim. 2006. Learning information status of discourse entities. In *Proceedings of EMNLP*, pages 94–102, Morristown, NJ, USA. Association for Computational Linguistics.

Jacki O'Neill and David Martin. 2003. Text chat in action. In *GROUP '03: Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*, pages 40–49, New York, NY, USA. ACM Press.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 186–195, Honolulu, Hawaii, October. Association for Computational Linguistics.

Massimo Poesio, Mijail Alexandrov-Kabadjov, Renata Vieira, Rodrigo Goulart, and Olga Uryupina. 2005. Does discourse-new detection help definite description resolution? In *Proceedings of the Sixth International Workshop on Computational Semantics*, Tillburg.

Amruta Purandare and Diane J. Litman. 2008. Analyzing dialog coherence using transition patterns in lexical and semantic features. In *FLAIRS Conference'08*, pages 195–200.

Dou Shen, Qiang Yang, Jian-Tao Sun, and Zheng Chen. 2006. Thread detection in dynamic text message

streams. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 35–42, New York, NY, USA. ACM.

Radu Soricut and Daniel Marcu. 2006. Discourse generation using utility-trained coherence models. In *Proceedings of the Association for Computational Linguistics Conference (ACL-2006)*.

Lidan Wang and Douglas W. Oard. 2009. Context-based message expansion for disentanglement of interleaved text conversations. In *Proceedings of NAACL-09*.

An Affect-Enriched Dialogue Act Classification Model for Task-Oriented Dialogue

Kristy Elizabeth Boyer | **Joseph F. Grafsgaard** | **Eun Young Ha** | **Robert Phillips*** | **James C. Lester**

Department of Computer Science
North Carolina State University
Raleigh, NC, USA

* Dual Affiliation with Applied Research Associates, Inc.
Raleigh, NC, USA

{keboyer, jfgrafsg, eha, rphilli, lester}@ncsu.edu

Abstract

Dialogue act classification is a central challenge for dialogue systems. Although the importance of emotion in human dialogue is widely recognized, most dialogue act classification models make limited or no use of affective channels in dialogue act classification. This paper presents a novel affect-enriched dialogue act classifier for task-oriented dialogue that models facial expressions of users, in particular, facial expressions related to confusion. The findings indicate that the affect-enriched classifiers perform significantly better for distinguishing user requests for feedback and grounding dialogue acts within textual dialogue. The results point to ways in which dialogue systems can effectively leverage affective channels to improve dialogue act classification.

1 Introduction

Dialogue systems aim to engage users in rich, adaptive natural language conversation. For these systems, understanding the role of a user's utterance in the broader context of the dialogue is a key challenge (Sridhar, Bangalore, & Narayanan, 2009). Central to this endeavor is dialogue act classification, which categorizes the intention behind the user's move (e.g., asking a question, providing declarative information). Automatic dialogue act classification has been the focus of a large body of research, and a variety of approaches, including sequential models (Stolcke et al., 2000), vector-based models (Sridhar, Bangalore, & Narayanan, 2009), and most recently, feature-enhanced latent semantic analysis (Di Eugenio, Xie, & Serafin, 2010), have shown promise. These models may be further improved by leveraging regularities of the dialogue from both linguistic and extra-linguistic sources. Users' expressions of emotion are one such source.

Human interaction has long been understood to include rich phenomena consisting of verbal and nonverbal cues, with facial expressions playing a vital role (Knapp & Hall, 2006; McNeill, 1992; Mehrabian, 2007; Russell, Bachorowski, & Fernandez-Dols, 2003; Schmidt & Cohn, 2001). While the importance of emotional expressions in dialogue is widely recognized, the majority of dialogue act classification projects have focused either peripherally (or not at all) on emotion, such as by leveraging acoustic and prosodic features of spoken utterances to aid in online dialogue act classification (Sridhar, Bangalore, & Narayanan, 2009). Other research on emotion in dialogue has involved detecting affect and adapting to it within a dialogue system (Forbes-Riley, Rotaru, Litman, & Tetreault, 2009; López-Cózar, Silovsky, & Griol, 2010), but this work has not explored leveraging affect information for automatic user dialogue act classification. Outside of dialogue, sentiment analysis within discourse is an active area of research (López-Cózar et al., 2010), but it is generally lim-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1190–1199,
Portland, Oregon, June 19-24, 2011.

ited to modeling textual features and not multimodal expressions of emotion such as facial actions. Such multimodal expressions have only just begun to be explored within corpus-based dialogue research (Calvo & D'Mello, 2010; Cavicchio, 2009).

This paper presents a novel affect-enriched dialogue act classification approach that leverages knowledge of users' facial expressions during computer-mediated textual human-human dialogue. Intuitively, the user's affective state is a promising source of information that may help to distinguish between particular dialogue acts (e.g., a *confused* user may be more likely to ask a question). We focus specifically on occurrences of students' confusion-related facial actions during task-oriented tutorial dialogue.

Confusion was selected as the focus of this work for several reasons. First, confusion is known to be prevalent within tutoring, and its implications for student learning are thought to run deep (Graesser, Lu, Olde, Cooper-Pye, & Whitten, 2005). Second, while identifying the "ground truth" of emotion based on any external display by a user presents challenges, prior research has demonstrated a correlation between particular facial action units and confusion during learning (Craig, D'Mello, Witherspoon, Sullins, & Graesser, 2004; D'Mello, Craig, Sullins, & Graesser, 2006; McDaniel et al., 2007). Finally, automatic facial action recognition technologies are developing rapidly, and confusion-related facial action events are among those that can be reliably recognized automatically (Bartlett et al., 2006; Cohn, Reed, Ambadar, Xiao, & Moriyama, 2004; Pantic & Bartlett, 2007; Zeng, Pantic, Roisman, & Huang, 2009). This promising development bodes well for the feasibility of automatic real-time confusion detection within dialogue systems.

2 Background and Related Work

2.1 Dialogue Act Classification

Because of the importance of dialogue act classification within dialogue systems, it has been an active area of research for some time. Early work on automatic dialogue act classification modeled discourse structure with hidden Markov models, experimenting with lexical and prosodic features, and applying the dialogue act model as a constraint to aid in automatic speech recognition (Stolcke et al., 2000). In contrast to this sequential modeling approach, which is best suited to offline processing, recent work has explored how lexical, syntactic, and prosodic features perform for online dialogue act tagging (when only partial dialogue sequences are available) within a maximum entropy framework (Sridhar, Bangalore, & Narayanan, 2009). A recently proposed alternative approach involves treating dialogue utterances as documents within a latent semantic analysis framework, and applying feature enhancements that incorporate such information as speaker and utterance duration (Di Eugenio et al., 2010). Of the approaches noted above, the modeling framework presented in this paper is most similar to the vector-based maximum entropy approach of Sridhar et al. (2009). However, it takes a step beyond the previous work by including multimodal affective displays, specifically facial expressions, as features available to an affect-enriched dialogue act classification model.

2.2 Detecting Emotions in Dialogue

Detecting emotional states during spoken dialogue is an active area of research, much of which focuses on detecting frustration so that a user can be automatically transferred to a human dialogue agent (López-Cózar et al., 2010). Research on spoken dialogue has leveraged lexical features along with discourse cues and acoustic information to classify user emotion, sometimes at a coarse grain along a positive/negative axis (Lee & Narayanan, 2005). Recent work on an affective companion agent has examined user emotion classification within conversational speech (Cavazza et al., 2010). In contrast to that spoken dialogue research, the work in this paper is situated within textual dialogue, a widely used modality of communication for which a deeper understanding of user affect may substantially improve system performance.

While many projects have focused on linguistic cues, recent work has begun to explore numerous channels for affect detection including facial actions, electrocardiograms, skin conductance, and posture sensors (Calvo & D'Mello, 2010). A recent project in a map task domain investigates some of these sources of affect data within task-oriented dialogue (Cavicchio, 2009). Like that work, the current project utilizes facial action tagging, for

which promising automatic technologies exist (Bartlett et al., 2006; Pantic & Bartlett, 2007; Zeng, Pantic, Roisman, & Huang, 2009). However, we leverage the recognized expressions of emotion for the task of dialogue act classification.

2.3 Categorizing Emotions within Dialogue and Discourse

Sets of emotion taxonomies for discourse and dialogue are often application-specific, for example, focusing on the frustration of users who are interacting with a spoken dialogue system (López-Cózar et al., 2010), or on uncertainty expressed by students while interacting with a tutor (Forbes-Riley, Rotaru, Litman, & Tetreault, 2007). In contrast, the most widely utilized emotion frameworks are not application-specific; for example, Ekman's Facial Action Coding System (FACS) has been widely used as a rigorous technique for coding facial movements based on human facial anatomy (Ekman & Friesen, 1978). Within this framework, facial movements are categorized into facial action units, which represent discrete movements of muscle groups. Additionally, facial action descriptors (for movements not derived from facial muscles) and movement and visibility codes are included. Ekman's basic emotions (Ekman, 1999) have been used in recent work on classifying emotion expressed within blog text (Das & Bandyopadhyay, 2009), while other recent work (Nguyen, 2010) utilizes Russell's core affect model (Russell, 2003) for a similar task.

During tutorial dialogue, students may not frequently experience Ekman's basic emotions of *happiness*, *sadness*, *anger*, *fear*, *surprise*, and *disgust*. Instead, students appear to more frequently experience cognitive-affective states such as *flow* and *confusion* (Calvo & D'Mello, 2010). Our work leverages Ekman's facial tagging scheme to identify a particular facial action unit, Action Unit 4 (AU4), that has been observed to correlate with confusion (Craig, D'Mello, Witherspoon, Sullins, & Graesser, 2004; D'Mello, Craig, Sullins, & Graesser, 2006; McDaniel et al., 2007).

2.4 Importance of Confusion in Tutorial Dialogue

Among the affective states that students experience during tutorial dialogue, confusion is prevalent, and its implications for student learning are significant. Confusion is associated with *cognitive disequilibrium*, a state in which students' existing knowledge is inconsistent with a novel learning experience (Graesser, Lu, Olde, Cooper-Pye, & Whitten, 2005). Students may express such confusion within dialogue as *uncertainty*, to which human tutors often adapt in a context-dependent fashion (Forbes-Riley et al., 2007). Moreover, implementing adaptations to student uncertainty within a dialogue system can improve the effectiveness of the system (Forbes-Riley et al., 2009).

For tutorial dialogue, the importance of understanding student utterances is paramount for a system to positively impact student learning (Dzikovska, Moore, Steinhauser, & Campbell, 2010). The importance of frustration as a cognitive-affective state during learning suggests that the presence of student confusion may serve as a useful constraining feature for dialogue act classification of student utterances. This paper explores the use of facial expression features in this way.

3 Task-Oriented Dialogue Corpus

The corpus was collected during a textual human-human tutorial dialogue study in the domain of introductory computer science (Boyer, Phillips, et al., 2010). Students solved an introductory computer programming problem and carried on textual dialogue with tutors, who viewed a synchronized version of the students' problem-solving workspace. The original corpus consists of 48 dialogues, one per student. Each student interacted with one of two tutors. Facial videos of students were collected using built-in webcams, but were not shown to the tutors. Video quality was ranked based on factors such as obscured foreheads due to hats or hair, and improper camera position resulting in students' faces not being fully captured on the video. The highest-quality set contained 14 videos, and these videos were used in this analysis. They have a total running time of 11 hours and 55 minutes, and include dialogues with three female subjects and eleven male subjects.

3.1 Dialogue act annotation

The dialogue act annotation scheme (Table 1) was applied manually. The kappa statistic for inter-annotator agreement on a 10% subset of the corpus was κ=0.80, indicating good reliability.

Table 1. Dialogue act tags and relative frequencies across fourteen dialogues in video corpus

Student Dialogue Act	Example	Rel. Freq.
EXTRA-DOMAIN (EX)	*Little sleep deprived today*	.08
GROUNDING (G)	*Ok* or *Thanks*	.21
NEGATIVE FEEDBACK WITH ELABORATION (NE)	*I'm still confused on what this next for loop is doing.*	.02
NEGATIVE FEEDBACK (N)	*I don't see the diff.*	.04
POSITIVE FEEDBACK WITH ELABORATION (PE)	*It makes sense now that you explained it, but I never used an else if in any of my other programs*	.04
POSITIVE FEEDBACK (P)	*Second part complete.*	.11
QUESTION (Q)	*Why couldn't I have said if (i<5)*	.11
STATEMENT (S)	*i is my only index*	.07
REQUEST FOR FEEDBACK (RF)	*So I need to create a new method that sees how many elements are in my array?*	.16
RESPONSE (RSP)	*You mean not the length but the contents*	.14
UNCERTAIN FEEDBACK WITH ELABORATION (UE)	*I'm trying to remember how to copy arrays*	.008
UNCERTAIN FEEDBACK (U)	*Not quite yet*	.008

3.2 Task action annotation

The tutoring sessions were task-oriented, focusing on a computer programming exercise. The task had several subtasks consisting of programming modules to be implemented by the student. Each of those subtasks also had numerous fine-grained goals, and student task actions either contributed or did not contribute to the goals. Therefore, to obtain a rich representation of the task, a manual annotation along two dimensions was conducted (Boyer, Phillips, et al., 2010). First, the subtask structure was annotated hierarchically, and then each task action was labeled for correctness according to the requirements of the assignment. Inter-annotator agreement was computed on 20% of the corpus at the leaves of the subtask tagging scheme, and resulted in a simple kappa of κ=.56. However, the leaves of the annotation scheme feature an implicit ordering (subtasks were completed in order, and adjacent subtasks are semantically more similar than subtasks at a greater distance); therefore, a weighted kappa is also meaningful to consider for this annotation. The weighted kappa is $\kappa_{weighted}$=.80. An annotated excerpt of the corpus is displayed in Table 2.

Table 2. Excerpt from corpus illustrating annotations and interplay between dialogue and task

13:38:09	Student:	How do I know where to end? **[RF]**
13:38:26	Tutor:	Well you told me how to get how many elements in an array by using .length right?
13:38:26	Student:	**[Task action: Subtask 1-a-iv, Buggy]**
13:38:56	Tutor:	Great
13:38:56	Student:	**[Task action: Subtask 1-a-v, Correct]**
13:39:35	Student:	Well is it "array.length"? **[RF]** ****Facial Expression: AU4**
13:39:46	Tutor:	You just need to use the correct array name
13:39:46	Student:	**[Task action: Subtask 1-a-iv, Buggy]**

3.3 Lexical and Syntactic Features

In addition to the manually annotated dialogue and task features described above, syntactic features of each utterance were automatically extracted using the Stanford Parser (De Marneffe et al., 2006). From the phrase structure trees, we extracted the top-most syntactic node and its first two children. In the case where an utterance consisted of more than one sentence, only the phrase structure tree of the first sentence was considered. Individual word tokens in the utterances were further processed with the Porter Stemmer (Porter, 1980) in the NLTK package (Loper & Bird, 2004). Our prior work has shown that these lexical and syntactic features are highly predictive of dialogue acts during task-oriented tutorial dialogue (Boyer, Ha et al. 2010).

4 Facial Action Tagging

An annotator who was certified in the Facial Action Coding System (FACS) (Ekman, Friesen, & Hager, 2002) tagged the video corpus consisting of fourteen dialogues. The FACS certification process requires annotators to pass a test designed to analyze their agreement with reference coders on a set of spontaneous facial expressions (Ekman & Rosenberg, 2005). This annotator viewed the videos continuously and paused the playback whenever notable facial displays of Action Unit 4 (AU4: Brow Lowerer) were seen. This action unit was chosen for this study based on its correlations with confusion in prior research (Craig, D'Mello, Witherspoon, Sullins, & Graesser, 2004; D'Mello, Craig, Sullins, & Graesser, 2006; McDaniel et al., 2007).

To establish reliability of the annotation, a second FACS-certified annotator independently annotated 36% of the video corpus (5 of 14 dialogues), chosen randomly after stratification by gender and tutor. This annotator followed the same method as the first annotator, pausing the video at any point to tag facial action events. At any given time in the video, the coder was first identifying whether an action unit event existed, and then describing the facial movements that were present. The annotators also specified the beginning and ending time of each event. In this way, the action unit event tags spanned discrete durations of varying length, as specified by the coders. Because the two coders were not required to tag at the same point in time, but rather were permitted the freedom to stop the video at any point where they felt a notable facial action event occurred, calculating agreement between annotators required discretizing the continuous facial action time windows across the tutoring sessions. This discretization was performed at granularities of 1/4, 1/2, 3/4, and 1 second, and inter-rater reliability was calculated at each level of granularity (Table 3). Windows in which both annotators agreed that no facial action event was present were tagged by default as *neutral*. Figure 1 illustrates facial expressions that display facial Action Unit 4.

Table 3. Kappa values for inter-annotator agreement on facial action events

	Granularity			
	¼ sec	½ sec	¾ sec	1 sec
Presence of AU4 (Brow Lowerer)	*.84*	*.87*	*.86*	*.86*

Figure 1. Facial expressions displaying AU4 (Brow Lowerer)

Despite the fact that promising automatic approaches exist to identifying many facial action units (Bartlett et al., 2006; Cohn, Reed, Ambadar, Xiao, & Moriyama, 2004; Pantic & Bartlett, 2007; Zeng, Pantic, Roisman, & Huang, 2009), manual annotation was selected for this project for two reasons. First, manual annotation is more robust than automatic recognition of facial action units, and manual annotation facilitated an exploratory, comprehensive view of student facial expressions during learning through task-oriented dialogue. Although a detailed discussion of the other emotions present in the corpus is beyond the scope of this paper, Figure 2 illustrates some other spontaneous student facial expressions that differ from those associated with confusion.

Figure 2. Other facial expressions from the corpus

5 Models

The goal of the modeling experiment was to determine whether the addition of confusion-related facial expression features significantly boosts dialogue act classification accuracy for student utterances.

5.1 Features

We take a vector-based approach, in which the features consist of the following:

Utterance Features

- *Dialogue act features*: Manually annotated dialogue act for the past three utterances. These features include tutor dialogue acts, annotated with a scheme analogous to that used to annotate student utterances (Boyer et al., 2009).
- *Speaker*: Speaker for past three utterances
- *Lexical features*: Word unigrams
- *Syntactic features*: Top-most syntactic node and its first two children

Task-based Features

- *Subtask*: Hierarchical subtask structure for past three task actions (semantic programming actions taken by student)
- *Correctness*: Correctness of past three task actions taken by student
- *Preceded by task*: Indicator for whether the most recent task action immediately preceded the target utterance, or whether it was immediately preceded by the last dialogue move

Facial Expression Features

- *AU4_1sec:* Indicator for the display of the brow lowerer within 1 second prior to this utterance being sent, for the most recent three utterances
- *AU4_5sec:* Indicator for the display of the brow lowerer within 5 seconds prior to this utterance being sent, for the most recent three utterances
- *AU4_10sec:* Indicator for the display of the brow lowerer within 10 seconds prior to this utterance being sent, for the most recent three utterances

5.2 Modeling Approach

A logistic regression approach was used to classify the dialogue acts based on the above feature vectors. The Weka machine learning toolkit (Hall et al., 2009) was used to learn the models and to first perform feature selection in a best-first search. Logistic regression is a generalized maximum likelihood model that discriminates between pairs of output values by calculating a feature weight vector over the predictors.

The goal of this work is to explore the utility of confusion-related facial features in the context of particular dialogue act types. For this reason, a specialized classifier was learned by dialogue act.

5.3 Classification Results

The classification accuracy and kappa for each specialized classifier is displayed in Table 4. Note that kappa statistics adjust for the accuracy that would be expected by majority-baseline chance; a kappa statistic of zero indicates that the classifier performed equal to chance, and a positive kappa statistic indicates that the classifier performed better than chance. A kappa of 1 constitutes perfect agreement. As the table illustrates, the feature selection chose to utilize the AU4 feature for every dialogue act except STATEMENT (S). When considering the accuracy of the model across the ten folds, two of the affect-enriched classifiers exhibited statistically significantly better performance. For GROUNDING (G) and REQUEST FOR FEEDBACK (RF), the facial expression features significantly

improved the classification accuracy compared to a model that was learned without affective features.

6 Discussion

Dialogue act classification is an essential task for dialogue systems, and it has been addressed with a variety of modeling approaches and feature sets. We have presented a novel approach that treats facial expressions of students as constraining features for an affect-enriched dialogue act classification model in task-oriented tutorial dialogue. The results suggest that knowledge of the student's confusion-related facial expressions can significantly enhance dialogue act classification for two types of dialogue acts, GROUNDING and REQUEST FOR FEEDBACK.

Table 4. Classification accuracy and kappa for specialized DA classifiers. Statistically significant differences (across ten folds, one-tailed *t*-test) are shown in bold.

Dialogue Act	Classifier with AU4		Classifier without AU4		*p*-value
	% acc	κ	% acc	κ	
EX	90.7	.62	89.0	.28	>.05
G	**92.6**	**.76**	**91**	**.71**	**.018**
P	93	.49	92.2	.40	>.05
Q	94.6	.72	94.2	.72	>.05
S	Not chosen in feat. sel.		93	.22	n/a
RF	**90.7**	**.62**	**88.3**	**.53**	**.003**
RSP	93	.68	95	.75	>.05
NE	*		*		
N	*		*		
PE	*		*		
U	*		*		
UE	*		*		

*Too few instances for ten-fold cross-validation.

6.1 Features Selected for Classification

Out of more than 1500 features available during feature selection, each of the specialized dialogue act classifiers selected between 30 and 50 features in each condition (with and without affect features). To gain insight into the specific features that were useful for classifying these dialogue acts, it is useful to examine which of the AU4 history features were chosen during feature selection.

For GROUNDING, features that indicated the presence of absence of AU4 in the immediately preceding utterance, either at the 1 second or 5 second granularity, were selected. Absence of this confusion-related facial action unit was associated with a higher probability of a grounding act, such as an acknowledgement. This finding is consistent with our understanding of how students and tutors interacted in this corpus; when a student experienced confusion, she would be unlikely to then make a simple grounding dialogue move, but instead would tend to inspect her computer program, ask a question, or wait for the tutor to explain more.

For REQUEST FOR FEEDBACK, the predictive features were presence or absence of AU4 within ten seconds of the longest available history (three turns in the past), as well as the presence of AU4 within five seconds of the current utterance (the utterance whose dialogue act is being classified). This finding suggests that there may be some lag between the student experiencing confusion and then choosing to make a request for feedback, and that the confusion-related facial expressions may re-emerge as the student is making a request for feedback, since the five-second window prior to the student sending the textual dialogue message would overlap with the student's construction of the message itself.

Although the improvements seen with AU4 features for QUESTION, POSITIVE FEEDBACK, and EXTRA-DOMAIN acts were not statistically reliable, examining the AU4 features that were selected for classifying these moves points toward ways in which facial expressions may influence classification of these acts (Table 5).

Table 5. Number of features, and AU4 features selected, for specialized DA classifiers

Dialogue Act	# features selected	AU4 features selected
G	43	One utterance ago: AU4_1sec, AU4_5sec
RF	37	Three utterances ago: AU4_10sec Target utterance: AU4_5sec
EX	50	Three utterances ago: AU4_1sec
P	36	Current utterance: AU4_10sec
Q	30	One utterance ago: AU4_5sec

6.2 Implications

The results presented here demonstrate that leveraging knowledge of user affect, in particular of spontaneous facial expressions, may improve the performance of dialogue act classification models. Perhaps most interestingly, displays of confusion-related facial actions prior to a student dialogue move enabled an affect-enriched classifier to recognize requests for feedback with significantly greater accuracy than a classifier that did not have access to the facial action features. Feedback is known to be a key component of effective tutorial dialogue, through which tutors provide adaptive help (Shute, 2008). Requesting feedback also seems to be an important behavior of students, characteristically engaged in more frequently by women than men, and more frequently by students with lower incoming knowledge than by students with higher incoming knowledge (Boyer, Vouk, & Lester, 2007).

6.3 Limitations

The experiments reported here have several notable limitations. First, the time-consuming nature of manual facial action tagging restricted the number of dialogues that could be tagged. Although the highest quality videos were selected for annotation, other medium quality videos would have been sufficiently clear to permit tagging, which would have increased the sample size and likely revealed statistically significant trends. For example, the performance of the affect-enriched classifier was better for dialogue acts of interest such as positive feedback and questions, but this difference was not statistically reliable.

An additional limitation stems from the more fundamental question of which affective states are indicated by particular external displays. The field is only just beginning to understand facial expressions during learning and to correlate these facial actions with emotions. Additional research into the "ground truth" of emotion expression will shed additional light on this area. Finally, the results of manual facial action annotation may constitute upper-bound findings for applying automatic facial expression analysis to dialogue act classification.

7 Conclusions and Future Work

Emotion plays a vital role in human interactions. In particular, the role of facial expressions in human-human dialogue is widely recognized. Facial expressions offer a promising channel for understanding the emotions experienced by users of dialogue systems, particularly given the ubiquity of webcam technologies and the increasing number of dialogue systems that are deployed on webcam-enabled devices. This paper has reported on a first step toward using knowledge of user facial expressions to improve a dialogue act classification model for tutorial dialogue, and the results demonstrate that facial expressions hold great promise for distinguishing the pedagogically relevant dialogue act REQUEST FOR FEEDBACK, and the conversational moves of GROUNDING.

These early findings highlight the importance of future work in this area. Dialogue act classification models have not fully leveraged some of the techniques emerging from work on sentiment analysis. These approaches may prove particularly useful for identifying emotions in dialogue utterances. Another important direction for future work involves more fully exploring the ways in which affect expression differs between textual and spoken dialogue. Finally, as automatic facial tagging technologies mature, they may prove powerful enough to enable broadly deployed dialogue systems to feasibly leverage facial expression data in the near future.

Acknowledgments

This work is supported in part by the North Carolina State University Department of Computer Science and by the National Science Foundation through Grants REC-0632450, IIS-0812291, DRL-1007962 and the STARS Alliance Grant CNS-0739216. Any opinions, findings, conclusions, or recommendations expressed in this report are those of the participants, and do not necessarily represent the official views, opinions, or policy of the National Science Foundation.

References

A. Andreevskaia and S. Bergler. 2008. When specialists and generalists work together: Overcoming domain dependence in sentiment tagging. Proceedings of the Annual Meeting of the Association for Computational Linguistics and Human Language Technologies (ACL HLT), 290-298.

M.S. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, and J. Movellan. 2006. Fully Automatic Facial Action Recognition in Spontaneous Behavior. 7th International Conference on Automatic Face and Gesture Recognition (FGR06), 223-230.

K.E. Boyer, M. Vouk, and J.C. Lester. 2007. The influence of learner characteristics on task-oriented tutorial dialogue. Proceedings of the International Conference on Artificial Intelligence in Education, 365–372.

K.E. Boyer, E.Y. Ha, R. Phillips, M.D. Wallis, M. Vouk, and J.C. Lester. 2010. Dialogue act modeling in a complex task-oriented domain. Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), 297-305.

K.E. Boyer, R. Phillips, E.Y. Ha, M.D. Wallis, M.A. Vouk, and J.C. Lester. 2009. Modeling dialogue structure with adjacency pair analysis and hidden Markov models. Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Short Papers, 49-52.

K.E. Boyer, R. Phillips, E.Y. Ha, M.D. Wallis, M.A. Vouk, and J.C. Lester. 2010. Leveraging hidden dialogue state to select tutorial moves. Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications, 66-73.

R.A. Calvo and S. D'Mello. 2010. Affect Detection: An Interdisciplinary Review of Models, Methods, and Their Applications. IEEE Transactions on Affective Computing, 1(1): 18-37.

M. Cavazza, R.S.D.L. Cámara, M. Turunen, J. Gil, J. Hakulinen, N. Crook, et al. 2010. How was your day? An affective companion ECA prototype. Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), 277-280.

F. Cavicchio. 2009. The modulation of cooperation and emotion in dialogue: the REC Corpus. Proceedings of the ACL-IJCNLP 2009 Student Research Workshop, 43-48.

J.F. Cohn, L.I. Reed, Z. Ambadar, J. Xiao, and T. Moriyama. 2004. Automatic Analysis and Recognition of Brow Actions and Head Motion in Spontaneous Facial Behavior. IEEE International Conference on Systems, Man and Cybernetics, 610-616.

S.D. Craig, S. D'Mello, A. Witherspoon, J. Sullins, and A.C. Graesser. 2004. Emotions during learning: The first steps toward an affect sensitive intelligent tutoring system. In J. Nall and R. Robson (Eds.), E-learn 2004: World conference on E-learning in Corporate, Government, Healthcare, & Higher Education, 241-250.

D. Das and S. Bandyopadhyay. 2009. Word to sentence level emotion tagging for Bengali blogs. Proceedings of the ACL-IJCNLP Conference, Short Papers, 149-152.

S. Dasgupta and V. Ng. 2009. Mine the easy, classify the hard: a semi-supervised approach to automatic sentiment classification. Proceedings of the 46th Annual Meeting of the ACL and the 4th IJCNLP, 701-709.

B. Di Eugenio, Z. Xie, and R. Serafin. 2010. Dialogue Act Classification, Higher Order Dialogue Structure, and Instance-Based Learning. Dialogue & Discourse, 1(2): 1-24.

M. Dzikovska, J.D. Moore, N. Steinhauser, and G. Campbell. 2010. The impact of interpretation problems on tutorial dialogue. Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Short Papers, 43-48.

S. D'Mello, S.D. Craig, J. Sullins, and A.C. Graesser. 2006. Predicting Affective States expressed through an Emote-Aloud Procedure from AutoTutor's Mixed- Initiative Dialogue. International Journal of Artificial Intelligence in Education, 16(1): 3-28.

P. Ekman. 1999. Basic Emotions. In T. Dalgleish and M. J. Power (Eds.), Handbook of Cognition and Emotion. New York: Wiley.

P. Ekman, W.V. Friesen. 1978. Facial Action Coding System. Palo Alto, CA: Consulting Psychologists Press.

P. Ekman, W.V. Friesen, and J.C. Hager. 2002. Facial Action Coding System: Investigator's Guide. Salt Lake City, USA: A Human Face.

P. Ekman and E.L. Rosenberg (Eds.). 2005. What the Face Reveals: Basic and Applied Studies of Spontaneous Expression Using the Facial Action Coding System (FACS) (2nd ed.). New York: Oxford University Press.

K. Forbes-Riley, M. Rotaru, D.J. Litman, and J. Tetreault. 2007. Exploring affect-context dependencies for adaptive system development. The Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technologies (NAACL HLT), Short Papers, 41-44.

K. Forbes-Riley, M. Rotaru, D.J. Litman, and J. Tetreault. 2009. Adapting to student uncertainty improves tutoring dialogues. Proceedings of the 14th International Conference on Artificial Intelligence in Education (AIED), 33-40.

A.C. Graesser, S. Lu, B. Olde, E. Cooper-Pye, and S. Whitten. 2005. Question asking and eye tracking during cognitive disequilibrium: comprehending illustrated texts on devices when the devices break down. Memory & Cognition, 33(7): 1235-1247.

S. Greene and P. Resnik. 2009. More than words: Syntactic packaging and implicit sentiment. Proceedings of the 2009 Annual Conference of the North American Chapter of the ACL and Human Language Technologies (NAACL HLT), 503-511.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. 2009. The WEKA data mining software: An update. SIGKDD Explorations, 11(1): 10–18.

R. Iida, S. Kobayashi, and T. Tokunaga. 2010. Incorporating extra-linguistic information into reference resolution in collaborative task dialogue. Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, 1259-1267.

M.L. Knapp and J.A. Hall. 2006. Nonverbal Communication in Human Interaction (6th ed.). Belmont, CA: Wadsworth/Thomson Learning.

C.M. Lee, S.S. Narayanan. 2005. Toward detecting emotions in spoken dialogs. IEEE Transactions on Speech and Audio Processing, 13(2): 293-303.

R. López-Cózar, J. Silovsky, and D. Griol. 2010. F2– New Technique for Recognition of User Emotional States in Spoken Dialogue Systems. Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), 281-288.

B.T. McDaniel, S. D'Mello, B.G. King, P. Chipman, K. Tapp, and A.C. Graesser. 2007. Facial Features for Affective State Detection in Learning Environments. Proceedings of the 29th Annual Cognitive Science Society, 467-472.

D. McNeill. 1992. Hand and mind: What gestures reveal about thought. Chicago: University of Chicago Press.

A. Mehrabian. 2007. Nonverbal Communication. New Brunswick, NJ: Aldine Transaction.

T. Nguyen. 2010. Mood patterns and affective lexicon access in weblogs. Proceedings of the ACL 2010 Student Research Workshop, 43-48.

M. Pantic and M.S. Bartlett. 2007. Machine Analysis of Facial Expressions. In K. Delac and M. Grgic (Eds.), Face Recognition, 377-416. Vienna, Austria: I-Tech Education and Publishing.

J.A. Russell. 2003. Core affect and the psychological construction of emotion. Psychological Review, 110(1): 145-172.

J.A. Russell, J.A. Bachorowski, and J.M. Fernandez-Dols. 2003. Facial and vocal expressions of emotion. Annual Review of Psychology, 54, 329-49.

K.L. Schmidt and J.F. Cohn. 2001. Human Facial Expressions as Adaptations: Evolutionary Questions in Facial Expression Research. Am J Phys Anthropol, 33: 3-24.

V.J. Shute. 2008. Focus on Formative Feedback. Review of Educational Research, 78(1): 153-189.

V.K.R Sridar, S. Bangalore, and S.S. Narayanan. 2009. Combining lexical, syntactic and prosodic cues for improved online dialog act tagging. Computer Speech & Language, 23(4): 407-422. Elsevier Ltd.

A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, et al. 2000. Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech. Computational Linguistics, 26(3): 339-373.

C. Toprak, N. Jakob, and I. Gurevych. 2010. Sentence and expression level annotation of opinions in user-generated discourse. Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, 575-584.

T. Wilson, J. Wiebe, and P. Hoffmann. 2009. Recognizing Contextual Polarity: An Exploration of Features for Phrase-Level Sentiment Analysis. Computational Linguistics, 35(3): 399-433.

Z. Zeng, M. Pantic, G.I. Roisman, and T.S. Huang. 2009. A Survey of Affect Recognition Methods: Audio, Visual, and Spontaneous Expressions. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(1): 39-58.

Fine-Grained Class Label Markup of Search Queries

Joseph Reisinger*
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712
joeraii@cs.utexas.edu

Marius Paşca
Google Inc.
1600 Amphitheatre Parkway
Mountain View, California 94043
mars@google.com

Abstract

We develop a novel approach to the semantic analysis of short text segments and demonstrate its utility on a large corpus of Web search queries. Extracting meaning from short text segments is difficult as there is little semantic redundancy between terms; hence methods based on shallow semantic analysis may fail to accurately estimate meaning. Furthermore search queries lack explicit syntax often used to determine intent in question answering. In this paper we propose a hybrid model of semantic analysis combining explicit class-label extraction with a latent class PCFG. This *class-label correlation* (CLC) model admits a robust parallel approximation, allowing it to scale to large amounts of query data. We demonstrate its performance in terms of (1) its predicted label accuracy on polysemous queries and (2) its ability to accurately chunk queries into base constituents.

1 Introduction

Search queries are generally short and rarely contain much explicit syntax, making query understanding a purely semantic endeavor. Furthermore, as in noun-phrase understanding, shallow lexical semantics is often irrelevant or misleading; e.g., the query [*tropical breeze cleaners*] has little to do with island vacations, nor are desert birds relevant to [*1970 road runner*], which refers to a car model.

This paper introduces *class-label correlation* (CLC), a novel unsupervised approach to extracting shallow semantic content that combines class-based semantic markup (e.g., *road runner* is a *car model*) with a latent variable model for capturing weakly compositional interactions between query constituents. Constituents are tagged with IsA class labels from a large, automatically extracted lexicon, using a probabilistic context free grammar (PCFG). Correlations between the resulting label→term distributions are captured using a set of latent production rules specified by a hierarchical Dirichlet Process (Teh et al., 2006) with latent data groupings.

Concretely, the IsA tags capture the inventory of potential meanings (e.g., *jaguar* can be labeled as *european car* or *large cat*) and relevant constituent spans, while the latent variable model performs sense and theme disambiguation (e.g., [*jaguar habitat*] would lend evidence for the *large cat* label). In addition to broad sense disambiguation, CLC can distinguish closely related usages, e.g., the use of *dell* in [*dell motherboard replacement*] and [*dell stock price*].[1] Furthermore, by employing IsA class labeling as a preliminary step, CLC can account for common non-compositional phrases, such as *big apple* unlike systems relying purely on lexical semantics. Additional examples can be found later, in Figure 5.

In addition to improving query understanding, potential applications of CLC include: (1) relation extraction (Baeza-Yates and Tiberi, 2007), (2) query substitutions or broad matching (Jones et al., 2006), and (3) classifying other short textual fragments such as SMS messages or tweets.

We implement a parallel inference procedure for

*Contributions made during an internship at Google.

[1] Dell the *computer system* vs. Dell the *technology company*.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1200–1209,
Portland, Oregon, June 19-24, 2011.

CLC and evaluate it on a sample of 500M search queries along two dimensions: (1) query constituent chunking precision (i.e., how accurate are the inferred spans breaks; cf., Bergsma and Wang (2007); Tan and Peng (2008)), and (2) class label assignment precision (i.e., given the query intent, how relevant are the inferred class labels), paying particular attention to cases where queries contain ambiguous constituents. CLC compares favorably to several simpler submodels, with gains in performance stemming from coarse-graining related class labels and increasing the number of clusters used to capture between-label correlations.

(**Paper organization**): Section 2 discusses relevant background, Section 3 introduces the CLC model, Section 4 describes the experimental setup employed, Section 5 details results, Section 6 introduces areas for future work and Section 7 concludes.

2 Background

Query understanding has been studied extensively in previous literature. Li (2010) defines the semantic structure of noun-phrase queries as *intent heads* (attributes) coupled with some number of *intent modifiers* (attribute values), e.g., the query [*alice in wonderland 2010 cast*] is comprised of an intent head *cast* and two intent modifiers *alice in wonderland* and *2010*. In this work we focus on semantic class markup of query constituents, but our approach could be easily extended to account for query structure as well.

Popescu et al. (2010) describe a similar class-label-based approach for query interpretation, explicitly modeling the importance of each label for a given entity. However, details of their implementation were not publicly available, as of publication of this paper.

For simplicity, we extract class labels using the seed-based approach proposed by Van Durme and Paşca (2008) (in particular Paşca (2010)) which generalizes Hearst (1992). Talukdar and Pereira (2010) use graph-based semi-supervised learning to acquire class-instance labels; Wang et al. (2009) introduce a similar CRF-based approach but only apply it to a small number of verticals (i.e., *Computing and Electronics* or *Clothing and Shoes*). Snow et al. (2006) describe a learning approach for automatically acquiring patterns indicative of hypernym (IsA) relations. Semantic class label lexicons derived from any of these approaches can be used as input to CLC.

Several authors have studied query clustering in the context of information retrieval (e.g., Beeferman and Berger, 2000). Our approach is novel in this regard, as we cluster queries in order to capture correlations between span labels, rather than explicitly for query understanding.

Tratz and Hovy (2010) propose a taxonomy for classifying and interpreting noun-compounds, focusing specifically on the relationships holding between constituents. Our approach yields similar topical decompositions of noun-phrases in queries and is completely unsupervised.

Jones et al. (2006) propose an automatic method for *query substitution*, i.e., replacing a given query with another query with the similar meaning, overcoming issues with poor paraphrase coverage in tail queries. Correlations mined by our approach are readily useful for downstream query substitution.

Bergsma and Wang (2007) develop a supervised approach to query chunking using 500 hand-segmented queries from the AOL corpus. Tan and Peng (2008) develop a generative model of query segmentation that makes use of a language model and concepts derived from Wikipedia article titles. CLC differs fundamentally in that it learns concept label markup in addition to segmentation and uses in-domain concepts derived from queries themselves. This work also differs from both of these studies significantly in scope, training on 500M queries instead of just 500.

At the level of class-label markup, our model is related to Bayesian PCFGs (Liang et al., 2007; Johnson et al., 2007b), and is a particular realization of an *Adaptor Grammar* (Johnson et al., 2007a; Johnson, 2010).

Szpektor et al. (2008) introduce a model of *contextual preferences*, generalizing the notion of selectional preference (cf. Ritter et al., 2010) to arbitrary terms, allowing for context-sensitive inference. Our approach differs in its use of class-instance labels for generalizing terms, a necessary step for dealing with the lack of syntactic information in queries.

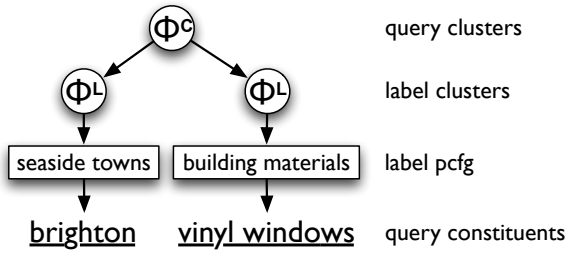


Figure 1: Overview of CLC markup generation for the query [*brighton vinyl windows*]. Arrows denote multinomial distributions.

class label→query span
recreational facilities→jacuzzi
rural areas→wales
destinations→wales
seaside towns→brighton
building materials→vinyl windows
consumer goods→european clothing

Table 1: Example production rules collected using the semi-supervised approach of Van Durme and Paşca (2008).

3 Latent Class-Label Correlation

Input to CLC consists of raw search queries and a partial grammar mapping class labels to query spans (e.g., building materials→vinyl windows). CLC infers two additional latent productions types on top of these class labels: (1) a potentially infinite set of label clusters $\phi^L_{l_k}$ coarse-graining the raw input label productions V, and (2) a finite set of query clusters $\phi^C_{c_i}$ specifying distributions over label clusters; see Figure 1 for an overview.

Operationally, CLC is implemented as a Hierarchical Dirichlet Process (HDP; Teh et al., 2006) with latent groups coupled with a Probabilistic Context Free Grammar (PCFG) likelihood function (Figure 2). We motivate our use of an HDP latent class model instead of a full PCFG with binary productions by the fact that the space of possible binary rule combinations is prohibitively large (561K base labels; 314B binary rules). The next sections discuss the three main components of CLC: §3.1 the raw IsA class labels, §3.2 the PCFG likelihood, and §3.3 the HDP with latent groupings.

3.1 IsA Label Extraction

IsA class labels (hypernyms) V are extracted from a large corpus of raw Web text using the method proposed by Van Durme and Paşca (2008) and extended by Paşca (2010). Manually specified patterns are used to extract a seed set of class labels and the resulting label lists are reranked using cluster purity measures. 561K labels for base noun phrases are collected. Table 1 shows an example set of class labels extracted for several common noun phrases. Similar repositories of IsA labels, extracted using other methods, are available for experimental purposes (Talukdar and Pereira, 2010). In addition to extracted rules, the CLC grammar is augmented with a set of *null rules*, one per unigram, ensuring that every query has a valid parse.

3.2 Class-Label PCFG

In addition to the observed class-label production rules, CLC incorporates two sets of latent production rules coupled via an HDP (Figure 1). Class label→query span productions extracted from raw text are clustered into a set of latent *label production clusters* $\mathcal{L} = \{l_1, \ldots, l_\infty\}$. Each label production cluster l_k defines a multinomial distribution over class labels V parametrized by $\phi^L_{l_k}$. Conceptually, $\phi^L_{l_k}$ captures a set of class labels with similar productions that are found in similar queries, for example the class labels *states, northeast states, u.s. states, state areas, eastern states,* and *certain states* might be included in the same coarse-grained cluster due to similarities in their productions.

Each query $q \in \mathcal{Q}$ is assigned to a latent *query cluster* $c_q \in \mathcal{C}\{c_1, \ldots, c_\infty\}$, which defines a distribution over label production clusters $\mathcal{L}$, denoted $\phi^C_{c_q}$. Query clusters capture broad correlations between label production clusters and are necessary for performing sense disambiguation and capturing selectional preference. Query clusters and label production clusters are linked using a single HDP, allowing the number of label clusters to vary over the course of Gibbs sampling, based on the variance of the underlying data (Section 3.3). Viewed as a grammar, CLC only contains unary rules mapping labels to query spans; production correlations are captured directly by the query cluster, unlike in HDP-PCFG (Liang et al., 2007), as branching parses over the en-

		Indices	Cardinality
HDP base measure	$\boldsymbol{\beta} \sim \text{GEM}(\boldsymbol{\gamma})$	-	$\lvert\mathcal{L}\rvert \to \infty$
Query cluster	$\boldsymbol{\phi}^C_i \sim \text{DP}(\boldsymbol{\alpha}^C, \boldsymbol{\beta})$	$i \in \lvert C\rvert$	$\lvert\mathcal{L}\rvert \to \infty$
Label cluster	$\boldsymbol{\phi}^L_k \sim \text{Dirichlet}(\boldsymbol{\alpha}^L)$	$k \in \lvert\mathcal{L}\rvert$	$\lvert V\rvert$
Query cluster ind	$\boldsymbol{\pi}_q \sim \text{Dirichlet}(\boldsymbol{\xi})$	$q \in \lvert\mathcal{Q}\rvert$	$\lvert C\rvert$
	$c_q \sim \boldsymbol{\pi}_q$	$q \in \lvert\mathcal{Q}\rvert$	1
Label cluster ind	$z_{q,t} \sim \boldsymbol{\phi}^C_{c_q}$	$\mathbf{t} \in q, q \in \lvert\mathcal{Q}\rvert$	1
Label ind	$l_{q,t} \sim \boldsymbol{\phi}^L_{z_{q,t}}$	$\mathbf{t} \in q, q \in \lvert\mathcal{Q}\rvert$	1

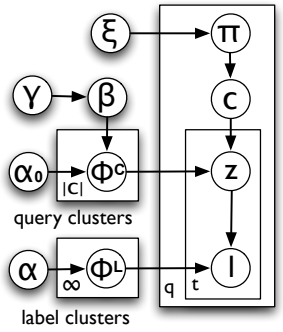


Figure 2: Generative process and graphical model for CLC. The top section of the model is the standard HDP prior; the middle section is the additional machinery necessary for modeling latent groupings and the bottom section contains the indicators for the latent class model. PCFG likelihood is not shown.

tire label sparse are intractably large.

Given a query q, a query cluster assignment c_q and a set of label production clusters $\mathcal{L}$, we define a parse of q to be a sequence of productions $\mathbf{t}_q$ forming a parse tree consuming all the tokens in q. As with Bayesian PCFGs (Johnson, 2010), the probability of a tree $\mathbf{t}_q$ is the product of the probabilities of the production rules used to construct it

$$P(\mathbf{t}_q|\boldsymbol{\phi}^L, \boldsymbol{\phi}^C, c_q) = \prod_{r \in R_q} P(r|\boldsymbol{\phi}^L_{l_r}) P(l_r|\boldsymbol{\phi}^C_{c_q})$$

where R_q is the set of production rules used to derive $\mathbf{t}_q$, $P(r|\phi^L_{l_r})$ is the probability of r given its label cluster assignment l_r, and $P(l_r|\phi^C_{c_q})$ is the probability of label cluster l_r in query cluster c.

The probability of a query q is the sum of the probabilities of the parse trees that can generate it,

$$P(q|\boldsymbol{\phi}^L, \boldsymbol{\phi}^C, c_q) = \sum_{\{\mathbf{t}|y(\mathbf{t})=q\}} P(\mathbf{t}|\boldsymbol{\phi}^L, \boldsymbol{\phi}^C, c_q)$$

where $\{\mathbf{t}|y(\mathbf{t}) = q\}$ is the set of trees with q as their *yield* (i.e., generate the string of tokens in q).

3.3 Hierarchical Dirichlet Process with Latent Groups

We complete the Bayesian generative specification of CLC with an HDP prior linking $\boldsymbol{\phi}^C$ and $\boldsymbol{\phi}^L$. The HDP is a Bayesian generative model of shared structure for grouped data (Teh et al., 2006). A set of base clusters $\boldsymbol{\beta} \sim \text{GEM}(\boldsymbol{\gamma})$ is drawn from a Dirichlet Process with base measure $\boldsymbol{\gamma}$ using the stick-breaking construction, and clusters for each group k, $\boldsymbol{\phi}^C_k \sim \text{DP}(\boldsymbol{\beta})$, are drawn from a separate Dirichlet Process with base measure $\boldsymbol{\beta}$, defined over the space of label clusters. Data in each group k are conditionally independent given $\boldsymbol{\beta}$. Intuitively, $\boldsymbol{\beta}$ defines a common "menu" of label clusters, and each query cluster $\boldsymbol{\phi}^C_k$ defines a separate distribution over the label clusters.

$\boldsymbol{\gamma}$	– HDP-LG base-measure smoother; higher values lead to more uniform mass over label clusters.
$\boldsymbol{\alpha}^C$	– Query cluster smoothing; higher values lead to more uniform mass over label clusters.
$\boldsymbol{\alpha}^L$	– Label cluster smoothing; higher values lead to more label diversity within clusters.
$\boldsymbol{\xi}$	– Query cluster assignment smoothing; higher values lead to more uniform assignment.

Table 2: CLC-HDP-LG hyperparameters.

In order to account for variable query-cluster assignment, we extend the HDP model with *latent groupings* $\boldsymbol{\pi}_q \sim \text{Dir}(\xi)$ for each query. The resulting *Hierarchical Dirichlet Process with Latent Groups* (HDP-LG) can be used to define a set of query clusters over a set of (potentially infinite) base label clusters (Figure 2). Each query cluster $\boldsymbol{\phi}^C$ (latent group) assigns weight to different subsets of the available label clusters $\boldsymbol{\phi}^L$, capturing correlations between them at the query level. Each query q maintains a distribution over query clusters $\boldsymbol{\pi}_q$, capturing its affinity for each latent group. The full generative specification of CLC is shown in Figure 2; hyperparameters are shown in Table 2.

In addition to the full joint CLC model, we evalu-

ate several simpler models:

1. CLC-BASE – no query clusters, one label per label cluster.
2. CLC-DPMM – no query clusters, DPMM($\boldsymbol{\alpha}^C$) distribution over labels.
3. CLC-HDP-LG – full HDP-LG model with $|C|$ query clusters over a potentially infinite number of query clusters.

as well as various hyperparameter settings.

3.4 Parallel Approximate Gibbs Sampler

We perform inference in CLC via Gibbs sampling, leveraging Multinomial-Dirichlet conjugacy to integrate out $\boldsymbol{\pi}$, $\boldsymbol{\phi}^C$ and $\boldsymbol{\phi}^L$ (Teh et al., 2006; Johnson et al., 2007b). The remaining indicator variables $\mathbf{c}$, $\mathbf{z}$ and $\mathbf{l}$ are sampled iteratively, conditional on all other variable assignments. Although there are an exponential number of parse trees for a given query, this space can be sampled efficiently using dynamic programming (Finkel et al., 2006; Johnson et al., 2007b)

In order to apply CLC to Web-scale data, we implement an efficient parallel approximate Gibbs sampler in the MapReduce framework Dean and Ghemawat (2004). Each Gibbs iteration consists of a single MapReduce step for sampling, followed by an additional MapReduce step for computing marginal counts. [2] Relevant assignments $\mathbf{c}$, $\mathbf{z}$ and $\mathbf{l}$ are stored locally with each query and are distributed across compute nodes. Each node is responsible only for resampling assignments for its local set of queries. Marginals are fetched opportunistically from a separate distributed hash server as they are needed by the sampler. Each Map step computes a single Gibbs step for 10% of the available data, using the marginals computed at the previous step. By resampling only 10% of the available data each iteration, we minimize the potentially negative effects of using the previous step's marginal distribution.

4 Experimental Setup

4.1 Query Corpus

Our dataset consists of a sample of 450M English queries submitted by anonymous Web users to Google. The queries have an average of 3.81 tokens per query (1.7B tokens). Single token queries are removed as the model is incapable of using context to disambiguate their meaning. Figure 3 shows the distribution of remaining queries. During training, we include 10 copies of each query (4.5B queries total), allowing an estimate of the Bayes average posterior from a single Gibbs sample.

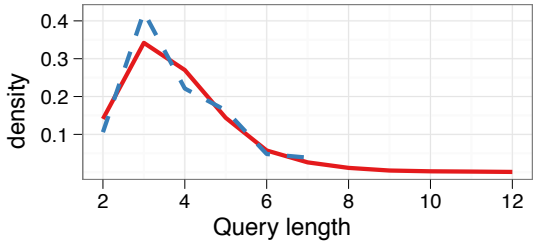


Figure 3: Distribution in the query corpus, broken down by query length (red/solid=all queries; blue/dashed=queries with ambiguous spans); most queries contain between 2-6 tokens.

4.2 Evaluations

Query markup is evaluated for phrase-chunking precision (Section 5.1) and label precision (Section 5.2) by human raters across two different samples: (1) an unbiased sample from the original corpus, and (2) a biased sample of queries containing ambiguous spans.

Two raters scored a total of 10K labels from 800 spans across 300 queries. Span labels were marked as *incorrect* (0.0), *badspan* (0.0), *ambiguous* (0.5), or *correct* (1.0), with numeric scores for label precision as indicated. Chunking precision is measured as the percentage of labels not marked as *badspan*.

We report two sets of precision scores depending on how *null* labels are handled: *Strict* evaluation treats null-labeled spans as incorrect, while *Normal* evaluation removes null-labeled spans from the precision calculation. Normal evaluation was included since the simpler models (e.g., CLC-BASE) tend to produce a significantly higher number of *null* assignments.

Model evaluations were broken down into *maximum a posteriori* (MAP) and *Bayes average* estimates. MAP estimates are calculated as the single most likely label/cluster assignment across all query copies; all assignments in the sample are averaged

[2] This approximation and architecture is similar to Smola and Narayanamurthy (2010).

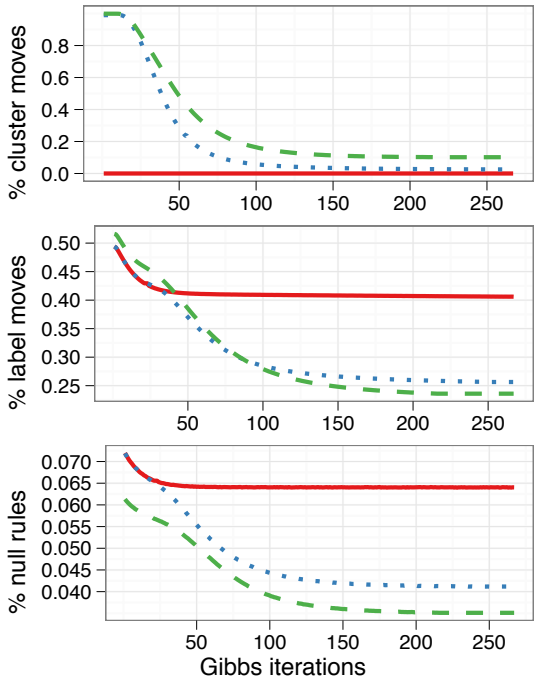


Figure 4: Convergence rates of CLC-BASE (red/solid), CLC-HDP-LG $100C$,$40L$ (green/dashed), CLC-HDP-LG $1000C$,$40L$ (blue/dotted) in terms of % of query cluster swaps, label cluster swaps and null rule assignments.

to obtain the Bayes average precision estimate.[3]

5 Results

A total of five variants of CLC were evaluated with different combinations of $|C|$ and HDP prior concentration α^C (controlling the effective number of label clusters). Referring to models in terms of their parametrizations is potentially confusing. Therefore, we will make use of the fact that models with $\boldsymbol{\alpha}^C = 1$ yielded roughly 40 label clusters on average, and models with $\boldsymbol{\alpha}^C = 0.1$ yielded roughly 200 label clusters, naming model variants simply by the number of query and label clusters: (1) CLC-BASE, (2) CLC-DPMM 1C-40L, (3) CLC-HDP-LG 100C-40L, (4) CLC-HDP-LG 1000C-40L, and (5) CLC-HDP-LG 1000C-200L. Figure 4 shows the model convergence for CLC-BASE, CLC-HDP-LG 100C-40L, and CLC-HDP-LG 1000C-40L.

[3]We calculate the Bayes average precision estimates at the top 10 (Bayes@10) and top 20 (Bayes@20) parse trees, weighted by probability.

5.1 Chunking Precision

Chunking precision scores for each model are shown in Table 3 (average % of labels not marked *badspan*). CLC-HDP-LG 1000C-40L has the highest precision across both MAP and Bayes estimates (∼93% accuracy), followed by CLC-HDP-LG 1000C-200L (∼90% accuracy) and CLC-DPMM 1C-40L (∼85%). CLC-BASE performed the worst by a significant margin (∼78%), indicating that label coarse-graining is more important than query clustering for chunking accuracy. No significant differences in label chunking accuracy were found between Bayes and MAP inference.

5.2 Predicting Span Labels

The full CLC-HDP-LG model variants obtain higher label precision than the simpler models, with CLC-HDP-LG 1000C-40L achieving the highest precision of the three (∼63% accuracy). Increasing the number of label clusters too high, however, significantly reduces precision: CLC-HDP-LG 1000C-200L obtains only ∼51% accuracy. However, comparing to CLC-DPMM 1C-40L and CLC-BASE demonstrates that the addition of label clusters and query clusters both lead to gains in label precision. These relative rankings are robust across *strict* and *normal* evaluation regimes.

The breakdown over MAP and Bayes posterior estimation is less clear when considering label precision: the simpler models CLC-BASE and CLC-DPMM 1C-40L perform significantly worse than Bayes when using MAP estimation, while in CLC-HDP-LG the reverse holds.

There is little evidence for correlation between precision and query length (weak, not statistically significant negative correlation using Spearman's ρ). This result is interesting as the relative prevalence of natural language queries increases with query length, potentially degrading performance. However, we did find a strong positive correlation between precision and the number of labels productions applicable to a query, i.e., production rule fertility is a potential indicator of semantic quality.

Finally, the *histogram* column in Table 3 shows the distribution of rater responses for each model. In general, the more precise models tend to have a significantly lower proportion of missing spans

Model	Chunking Precision	Label Precision			Ambiguous Label Precision		Spearman's ρ	
		normal	strict	hist	normal	strict	q. len	# labels
Class-Label Correlation Base								
Bayes@10	78.7±1.1	37.7±1.2	35.8±1.2		35.4±2.0	33.2±1.9	-0.13	0.51•
Bayes@20	78.7±1.1	37.7±1.2	35.8±1.2		35.4±2.0	33.2±1.9	-0.13	0.51•
MAP	76.3±2.2	33.3±2.2	31.8±2.2		36.2±4.0	33.2±3.8	-0.13	0.52•
Class-Label Correlation DPMM 1C 40L								
Bayes@10	84.9±0.4	46.6±0.6	44.3±0.5		36.0±1.1	33.7±1.0	-0.05	0.25
Bayes@20	84.8±0.4	47.4±0.5	45.2±0.5		37.8±1.0	35.5±1.0	-0.02	0.23
MAP	84.1±0.8	42.6±1.0	40.5±0.9		11.2±1.3	10.6±1.3	-0.03	0.12
Class-Label Correlation HDP-LG 100C 40L								
Bayes@10	83.8±0.4	55.6±0.5	51.0±0.5		**55.6±1.0**	**47.7±1.0**	0.03	0.44•
Bayes@20	83.6±0.4	56.9±0.5	52.3±0.5		**57.4±1.0**	**49.8±0.9**	0.04	0.41•
MAP	82.7±0.5	58.5±0.5	53.6±0.5		**60.4±1.1**	**51.5±1.0**	0.02	0.41•
Class-Label Correlation HDP-LG 1000C 40L								
Bayes@10	**93.1±0.2**	**61.1±0.3**	**60.0±0.3**		43.2±0.9	40.2±0.9	-0.06	0.26•
Bayes@20	**92.8±0.2**	**62.6±0.3**	**61.7±0.3**		44.9±0.8	42.2±0.8	-0.10	0.27•
MAP	**92.7±0.2**	**63.7±0.3**	**62.7±0.3**		44.1±0.9	41.1±0.9	-0.12	0.28•
Class-Label Correlation HDP-LG 1000C 200L								
Bayes@10	90.3±0.5	50.9±0.8	48.6±0.7		45.8±1.5	42.5±1.3	-0.10	0.13
Bayes@20	89.9±0.5	50.2±0.7	48.0±0.7		44.4±1.4	41.3±1.3	-0.08	0.11
MAP	90.0±0.6	51.0±0.8	48.9±0.8		49.2±1.5	46.0±1.4	-0.07	0.04

Table 3: Chunking and label precision across five models. Confidence intervals are standard error; sparklines show distribution of precision scores (left is zero, right is one). *Hist* shows the distribution of human rating response (log y scale): green/first is correct, blue/second is ambiguous, cyan/third is missing and red/fourth is incorrect. Spearman's ρ columns give label precision correlations with query length (weak negative correlation) and the number of applicable labels (weak to strong positive correlation); dots indicate significance.

(blue/second bar; due to null rule assignment) in additional to more correct (green/first) and fewer incorrect (red/fourth) spans.

5.3 High Polysemy Subset

We repeat the analysis of label precision on a subset of queries containing one of the manually-selected polysemous spans shown in Table 4. The CLC-HDP-LG -based models still significantly outperform the simpler models, but unlike in the broader setting, CLC-HDP-LG 100C-40L significantly outperforms CLC-HDP-LG 1000C-40L, indicating that lower query cluster granularity helps address polysemy (Table 3).

5.4 Error Analysis

Figure 5 gives examples of both high-precision and low-precision queries markups inferred by CLC-HDP-LG. In general, CLC performs well on queries with clear *intent head / intent modifier* structure (Li, 2010). More complex queries, such as [*never know until you try quotes*] or [*how old do you have to be a bartender in new york*] do not fit this model; however, expanding the set of extracted labels to also cover instances such as *never know until you try* would mitigate this problem, motivating the use of n-gram language models with semantic markup.

acapella, alamo, apple, atlas, bad, bank, batman, beloved, black forest, bravo, bush, canton, casino, champion, club, comet, concord, dallas, diamond, driver, english, ford, gamma, ion, lemon, manhattan, navy, pa, palm, port, put, resident evil, ronaldo, sacred heart, saturn, seven, solution, sopranos, sparta, supra, texas, village, wolf, young

Table 4: Samples from a list of 90 manually selected ambiguous spans used to evaluate model performance under polysemy.

A large number of mistakes made by CLC are

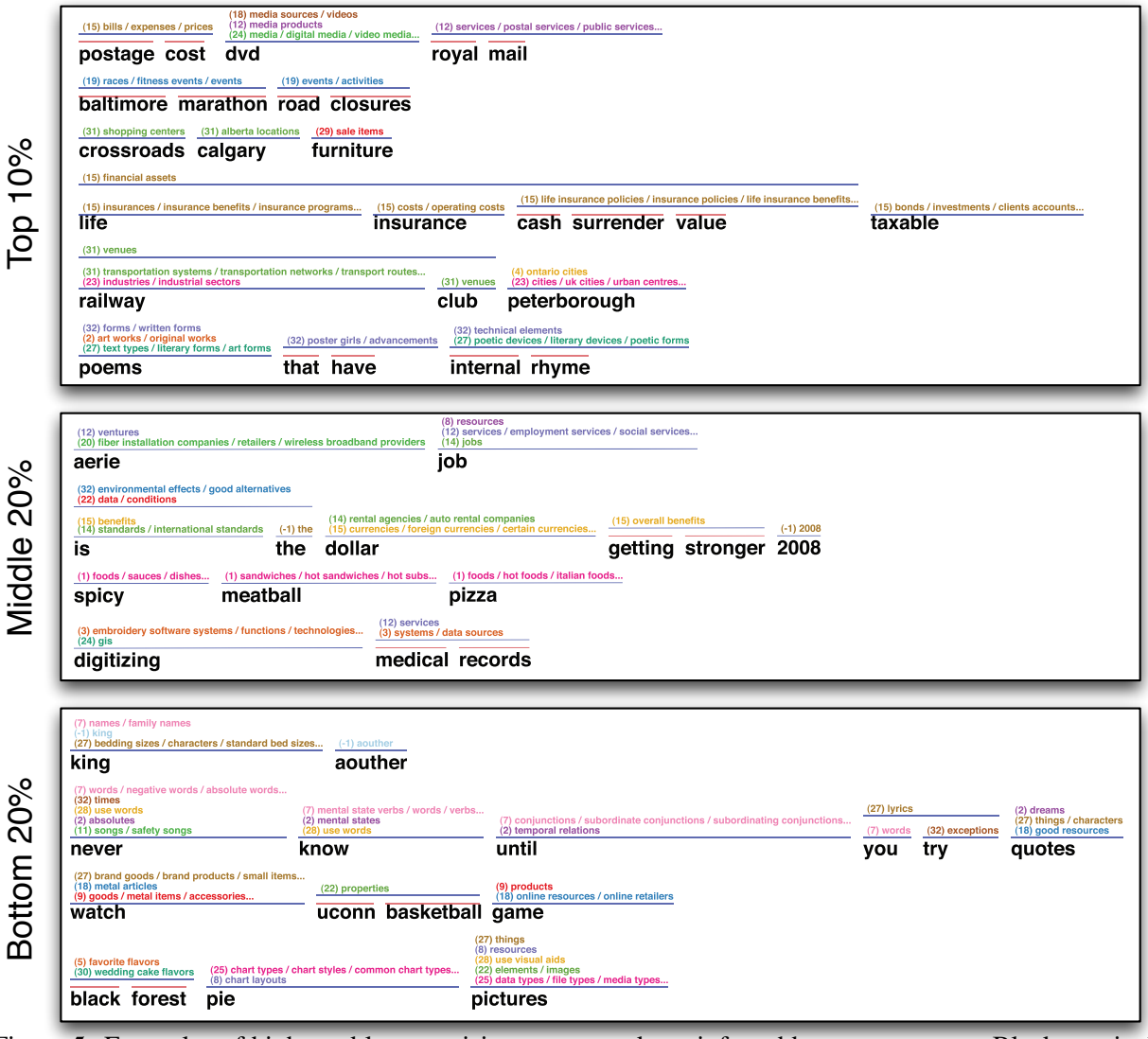


Figure 5: Examples of high- and low-precision query markups inferred by CLC-HDP-LG. Black text is the original query; lines indicate potential spans; small text shows potential labels colored and numbered by label cluster; small bar shows percentage of assignments to that label cluster.

due to named-entity categories with weak semantics such as rock bands or businesses (e.g., [*tropical breeze cleaners*], [*cosmic railroad band*] or [*sopranos cigars*]). When the named entity is common enough, it is detected by the rule set, but for the long tail of named entities this is not the case. One potential solution is to use a stronger notion of *selectional preference* and slot-filling, rather than just relying on correlation between labels.

Other examples of common errors include interpreting *weymouth* in [*weymouth train time table*] as a town in Massachusetts instead of a town in the UK (lack of domain knowledge), and using lower quality semantic labels (e.g., *neighboring countries* for *france*, or *great retailers* for *target*).

6 Discussion and Future Work

Adding both latent label clusters (DPMM) and latent query clusters (extending to HDP-LG) improve chunking and label precision over the baseline CLC-BASE system. The label clusters are important because they capture intra-group correlations between class labels, while the query clusters are important for capturing inter-group correlations. However, the algorithm is sensitive to the relative number of clusters in each case: Too many labels/label clusters rel-

ative to the number of query clusters make it difficult to learn correlations ($O(n^2)$ query clusters are required to capture pairwise interactions). Too many query clusters, on the other hand, make the model intractable computationally. The HDP automates selecting the number of clusters, but still requires manual hyperparameter setting.

(**Future Work**) Many query slots have weak semantics and hence are misleading for CLC. For example [*pacific breeze cleaners*] or [*dale hartley subaru*] should be parsed such that the type of the leading slot is determined not by its direct content, but by its context; seeing *subaru* or *cleaners* after a noun-phrase slot is a strong indicator of its type (*dealership* or *shop name*). The current CLC model only couples these slots through their correlations in query clusters, not directly through relative position or context. Binary productions in the PCFG or a discriminative learning model would help address this.

Finally, we did not measure label coverage with respect to a human evaluation set; coverage is useful as it indicates whether our inferred semantics are biased with respect to human norms.

7 Conclusions

We introduced CLC, a set of latent variable PCFG models for semantic analysis of short textual segments. CLC captures semantic information in the form of interactions between clusters of automatically extracted class-labels, e.g., finding that placenames commonly co-occur with business-names. We applied CLC to a corpus containing 500M search queries, demonstrating its scalability and straightforward parallel implementation using frameworks like MapReduce or Hadoop. CLC was able to chunk queries into spans more accurately and infer more precise labels than several sub-models even across a highly ambiguous query subset. The key to obtaining these results was coarse-graining the input class-label set and using a latent variable model to capture interactions between coarse-grained labels.

References

R. Baeza-Yates and A. Tiberi. 2007. Extracting semantic relations from query logs. In *Proceedings of the 13th ACM Conference on Knowledge Discovery and Data Mining (KDD-07)*, pages 76–85. San Jose, California.

D. Beeferman and A. Berger. 2000. Agglomerative clustering of a search engine query log. In *Proceedings of the 6th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD-00)*, pages 407–416.

S. Bergsma and Q. Wang. 2007. Learning noun phrase query segmentation. In *Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP-07)*, pages 819–826. Prague, Czech Republic.

J. Dean and S. Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI-04)*, pages 137–150. San Francisco, California.

J. Finkel, C. Manning, and A. Ng. 2006. Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP-06)*, pages 618–626. Sydney, Australia.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 539–545. Nantes, France.

M. Johnson. 2010. PCFGs, topic models, adaptor grammars and learning topical collocations and the structure of proper names. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, pages 1148–1157. Uppsala, Sweden.

M. Johnson, T. Griffiths, and S. Goldwater. 2007a. Adaptor grammars: a framework for specifying compositional nonparametric bayesian models. In *Advances in Neural Information Processing Systems 19*, pages 641–648. Vancouver, Canada.

M. Johnson, T. Griffiths, and S. Goldwater. 2007b. Bayesian inference for PCFGs via Markov Chain Monte Carlo. In *Proceedings of the 2007 Conference of the North American Association for Computational Linguistics (NAACL-HLT-07)*, pages 139–146. Rochester, New York.

R. Jones, B. Rey, O. Madani, and W. Greiner. 2006. Generating query substitutions. In *Proceedings of the 15h World Wide Web Conference (WWW-06)*, pages 387–396. Edinburgh, Scotland.

X. Li. 2010. Understanding the semantic structure of noun phrase queries. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, pages 1337–1345. Uppsala, Sweden.

P. Liang, S. Petrov, M. Jordan, and D. Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP-07)*, pages 688–697. Prague, Czech Republic.

M. Paşca. 2010. The role of queries in ranking labeled instances extracted from text. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING-10)*, pages 955–962. Beijing, China.

A. Popescu, P. Pantel, and G. Mishne. 2010. Semantic lexicon adaptation for use in query interpretation. In *Proceedings of the 19th World Wide Web Conference (WWW-10)*, pages 1167–1168. Raleigh, North Carolina.

A. Ritter, Mausam, and O. Etzioni. 2010. A latent Dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, pages 424–434. Uppsala, Sweden.

A. Smola and S. Narayanamurthy. 2010. An architecture for parallel topic models. In *Proceedings of the 36th Conference on Very Large Data Bases (VLDB-10)*, pages 703–710. singapore.

R. Snow, D. Jurafsky, and A. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*, pages 801–808. Sydney, Australia.

I. Szpektor, I. Dagan, R. Bar-Haim, and J. Goldberger. 2008. Contextual preferences. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pages 683–691. Columbus, Ohio.

P. Talukdar and F. Pereira. 2010. Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, pages 1473–1481. Uppsala, Sweden.

B. Tan and F. Peng. 2008. Unsupervised query segmentation using generative language models and Wikipedia. In *Proceedings of the 17th World Wide Web Conference (WWW-08)*, pages 347–356. Beijing, China.

Y. Teh, M. Jordan, M. Beal, and D. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

S. Tratz and E. Hovy. 2010. A taxonomy, dataset, and classifier for automatic noun compound interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, pages 678–687. Uppsala, Sweden.

B. Van Durme and M. Paşca. 2008. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI-08)*, pages 1243–1248. Chicago, Illinois.

T. Wang, R. Hoffmann, X. Li, and J. Szymanski. 2009. Semi-supervised learning of semantic classes for query understanding: from the Web and for the Web. In *Proceedings of the 18th International Conference on Information and Knowledge Management (CIKM-09)*, pages 37–46. Hong Kong, China.

Creating a manually error-tagged and shallow-parsed learner corpus

Ryo Nagata
Konan University
8-9-1 Okamoto,
Kobe 658-0072 Japan
rnagata @ konan-u.ac.jp.

Edward Whittaker **Vera Sheinman**
The Japan Institute for
Educational Measurement Inc.
3-2-4 Kita-Aoyama, Tokyo, 107-0061 Japan
{whittaker,sheinman}@jiem.co.jp

Abstract

The availability of learner corpora, especially those which have been manually error-tagged or shallow-parsed, is still limited. This means that researchers do not have a common development and test set for natural language processing of learner English such as for grammatical error detection. Given this background, we created a novel learner corpus that was manually error-tagged and shallow-parsed. This corpus is available for research and educational purposes on the web. In this paper, we describe it in detail together with its data-collection method and annotation schemes. Another contribution of this paper is that we take the first step toward evaluating the performance of existing POS-tagging/chunking techniques on learner corpora using the created corpus. These contributions will facilitate further research in related areas such as grammatical error detection and automated essay scoring.

1 Introduction

The availability of learner corpora is still somewhat limited despite the obvious usefulness of such data in conducting research on natural language processing of learner English in recent years. In particular, learner corpora tagged with grammatical errors are rare because of the difficulties inherent in learner corpus creation as will be described in Sect. 2. As shown in Table 1, error-tagged learner corpora are very few among existing learner corpora (see Leacock et al. (2010) for a more detailed discussion of learner corpora). Even if data is error-tagged, it is often not available to the public or its access is severely restricted. For example, the Cambridge Learner Corpus, which is one of the largest error-tagged learner corpora, can only be used by authors and writers working for Cambridge University Press and by members of staff at Cambridge ESOL.

Error-tagged learner corpora are crucial for developing and evaluating error detection/correction algorithms such as those described in (Rozovskaya and Roth, 2010b; Chodorow and Leacock, 2000; Chodorow et al., 2007; Felice and Pulman, 2008; Han et al., 2004; Han et al., 2006; Izumi et al., 2003b; Lee and Seneff, 2008; Nagata et al., 2004; Nagata et al., 2005; Nagata et al., 2006; Tetreault et al., 2010b). This is one of the most active research areas in natural language processing of learner English. Because of the restrictions on their availability, researchers have used their own learner corpora to develop and evaluate error detection/correction methods, which are often not commonly available to other researchers. This means that the detection/correction performance of each existing method is not directly comparable as Rozovskaya and Roth (2010a) and Tetreault et al. (2010a) point out. In other words, we are not sure which methods achieve the best performance. Commonly available error-tagged learner corpora are therefore essential to further research in this area.

For similar reasons, to the best of our knowledge, there exists no such learner corpus that is manually shallow-parsed and which is also publicly available, unlike, say, native-speaker corpora such as the Penn Treebank. Such a comparison brings up another crucial question: "Do existing POS taggers and chun-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1210–1219,
Portland, Oregon, June 19-24, 2011.

Name	Error-tagged	Parsed	Size (words)	Availability
Cambridge Learner Corpus	Yes	No	30 million	No
CLEC Corpus	Yes	No	1 million	Partially
ETLC Corpus	Partially	No	2 million	Not Known
HKUST Corpus	Yes	No	30 million	No
ICLE Corpus (Granger et al., 2009)	No	No	3.7 million+	Yes
JEFLL Corpus (Tono, 2000)	No	No	1 million	Partially
Longman Learners' Corpus	No	No	10 million	Not Known
NICT JLE Corpus (Izumi et al., 2003a)	Partially	No	2 million	Partially
Polish Learner English Corpus	No	No	0.5 million	No
Janus Pannoius University Learner Corpus	No	No	0.4 million	Not Known

In *Availability*, *Yes* denotes that the full texts of the corpus is available to the public. *Partially* denotes that it is accessible through specially-made interfaces such as a concordancer. The information in this table may not be consistent because many of the URLs of the corpora give only sparse information about them.

Table 1: Learner corpus list.

kers work on learner English as well as on edited text such as newspaper articles?" Nobody really knows the answer to the question. The only exception in the literature is the work by Tetreault et al. (2010b) who evaluated parsing performance in relation to prepositions. Nevertheless, a great number of researchers have used existing POS taggers and chunkers to analyze the writing of learners of English. For instance, error detection methods normally use a POS tagger and/or a chunker in the error detection process. It is therefore possible that a major cause of false positives and negatives in error detection may be attributed to errors in POS-tagging and chunking. In corpus linguistics, researchers (Aarts and Granger, 1998; Granger, 1998; Tono, 2000) use such tools to extract interesting patterns from learner corpora and to reveal learners' tendencies. However, poor performance of the tools may result in misleading conclusions.

Given this background, we describe in this paper a manually error-tagged and shallow-parsed learner corpus that we created. In Sect. 2, we discuss the difficulties inherent in learner corpus creation. Considering the difficulties, in Sect. 3, we describe our method for learner corpus creation, including its data collection method and annotation schemes. In Sect. 4, we describe our learner corpus in detail. The learner corpus is called the Konan-JIEM learner corpus (KJ corpus) and is freely available for research and educational purposes on the web[1]. Another contribution of this paper is that we take the first step toward answering the question about the performance of existing POS-tagging/chunking techniques on learner data. We report and discuss the results in Sect. 5.

2 Difficulties in Learner Corpus Creation

In addition to the common difficulties in creating any corpus, learner corpus creation has its own difficulties. We classify them into the following four categories of the difficulty in:

1. collecting texts written by learners;
2. transforming collected texts into a corpus;
3. copyright transfer; and
4. error and POS/parsing annotation.

The first difficulty concerns the problem in collecting texts written by learners. As in the case of other corpora, it is preferable that the size of a learner corpus be as large as possible where the size can be measured in several ways including the total number of texts, words, sentences, writers, topics, and texts per writer. However, it is much more difficult to create a large learner corpus than to create a

[1] http://www.gsk.or.jp/index_e.html

large native-speaker corpus. In the case of native-speaker corpora, published texts such as newspaper articles or novels can be used as a corpus. By contrast, in the case of learner corpora, we must find learners and then let them write since there are no such published texts written by learners of English (unless they are part of a learner corpus). Here, it should be emphasized that learners often do not spontaneously write but are typically obliged to write, for example, in class, or during an exam. Because of this, learners may soon become tired of writing. This in itself can affect learner corpus creation much more than one would expect especially when creating a longitudinal learner corpus. Thus, it is crucial to keep learners motivated and focused on the writing assignments.

The second difficulty arises when the collected texts are transformed into a learner corpus. This involves several time-consuming and troublesome tasks. The texts must be archived in electronic form, which requires typing every single collected text since learners normally write on paper. Besides, each text must be archived and maintained with accompanying information such as who wrote what text when and on what topic. Optionally, a learner corpus could include other pieces of information such as proficiency, first language, and age. Once the texts have been electronically archived, it is relatively easy to maintain and access them. However, this is not the case when the texts are first collected. Thus, it is better to have an efficient method for managing such information as well as the texts themselves.

The third difficulty concerning copyright is a daunting problem. The copyright for each text must be transferred to the corpus creator so that the learner corpus can be made available to the public. Consider the case when a number of learners participate in a learner corpus creation project and everyone has to sign a copyright transfer form. This issue becomes even more complicated when the writer does not actually have such a right to transfer copyright. For instance, under the Japanese law, those younger than 20 years of age do not have the right; instead their parents do. Thus, corpus creators have to ask learners' parents to sign copyright transfer forms. This is often the case since the writers in learner corpus creation projects are normally junior high school, high school, or college students.

The final difficulty is in error and POS/parsing annotation. For error annotation, several annotation schemes exist (for example, the NICT JLE scheme (Izumi et al., 2005)). While designing an annotation scheme is one issue, annotating errors is yet another. No matter how well an annotation scheme is designed, there will always be exceptions. Every time an exception appears, it becomes necessary to revise the annotation scheme. Another issue we have to remember is that there is a trade-off between the granularity of an annotation scheme and the level of the difficulty in error annotation. The more detailed an annotation scheme is, the more information it can contain and the more difficult identifying errors is, and vice versa.

For POS/parsing annotation, there are also a number of annotation schemes including the Brown tag set, the Claws tag set, and the Penn Treebank tag set. However, none of them are designed to be used for learner corpora. In other words, a variety of linguistic phenomena occur in learner corpora which the existing annotation schemes do not cover. For instance, spelling errors often appear in texts written by learners of English as in *sard year*, which should be *third year*. Grammatical errors prevent us applying existing annotation schemes, too. For instance, there are at least three possibilities for POS-tagging the word *sing* in the sentence *everyone sing together.* using the Penn Treebank tag set: sing/VB, sing/VBP, or sing/VBZ. The following example is more complicated: *I don't success cooking.* Normally, the word *success* is not used as a verb but as a noun. The instance, however, appears in a position where a verb appears. As a result, there are at least two possibilities for tagging: success/NN and success/VB. Errors in mechanics are also problematic as in *Tonight,we* and *beautifulhouse* (missing spaces)[2]. One solution is to split them to obtain the correct strings and then tag them with a normal scheme. However, this would remove the information that spaces were originally missing which we want to preserve. To handle these and other phenomena which are peculiar to learner corpora, we need to develop a novel annotation scheme.

[2]Note that the KJ corpus consists of typed essays.

3 Method

3.1 How to Collect and Maintain Texts Written by Learners

Our text-collection method is based on writing exercises. In the writing exercises, learners write essays on a blog system. This very simple idea of using a blog system naturally solves the problem of archiving texts in electronic form. In addition, the use of a blog system enables us to easily register and maintain accompanying information including who (user ID) writes when (uploaded time) and on what topic (title of blog item). Besides, once registered in the user profile, the optional pieces of information such as proficiency, first language, and age are also easy to maintain and access.

To design the writing exercises, we consulted with several teachers of English and conducted pre-experiments. Ten learners participated in the pre-experiments and were assigned five essay topics on average. Based on the experimental results, we designed the procedure of the writing exercise as shown in Table 2. In the first step, learners are assigned an essay topic. In the second step, they are given time to prepare during which they think about what to write on the given topic before they start writing. We found that this enables the students to write more. In the third step, they actually write an essay on the blog system. After they have finished writing, they submit their essay to the blog system to be registered.

The following steps were considered optional. We implemented an article error detection method (Nagata et al., 2006) in the blog system as a trial attempt to keep the learners motivated since learners are likely to become tired of doing the same exercise repeatedly. To reduce this, the blog system highlights where article errors exist after the essay has been submitted. The hope is that this might prompt the learners to write more accurately and to continue the exercises. In the pre-experiments, the detection did indeed seem to interest the learners and to provide them with additional motivation. Considering these results, we decided to include the fourth and fifth steps in the writing exercises when we created our learner corpus. At the same time, we should of course be aware that the use of error detection affects learners' writing. For example, it may change the distribution of errors. Nagata and Nakatani (2010) reported the effects in detail.

Step	Min.
1. Learner is assigned an essay topic	–
2. Learner prepares for writing	5
3. Learner writes an essay	35
4. System detects errors in the essay	5
5. Learner rewrites the essay	15

Table 2: Procedure of writing exercise.

To solve the problem of copyright transfer, we took legal professional advice but were informed that, in Japan at least, the only way to be sure is to have a copyright transfer form signed every time. We considered having it signed on the blog system, but it soon turned out that this did not work since participating learners may still be too young to have the legal right to sign the transfer. It is left for our long-term future work to devise a better solution to this legal issue.

3.2 Annotation Scheme

This subsection describes the error and POS/chunking annotation schemes. Note that errors and POS/chunking are annotated separately, meaning that there are two files for any given text. Due to space restrictions we limit ourselves to only summarizing our annotation schemes in this section. The full descriptions are available together with the annotated corpus on the web.

3.2.1 Error Annotation

We based our error annotation scheme on that used in the NICT JLE corpus (Izumi et al., 2003a), whose detailed description is readily available, for example, in Izumi et al. (2005). In that annotation scheme and accordingly in ours, errors are tagged using an XML syntax; an error is annotated by tagging a word or phrase that contains it. For instance, a tense error is annotated as follows: *I <v_tns crr="made">make</v_tns> pies last year.*

where *v_tns* denotes a tense error in a verb. It should be emphasized that the error tags contain the information on correction together with error annotation. For instance, *crr="made"* in the above example denotes the correct form of the verb is *made*. For missing word errors, error tags are placed where

a word or phrase is missing (e.g., *My friends live <prp crr="in"></prp> these places.*).

As a pilot study, we applied the NICT JLE annotation scheme to a learner corpus to reveal what modifications we needed to make. The learner corpus consisted of 455 essays (39,716 words) written by junior high and high school students[3]. The following describes the major modifications deemed necessary as a result of the pilot study.

The biggest difference between the NICT JLE corpus and our targeted corpus is that the former is spoken data and the latter is written data. This difference inevitably requires several modifications to the annotation scheme. In speech data, there are no errors in spelling and mechanics such as punctuation and capitalization. However, since such errors are not usually regarded as grammatical errors, we decided simply not to annotate them in our annotation schemes.

Another major difference is fragment errors. Fragments that do not form a complete sentence often appear in the writing of learners (e.g., *I have many books. Because I like reading.*). In written language, fragments can be regarded as a grammatical error. To annotate fragment errors, we added a new tag <f> (e.g., *I have many books.* <f>*Because I like reading.*</f>).

As discussed in Sect. 2, there is a trade-off between the granularity of an annotation scheme and the level of the difficulty in annotating errors. In our annotation scheme, we narrowed down the number of tags to 22 from 46 in the original NICT JLE tag set to facilitate the annotation; the 22 tags are shown in Appendix A. The removed tags are merged into the tag for *other*. For instance, there are only three tags for errors in nouns (number, lexis, and other) in our tag set whereas there are six in the NICT JLE corpus (inflection, number, case, countability, complement, and lexis); the *other* tag (<n_o>) covers the four removed tags.

3.2.2 POS/Chunking Annotation

We selected the Penn Treebank tag set, which is one of the most widely used tag sets, for our POS/chunking annotation scheme. Similar to the error annotation scheme, we conducted a pilot study to determine what modifications we needed to make to the Penn Treebank scheme. In the pilot study, we used the same learner corpus as in the pilot study for the error annotation scheme.

As a result of the pilot study, we found that the Penn Treebank tag set sufficed in most cases except for errors which learners made. Considering this, we determined a basic rule as follows: "Use the Penn Treebank tag set and preserve the original texts as much as possible." To handle such errors, we made several modifications and added two new POS tags (CE and UK) and another two for chunking (XP and PH), which are described below.

A major modification concerns errors in mechanics such as *Tonight,we* and *beautifulhouse* as already explained in Sect. 2. We use the symbol "-" to annotate such cases. For instance, the above two examples are annotated as follows: *Tonight,we/NN-,-PRP* and *beautifulhouse/JJ-NN*. Note that each POS tag is hyphenated. It can also be used for annotating chunks in the same manner. For instance, *Tonight,we* is annotated as *[NP-PH-NP Tonight,we/NN-,-PRP]*. Here, the tag *PH* stands for ϕ chunk label and denotes tokens which are not normally chunked (cf., [NP Tonight/NN] ,/, [NP we/PRP]).

Another major modification was required to handle grammatical errors. Essentially, POS/chunking tags are assigned according to the surface information of the word in question regardless of the existence of any errors. For example, *There is apples.* is annotated as *[NP There/EX] [VP is/VBZ] [NP apples/NNS] ./.* Additionally, we define the CE[4] tag to annotate errors in which learners use a word with a POS which is not allowed such as in *I don't success cooking.* The CE tag encodes a POS which is obtained from the surface information together with the POS which would have been assigned to the word if it were not for the error. For instance, the above example is tagged as *I don't success/CE:NN:VB cooking.* In this format, the second and third POSs are separated by ":" which denotes the POS which is obtained from the surface information and the POS which would be assigned

[3]The learner corpus had been created before this reported work started. Learners wrote their essays on paper. Unfortunately, this learner corpus cannot be made available to the public since the copyrights were not transferred to us.

[4]CE stands for cognitive error.

to the word without an error. The user can select either POS depending on his or her purposes. Note that the CE tag is compatible with the basic annotation scheme because we can retrieve the basic annotation by extracting only the second element (i.e., success/NN). If the tag is unknown because of grammatical errors or other phenomena, UK and XP[5] are used for POS and chunking, respectively.

For spelling errors, the corresponding POS and chunking tag are assigned to mistakenly spelled words if the correct forms can be guessed (e.g., [NP sird/JJ year/NN]); otherwise UK and XP are used.

4 The Corpus

We carried out a learner corpus creation project using the described method. Twenty six Japanese college students participated in the project. At the beginning, we had the students or their parents sign a conventional paper-based copyright transfer form. After that, they did the writing exercise described in Sect. 3 once or twice a week over three months. During that time, they were assigned ten topics, which were determined based on a writing textbook (Okihara, 1985). As described in Sect. 3, they used a blog system to write, submit, and rewrite their essays. Through out the exercises, they did not have access to the others' essays and their own previous essays.

As a result, 233 essays were collected; Table 3 shows the statistics on the collected essays. It turned out that the learners had no difficulties in using the blog system and seemed to focus on writing. Out of the 26 participants, 22 completed the 10 assignments while one student quit before the exercises started.

We annotated the grammatical errors of all 233 essays. Two persons were involved in the annotation. After the annotation, another person checked the annotation results; differences in error annotation were resolved by consulting the first two. The error annotation scheme was found to work well on them. The error-annotated essays can be used for evaluating error detection/correction methods.

Number of essays	233
Number of writers	25
Number of sentences	3,199
Number of words	25,537

Table 3: Statistics on the learner corpus.

[5]UK and XP stand for unknown and X phrase, respectively.

For POS/chunking annotation, we chose 170 essays out of 233. We annotated them using our POS/chunking scheme; hereafter, the 170 essays will be referred to as the shallow-parsed corpus.

5 Using the Corpus and Discussion

5.1 POS Tagging

The 170 essays in the shallow-parsed corpus was used for evaluating existing POS-tagging techniques on texts written by learners. It consisted of 2,411 sentences and 22,452 tokens.

HMM-based and CRF-based POS taggers were tested on the shallow-parsed corpus. The former was implemented using tri-grams by the author. It was trained on a corpus consisting of English learning materials (213,017 tokens). The latter was CRFTagger[6], which was trained on the WSJ corpus. Both use the Penn Treebank POS tag set.

The performance was evaluated using accuracy defined by

$$\frac{\text{number of tokens correctly POS-tagged}}{\text{number of tokens}}. \quad (1)$$

If the number of tokens in a sentence was different in the human annotation and the system output, the sentence was excluded from the calculation. This discrepancy sometimes occurred because the tokenization of the system sometimes differed from that of the human annotators. As a result, 19 and 126 sentences (215 and 1,352 tokens) were excluded from the evaluation in the HMM-based and CRF-based POS taggers, respectively.

Table 4 shows the results. The second column corresponds to accuracies on a native-speaker corpus (sect. 00 of the WSJ corpus). The third column corresponds to accuracies on the learner corpus.

As shown in Table 4, the CRF-based POS tagger suffers a decrease in accuracy as expected. Interestingly, the HMM-based POS tagger performed better on the learner corpus. This is perhaps because it

[6]"CRFTagger: CRF English POS Tagger," Xuan-Hieu Phan, `http://crftagger.sourceforge.net/`, 2006.

was trained on a corpus consisting of English learning materials whose distribution of vocabulary was expected to be relatively similar to that of the learner corpus. By contrast, it did not perform well on the native-speaker corpus because the size of the training corpus was relatively small and the distribution of vocabulary was not similar, and thus unknown words often appeared. This implies that selecting appropriate texts as a training corpus may improve the performance.

Table 5 shows the top five POSs mistakenly tagged as other POSs. An obvious cause of mistakes in both taggers is that they inevitably make errors in the POSs that are not defined in the Penn Treebank tag set, that is, UK and CE. A closer look at the tagging results revealed that phenomena which were common to the writing of learners were major causes of other mistakes. Errors in capitalization partly explain why the taggers made so many mistakes in NN (singular nouns). They often identified erroneously capitalized common nouns as proper nouns as in *This Summer/NNP Vacation/NNP*. Spelling errors affected the taggers in the same way. Grammatical errors also caused confusion between POSs. For instance, omission of a certain word often caused confusion between a verb and an adjective as in *I frightened/VBD.* which should be *I (was) frightened/JJ.* Another interesting case is expressions that learners overuse (e.g., *and/CC so/RB on/RB* and *so/JJ so/JJ*). Such phrases are not erroneous but are relatively infrequent in native-speaker corpora. Therefore, the taggers tended to identify their POSs according to the surface information on the tokens themselves when such phrases appeared in the learner corpus (e.g., *and/CC so/RB* <u>*on/IN*</u> and <u>*so/RB so/RB*</u>). We should be aware that tokenization is also problematic although failures in tokenization were excluded from the accuracies.

The influence of the decrease in accuracy on other NLP tasks is expected to be task and/or method dependent. Methods that directly use or handle sequences of POSs are likely to suffer from it. An example is the error detection method (Chodorow and Leacock, 2000), which identifies unnatural sequences of POSs as grammatical errors in the writing of learners. As just discussed above, existing techniques often fail in sequences of POSs that have a grammatical error. For instance, an existing POS tagger likely tags the sentence *I frightened.* as *I/PRP frightened/VBD ./.* as we have just seen, and in turn the error detection method cannot identify it as an error because the sequence *PRP VBD* is not unnatural; it would correctly detect it if the sentence were correctly tagged as *I/PRP frightened/JJ ./.* For the same reason, the decrease in accuracy may affect the methods (Aarts and Granger, 1998; Granger, 1998; Tono, 2000) for extracting interesting sequences of POSs from learner corpora; for example, BOS[7] PRP JJ is an interesting sequence but is never extracted unless the phrase is correctly POS-tagged. It requires further investigation to reveal how much impact the decrease has on these methods. By contrast, error detection/correction methods based on the bag-of-word features (or feature vectors) are expected to suffer less from it since mistakenly POS-tagged tokens are only one of the features. At the same time, we should notice that if the target errors are in the tokens that are mistakenly POS-tagged, the detection will likely fail (e.g., verbs should be correctly identified in tense error detection).

In addition to the above evaluation, we attempted to improve the POS taggers using the transformation-based POS-tagging technique (Brill, 1994). In the technique, transformation rules are obtained by comparing the output of a POS tagger and the human annotation so that the differences between the two are reduced. We used the shallow-

Method	Native Corpus	Learner Corpus
CRF	0.970	0.932
HMM	0.887	0.926

Table 4: POS-tagging accuracy.

HMM		CRF	
POS	Freq.	POS	Freq.
NN	259	NN	215
VBP	247	RB	166
RB	163	CE	144
CE	150	JJ	140
JJ	108	FW	86

Table 5: Top five POSs mistakenly tagged.

[7] BOS denotes a beginning of a sentence.

Method	Original	Improved
CRF	0.932	0.934
HMM	0.926	0.933

Table 6: Improvement obtained by transformation.

parsed corpus as a test corpus and the other manually POS-tagged corpus created in the pilot study described in Subsect. 3.2.1 as a training corpus. We used POS-based and word-based transformations as Brill (1994) described.

Table 6 shows the improvements together with the original accuracies. Table 6 reveals that even the simple application of Brill's technique achieves a slight improvement in both taggers. Designing the templates of the transformation for learner corpora may achieve further improvement.

5.2 Head Noun Identification

In the evaluation of chunking, we focus on head noun identification. Head noun identification often plays an important role in error detection/correction. For example, it is crucial to identify head nouns to detect errors in article and number.

We again used the shallow-parsed corpus as a test corpus. The essays contained 3,589 head nouns. We implemented an HMM-based chunker using 5-grams whose input is a sequence of POSs, which was obtained by the HMM-based POS tagger described in the previous subsection. The chunker was trained on the same corpus as the HMM-based POS tagger. The performance was evaluated by recall and precision defined by

$$\frac{\text{number of head nouns correctly identified}}{\text{number of head nouns}} \quad (2)$$

and

$$\frac{\text{number of head nouns correctly identified}}{\text{number of tokens identified as head noun}}, \quad (3)$$

respectively.

Table 7 shows the results. To our surprise, the chunker performed better than we had expected. A possible reason for this is that sentences written by learners of English tend to be shorter and simpler in terms of their structure.

The results in Table 7 also enable us to quantitatively estimate expected improvement in error detection/correction which is achieved by improving chunking. To see this, let us define the following symbols: r: Recall of head noun identification, R: recall of error detection without chunking error, $\hat{R}$ recall of error detection with chunking error. R and $\hat{R}$ are interpreted as the true recall of error detection and its observed value when chunking error exists, respectively. Here, note that $\hat{R}$ can be expressed as $\hat{R} = rR$. For instance, according to Han et al. (2006), their method achieves a recall of 0.40 (i.e., $\hat{R} = 0.4$), and thus $R = 0.44$ assuming that chunking errors exist and recall of head noun identification is $r = 0.90$ just as in this evaluation. Improving r to $r = 0.97$ would achieve $R = 0.43$ without any modification to the error detection method. Precision can also be estimated in a similar manner although it requires a more complicated calculation.

6 Conclusions

In this paper, we discussed the difficulties inherent in learner corpus creation and a method for efficiently creating a learner corpus. We described the manually error-annotated and shallow-parsed learner corpus which was created using this method. We also showed its usefulness in developing and evaluating POS taggers and chunkers. We believe that publishing this corpus will give researchers a common development and test set for developing related NLP techniques including error detection/correction and POS-tagging/chunking, which will facilitate further research in these areas.

A Error tag set

This is the list of our error tag set. It is based on the NICT JLE tag set (Izumi et al., 2005).

- n: noun
  - num: number
  - lxc: lexis
  - o: other
- v: verb
  - agr: agreement

Recall	Precision
0.903	0.907

Table 7: Performance on head noun identification.

  - tns: tense
  - lxc: lexis
  - o: other
- mo: auxiliary verb
- aj: adjective
  - lxc: lexis
  - o: other
- av: adverb
- prp: preposition
  - lxc: lexis
  - o: other
- at: article
- pn: pronoun
- con: conjunction
- rel: relative clause
- itr: interrogative
- olxc: errors in lexis in more than two words
- ord: word order
- uk: unknown error
- f: fragment error

References

Jan Aarts and Sylviane Granger. 1998. *Tag sequences in learner corpora: a key to interlanguage grammar and discourse*. Longman Pub Group, London.

Eric Brill. 1994. Some advances in transformation-based part of speech tagging. In *Proc. of 12th National Conference on Artificial Intelligence*, pages 722–727.

Martin Chodorow and Claudia Leacock. 2000. An unsupervised method for detecting grammatical errors. In *Proc. of 1st Meeting of the North America Chapter of ACL*, pages 140–147.

Martin Chodorow, Joel R. Tetreault, and Na-Rae Han. 2007. Detection of grammatical errors involving prepositions. In *Proc. of 4th ACL-SIGSEM Workshop on Prepositions*, pages 25–30.

Rachele De Felice and Stephen G. Pulman. 2008. A classifier-based approach to preposition and determiner error correction in L2 English. In *Proc. of 22nd International Conference on Computational Linguistics*, pages 169–176.

Sylviane Granger, Estelle Dagneaux, Fanny Meunier, and Magali Paquot. 2009. *International Corpus of Learner English v2*. Presses universitaires de Louvain.

Sylviane Granger. 1998. Prefabricated patterns in advanced EFL writing: collocations and formulae. In A. P. Cowie, editor, *Phraseology: theory, analysis, and application*, pages 145–160. Clarendon Press.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2004. Detecting errors in English article usage with a maximum entropy classifier trained on a large, diverse corpus. In *Proc. of 4th International Conference on Language Resources and Evaluation*, pages 1625–1628.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2006. Detecting errors in English article usage by non-native speakers. *Natural Language Engineering*, 12(2):115–129.

Emi Izumi, Toyomi Saiga, Thepchai Supnithi, Kiyotaka Uchimoto, and Hitoshi Isahara. 2003a. The development of the spoken corpus of Japanese learner English and the applications in collaboration with NLP techniques. In *Proc. of the Corpus Linguistics 2003 Conference*, pages 359–366.

Emi Izumi, Kiyotaka Uchimoto, Toyomi Saiga, Thepchai Supnithi, and Hitoshi Isahara. 2003b. Automatic error detection in the Japanese learners' English spoken data. In *Proc. of 41st Annual Meeting of ACL*, pages 145–148.

Emi Izumi, Kiyotaka Uchimoto, and Hitoshi Isahara. 2005. Error annotation for corpus of Japanese learner English. In *Proc. of 6th International Workshop on Linguistically Annotated Corpora*, pages 71–80.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. 2010. *Automated Grammatical Error Detection for Language Learners*. Morgan & Claypool, San Rafael.

John Lee and Stephanie Seneff. 2008. Correcting misuse of verb forms. In *Proc. of 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technology Conference*, pages 174–182.

Ryo Nagata and Kazuhide Nakatani. 2010. Evaluating performance of grammatical error detection to maximize learning effect. In *Proc. of 23rd International Conference on Computational Linguistics, poster volume*, pages 894–900.

Ryo Nagata, Fumito Masui, Atsuo Kawai, and Naoki Isu. 2004. Recognizing article errors based on the three

head words. In *Proc. of Cognition and Exploratory Learning in Digital Age*, pages 184–191.

Ryo Nagata, Takahiro Wakana, Fumito Masui, Atsuo Kawai, and Naoki Isu. 2005. Detecting article errors based on the mass count distinction. In *Proc. of 2nd International Joint Conference on Natural Language Processing*, pages 815–826.

Ryo Nagata, Atsuo Kawai, Koichiro Morihiro, and Naoki Isu. 2006. A feedback-augmented method for detecting errors in the writing of learners of English. In *Proc. of 44th Annual Meeting of ACL*, pages 241–248.

Katsuaki Okihara. 1985. *English writing (in Japanese)*. Taishukan, Tokyo.

Alla Rozovskaya and Dan Roth. 2010a. Annotating ESL errors: Challenges and rewords. In *Proc. of NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 28–36.

Alla Rozovskaya and Dan Roth. 2010b. Training paradigms for correcting errors in grammar and usage. In *Proc. of 2010 Annual Conference of the North American Chapter of the ACL*, pages 154–162.

Joel Tetreault, Elena Filatova, and Martin Chodorow. 2010a. Rethinking grammatical error annotation and evaluation with the Amazon Mechanical Turk. In *Proc. of NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 45–48.

Joel Tetreault, Jennifer Foster, and Martin Chodorow. 2010b. Using parse features for preposition selection and error detection. In *Proc. of 48nd Annual Meeting of the Association for Computational Linguistics Short Papers*, pages 353–358.

Yukio Tono. 2000. A corpus-based analysis of interlanguage development: analysing POS tag sequences of EFL learner corpora. In *Practical Applications in Language Corpora*, pages 123–132.

Crowdsourcing Translation: Professional Quality from Non-Professionals

Omar F. Zaidan and **Chris Callison-Burch**
Dept. of Computer Science, Johns Hopkins University
Baltimore, MD 21218, USA
{ozaidan,ccb}@cs.jhu.edu

Abstract

Naively collecting translations by crowdsourcing the task to non-professional translators yields disfluent, low-quality results if no quality control is exercised. We demonstrate a variety of mechanisms that increase the translation quality to near professional levels. Specifically, we solicit redundant translations and edits to them, and automatically select the best output among them. We propose a set of features that model both the translations and the translators, such as country of residence, LM perplexity of the translation, edit rate from the other translations, and (optionally) calibration against professional translators. Using these features to score the collected translations, we are able to discriminate between acceptable and unacceptable translations. We recreate the NIST 2009 Urdu-to-English evaluation set with Mechanical Turk, and quantitatively show that our models are able to select translations within the range of quality that we expect from professional translators. The total cost is more than an order of magnitude lower than professional translation.

1 Introduction

In natural language processing research, translations are most often used in statistical machine translation (SMT), where systems are trained using bilingual sentence-aligned parallel corpora. SMT owes its existence to data like the Canadian Hansards (which by law must be published in both French and English). SMT can be applied to any language pair for which there is sufficient data, and it has been shown to produce state-of-the-art results for language pairs like Arabic–English, where there is ample data. However, large bilingual parallel corpora exist for relatively few languages pairs.

There are various options for creating new training resources for new language pairs. These include harvesting the web for translations or comparable corpora (Resnik and Smith, 2003; Munteanu and Marcu, 2005; Smith et al., 2010; Uszkoreit et al., 2010), improving SMT models so that they are better suited to the low resource setting (Al-Onaizan et al., 2002; Probst et al., 2002; Oard et al., 2003; Niessen and Ney, 2004), or designing models that are capable of learning translations from monolingual corpora (Rapp, 1995; Fung and Yee, 1998; Schafer and Yarowsky, 2002; Haghighi et al., 2008). Relatively little consideration is given to the idea of simply hiring translators to create parallel data, because it would seem to be prohibitively expensive. For example, Germann (2001) estimated the cost of hiring professional translators to create a Tamil-English corpus at $0.36/word. At that rate, translating enough data to build even a small parallel corpus like the LDC's 1.5 million word Urdu–English corpus would exceed half a million dollars.

In this paper we examine the idea of creating low cost translations via crowdscouring. We use Amazon's Mechanical Turk to hire a large group of non-professional translators, and have them recreate an Urdu–English evaluation set at a fraction of the cost of professional translators. The original dataset already has professionally-produced reference translations, which allows us to objectively and quantitatively compare the quality of professional and non-professional translations. Although many of the individual non-expert translators produce low-quality, disfluent translations, we show that it is possible to

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1220–1229,
Portland, Oregon, June 19-24, 2011.

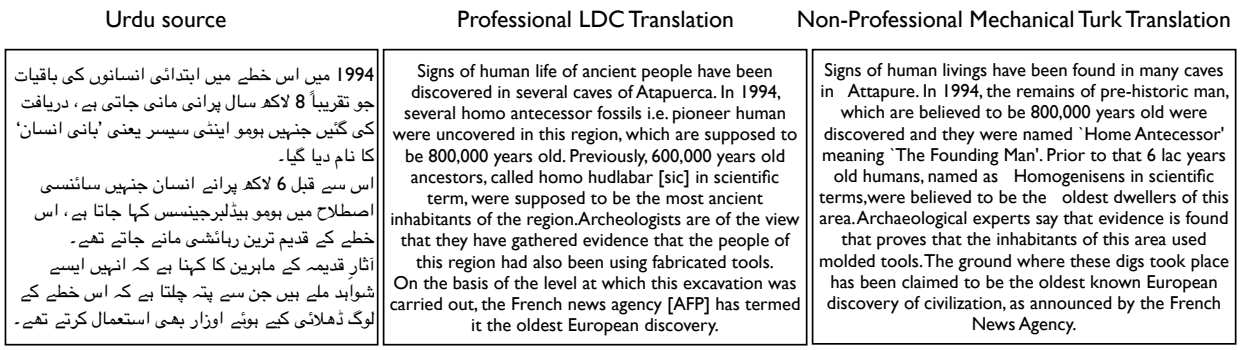


Figure 1: A comparison of professional translations provided by the LDC to non-professional translations created on Mechanical Turk.

get high quality translations in aggregate by soliciting multiple translations, redundantly editing them, and then selecting the best of the bunch.

To select the best translation, we use a machine-learning-inspired approach that assigns a score to each translation we collect. The scores discriminate acceptable translations from those that are not (and competent translators from those who are not). The scoring is based on a set of informative, intuitive, and easy-to-compute features. These include country of residence, number of years speaking English, LM perplexity of the translation, edit rate from the other translations, and (optionally) calibration against professional translators, with the weights set using a small set of gold standard data from professional translators.

2 Crowdsourcing Translation to Non-Professionals

To collect crowdsourced translations, we use Amazon's Mechanical Turk (MTurk), an online marketplace designed to pay people small sums of money to complete *Human Intelligence Tasks* (or HITs) – tasks that are difficult for computers but easy for people. Example HITs range from labeling images to moderating blog comments to providing feedback on relevance of results for search queries. Anyone with an Amazon account can either submit HITs or work on HITs that were submitted by others. Workers are referred to as "Turkers", and designers of HITs as "Requesters." A Requester specifies the reward to be paid for each completed item, sometimes as low as $0.01. Turkers are free to select whichever HITs interest them, and to bypass HITs they find uninteresting or which they deem pay too little.

The advantages of Mechanical Turk include:

- zero overhead for hiring workers
- a large, low-cost labor force
- easy micropayment system
- short turnaround time, as tasks get completed in parallel by many individuals
- access to foreign markets with native speakers of many rare languages

One downside is that Amazon does not provide any personal information about Turkers. (Each Turker is identifiable only through an anonymous ID like A23KO2TP7I4KK2.) In particular, no information is available about a worker's educational background, skills, or even native language(s). This makes it difficult to determine if a Turker is qualified to complete a translation task.

Therefore, soliciting translations from anonymous non-professionals carries a significant risk of poor translation quality. Whereas hiring a professional translator ensures a degree of quality and care, it is not very difficult to find bad translations provided by Turkers. One Urdu headline, professionally translated as *Barack Obama: America Will Adopt a New Iran Strategy*, was rendered disfluently by a Turker as *Barak* ***Obam*** *will* ***do a new policy*** *with Iran*. Another translated it with snarky sarcasm: *Barak Obama and America* ***weave new evil strategies*** *against Iran*. Figure 1 gives more typical translation examples. The translations often reflect non-native English, but are generally done conscientiously (in spite of the relatively small payment).

To improve the accuracy of noisy labels from non-experts, most existing quality control mechanisms

employ some form of voting, assuming a discrete set of possible labels. This is not the case for translations, where the 'labels' are full sentences. When dealing with such a structured output, the space of possible outputs is diverse and complex. We therefore need a different approach for quality control. That is precisely the focus of this work: to propose, and evaluate, such quality control mechanisms.

In the next section, we discuss reproducing the Urdu-to-English 2009 NIST evaluation set. We then describe a principled approach to discriminate good translations from bad ones, given a set of redundant translations for the same source sentence.

3 Datasets

3.1 The Urdu-to-English 2009 NIST Evaluation Set

We translated the Urdu side of the Urdu–English test set of the 2009 NIST MT Evaluation Workshop. The set consists of 1,792 Urdu sentences from a variety of news and online sources. The set includes four different reference translations for each source sentence, produced by professional translation agencies. NIST contracted the LDC to oversee the translation process and perform quality control.

This particular dataset, with its multiple reference translations, is very useful because we can measure the quality range for professional translators, which gives us an idea of whether or not the crowdsourced translations approach the quality of a professional translator.

3.2 Translation HIT design

We solicited English translations for the Urdu sentences in the NIST dataset. Amazon has enabled payments in rupees, which has attracted a large demographic of workers from India (Ipeirotis, 2010). Although it does not yet have s direct payment in Pakistan's local currency, we found that a large contingent of our workers are located in Pakistan.

Our HIT involved showing the worker a sequence of Urdu sentences, and asking them to provide an English translation for each one. The screen also included a brief set of instructions, and a short questionnaire section. The reward was set at $0.10 per translation, or roughly $0.005 per word.

In our first collection effort, we solicited only one translation per Urdu sentence. After confirming that the task is feasible due to the large pool of workers willing and able to provide translations, we carried out a second collection effort, this time soliciting *three* translations per Urdu sentence (from three distinct translators). The interface was also slightly modified, in the following ways:

- Instead of asking Turkers to translate a full document (as in our first pass), we instead split the data set into groups of 10 sentences per HIT.
- We converted the Urdu sentences into images so that Turkers could not cheat by copying-and-pasting the Urdu text into an MT system.
- We collected information about each worker's geographic location, using a JavaScript plugin.

The translations from the first pass were of noticeably low quality, most likely due to Turkers using automatic translation systems. That is why we used images instead of text in our second pass, which yielded significant improvements. That said, we do not discard the translations from the first pass, and we do include them in our experiments.

3.3 Post-editing and Ranking HITs

In addition to collecting four translations per source sentence, we also collected **post-edited** versions of the translations, as well as **ranking judgments** about their quality.

Figure 2 gives examples of the unedited translations that we collected in the translation pass. These typically contain many simple mistakes like misspellings, typos, and awkward word choice. We posted another MTurk task where we asked workers to edit the translations into more fluent and grammatical sentences. We restrict the task to US-based workers to increase the likelihood that they would be native English speakers.

We also asked US-based Turkers to rank the translations. We presented the translations in groups of four, and the annotator's task was to rank the sentences by fluency, from best to worst (allowing ties).

We collected redundant annotations in these two tasks as well. Each translation is edited three times (by three distinct editors). We solicited only one edit per translation from our first pass translation effort. So, in total, we had 10 post-edited translations for

Avoiding dieting to prevent from flu	abstention from dieting in order to avoid Flu	Abstain from decrease eating in order to escape from flue	In order to be safer from flu quit dieting
This research of American scientists came in front after experimenting on mice.	This research from the American Scientists have come up after the experiments on rats.	This research of American scientists was shown after many experiments on mouses.	According to the American Scientist this research has come out after much experimentations on rats.
Experiments proved that mice on a lower calorie diet had comparatively less ability to fight the flu virus.	in has been proven from experiments that rats put on diet with less calories had less ability to resist the Flu virus.	It was proved by experiments the low calories eaters mouses had low defending power for flue in ratio.	Experimentaions have proved that those rats on less calories diet have developed a tendency of not overcoming the flu virus.
research has proven this old myth wrong that its better to fast during fever.	Research disproved the old axiom that " It is better to fast during fever"	The research proved this old talk that decrease eating is useful in fever.	This Research has proved the very old saying wrong that it is good to starve while in fever.

Figure 2: We redundantly translate each source sentence by soliciting multiple translations from different Turkers. These translations are put through a subsequent editing set, where multiple edited versions are produced. We select the best translation from the set using features that predict the quality of each translation and each translator.

each source sentence (plus the four original translations). In the ranking task, we collected judgments from five distinct workers for each translation group.

3.4 Data Collection Cost

We paid a reward of \$0.10 to translate a sentence, \$0.25 to edit a set of ten sentences, and \$0.06 to rank a set of four translation groups. Therefore, we had the following costs:

- Translation cost: \$716.80
- Editing cost: \$447.50
- Ranking cost: \$134.40

(If not done redundantly, those values would be \$179.20, \$44.75, and \$26.88, respectively.)

Adding Amazon's 10% fee, this brings the grand total to under \$1,500, spent to collect 7,000+ translations, 17,000+ edited translations, and 35,000+ rank labels.[1] We also use about 10% of the existing professional references in most of our experiments (see 4.2 and 4.3). If we estimate the cost at \$0.30/word, that would roughly be an additional \$1,000.

3.5 MTurk Participation

52 different Turkers took part in the translation task, each translating 138 sentences on average. In the editing task, 320 Turkers participated, averaging 56 sentences each. In the ranking task, 245 Turkers participated, averaging 9.1 HITs each, or 146 rank labels (since each ranking HIT involved judging 16 translations, in groups of four).

[1]Data URL: `www.cs.jhu.edu/~ozaidan/RCLMT`.

4 Quality Control Model

Our approach to building a translation set from the available data is to select, for each Urdu sentence, the one translation that our model believes to be the best out of the available translations. We evaluate various selection techniques by comparing the selected Turker translations against existing professionally-produced translations. The more the selected translations resemble the professional translations, the higher the quality.

4.1 Features Used to Select Best Translations

Our model selects one of the 14 English options generated by Turkers. For a source sentence s_i, our model assigns a score to each sentence in the set of available translations $\{t_{i,1}, ...t_{i,14}\}$. The chosen translation is the highest scoring translation:

$$tr(s_i) = tr_{i,j^*} \text{ s.t. } j^* = \underset{j}{\operatorname{argmax}}\, score(t_{i,j}) \quad (1)$$

where $score(.)$ is the dot product:

$$score(t_{i,j}) \stackrel{\text{def}}{=} \vec{w} \cdot \vec{f}(t_{i,j}) \quad (2)$$

Here, $\vec{w}$ is the model's weight vector (tuned as described below in 4.2), and $\vec{f}$ is a translation's corresponding feature vector. Each feature is a function computed from the English sentence string, the Urdu sentence string, the workers (translators, editors, and rankers), and/or the rank labels. We use 21 features, categorized into the following three sets.

Sentence-level (6 features). Most of the Turkers performing our task were native Urdu speakers whose second language was English, and they do not always produce natural-sounding English sentences. Therefore, the first set of features attempt to discriminate good English sentences from bad ones.

- Language model features: each sentence is assigned a log probability and per-word perplexity score, using a 5-gram language model trained on the English Gigaword corpus.
- Sentence length features: a good translation tends to be comparable in length to the source sentence, whereas an overly short or long translation is probably bad. We add two features that are the ratios of the two lengths (one penalizes short sentences and one penalizes long ones).
- Web n-gram match percentage: we assign a score to each sentence based on the percentage of the n-grams (up to length 5) in the translation that exist in the Google N-Gram Database.
- Web n-gram geometric average: we calculate the average over the different n-gram match percentages (similar to the way BLEU is computed). We add three features corresponding to max n-gram lengths of 3, 4, and 5.
- Edit rate to other translations: a bad translation is likely not to be very similar to other translations, since there are many more ways a translation can be bad than for it to be good. So, we compute the average edit rate distance from the other translations (using the TER metric).

Worker-level (12 features). We add *worker*-level features that evaluate a translation based on *who* provided it.

- Aggregate features: for each sentence-level feature above, we have a corresponding feature computed over *all* of that worker's translations.
- Language abilities: we ask workers to provide information about their language abilities. We have a binary feature indicating whether Urdu is their native language, and a feature for how long they have spoken it. We add a pair of equivalent features for English.
- Worker location: two binary features reflect a worker's location, one to indicate if they are located in Pakistan, and one to indicate if they are located in India.

Ranking (3 features). The third set of features is based on the ranking labels we collected (see 3.3).

- Average rank: the average of the five rank labels provided for this translation.
- Is-Best percentage: how often the translation was top-ranked among the four translations.
- Is-Better percentage: how often the translation was judged as the better translation, over all pairwise comparisons extracted from the ranks.

Other features (not investigated here) could include source-target information, such as translation model scores or the number of source words translated correctly according to a bilingual dictionary.

4.2 Parameter Tuning

Once features are computed for the sentences, we must set the model's weight vector $\vec{w}$. Naturally, the weights should be chosen so that good translations get high scores, and bad translations get low scores. We optimize translation quality against a small subset (10%) of reference (professional) translations.

To tune the weight vector, we use the linear search method of Och (2003), which is the basis of Minimum Error Rate Training (MERT). MERT is an iterative algorithm used to tune parameters of an MT system, which operates by iteratively generating new candidate translations and adjusting the weights to give good translations a high score, then regenerating new candidates based on the updated weights, etc. In our work, the set of candidate translations is *fixed* (the 14 English sentences for each source sentence), and therefore iterating the procedure is not applicable. We use the Z-MERT software package (Zaidan, 2009) to perform the search.

4.3 The *Worker Calibration* Feature

Since we use a small portion of the reference translations to perform weight tuning, we can also use that data to compute another worker-specific feature. Namely, we can evaluate the competency of each worker by scoring their translations against the reference translations. We then use that feature for every translation given by that worker. The intuition

is that workers known to produce good translations are likely to continue to produce good translations, and the opposite is likely true as well.

4.4 Evaluation Strategy

To measure the quality of the translations, we make use of the existing professional translations. Since we have *four* professional translation sets, we can calculate the BLEU score (Papineni et al., 2002) for one professional translator P_1 using the other three $P_{2,3,4}$ as a reference set. We repeat the process four times, scoring each professional translator against the others, to calculate the expected range of professional quality translation. We can see how a translation set T (chosen by our model) compares to this range by calculating T's BLEU scores against the same four sets of three reference translations. We will evaluate different strategies for selecting such a set T, and see how much each improves on the BLEU score, compared to randomly picking from among the Turker translations.

We also evaluate Turker translation quality by using them as reference sets to score various submissions to the NIST MT evaluation. Specifically, we measure the correlation (using Pearson's r) between BLEU scores of MT systems measured against non-professional translations, and BLEU scores measured against professional translations. Since the main purpose of the NIST dataset was to compare MT systems against each other, this is a more direct fitness-for-task measure. We chose the middle 6 systems (in terms of performance) submitted to the NIST evaluation, out of 12, as those systems were fairly close to each other, with less than 2 BLEU points separating them.[2]

5 Experimental Results

We establish the performance of professional translators, calculate oracle upper bounds on Turker translation quality, and carry out a set of experiments that demonstrate the effectiveness of our model and that determine which features are most helpful.

Each number reported in this section is an average of four numbers, corresponding to the four possible ways of choosing 3 of the 4 reference sets. Furthermore, each of those 4 numbers is itself based on a five-fold cross validation, where 80% of the data is used to compute feature values, and 20% used for evaluation. The 80% portion is used to compute the aggregate worker-level features. For the worker calibration feature, we utilize the references for 10% of the data (which is within the 80% portion).

5.1 Translation Quality: BLEU Scores Compared to Professionals

We first evaluated the reference sets against each other, in order to quantify the concept of "professional quality". On average, evaluating one reference set against the other three gives a BLEU score of 42.38 (Figure 3). A Turker set of translations scores 28.13 on average, which highlights the loss in quality when collecting translations from amateurs. To make the gap clearer, the output of a state-of-the-art machine translation system (the syntax-based variant of Joshua; Li et al. (2010)) achieves a score of 26.91, a mere 1.22 worse than the Turkers.

We perform two oracle experiments to determine if there exist high-quality Turker translations in the first place. The first oracle operates on the segment level: for each source segment, choose from the four translations the one that scores highest against the reference sentence. The second oracle operates on the worker level: for each source segment, choose from the four translations the one provided by the worker whose translations (over all sentences) score the highest. The two oracles achieve BLEU scores of 43.75 and 40.64, respectively – well within the range of professional translators.

We examined two voting-inspired methods, since taking a majority vote usually works well when dealing with MTurk data. The first selects the translation with the minimum average TER (Snover et al., 2006) against the other three translations, since that would be a 'consensus' translation. The second method selects the translation that received the best average rank, using the rank labels assigned by other Turkers (see 3.3). These approaches achieve BLEU scores of 34.41 and 36.64, respectively.

The main set of experiments evaluated the features from 4.1 and 4.3. We applied our approach using each of the four feature types: sentence features, Turker features, rank features, and the cali-

[2]Using all 12 systems artificially inflates correlation, due to the vast differences between the systems. For instance, the top system outperforms the bottom system by **15** BLEU points!

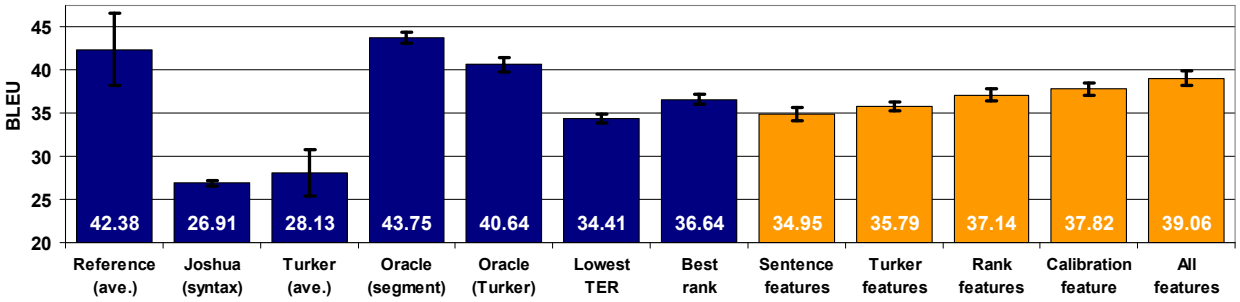


Figure 3: BLEU scores for different selection methods, measured against the reference sets. Each score is an average of four BLEU scores, each calculated against three LDC reference translations. The five right-most bars are colored in orange to indicate selection over a set that includes both original translations as well as edited versions of them.

bration feature. That yielded BLEU scores ranging from 34.95 to 37.82. With all features combined, we achieve a higher score of 39.06, which is within the range of scores for the professional translators.

5.2 Fitness for a Task: Correlation With Professionals When Ranking MT Systems

We evaluated the selection methods by measuring correlation with the references, in terms of BLEU scores assigned to outputs of MT systems. The results, in Table 1, tell a fairly similar story as evaluating with BLEU: references and oracles naturally perform very well, and the loss in quality when selecting arbitrary Turker translations is largely eliminated using our selection strategy.

Interestingly, when using the Joshua output as a reference set, the performance is quite abysmal. Even though its BLEU score is comparable to the Turker translations, it cannot be used to distinguish closely matched MT systems from each other.[3]

6 Analysis

The oracles indicate that there is usually an acceptable translation from the Turkers for any given sentence. Since the oracles select from a small group of only 4 translations per source segment, they are not overly optimistic, and rather reflect the true potential of the collected translations.

The results indicate that, although some features are more useful than others, much of the benefit from combining all the features can be obtained from any one set of features, with the benefit of adding more features being somewhat orthogonal.

[3]It should be noted that the Joshua system was not one of the six MT systems we scored in the correlation experiments.

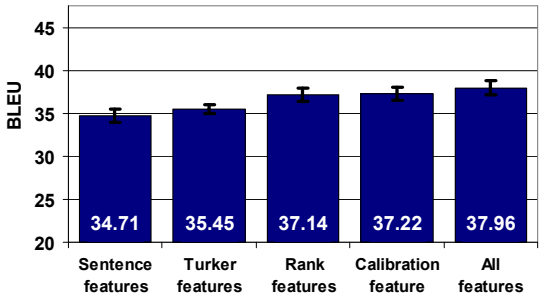


Figure 4: BLEU scores for the five right-most setups from Figure 3, constrained over the original translations.

Finally, we performed a series of experiments exploring the calibration feature, varying the amount of gold-standard references from 10% all the way up to 80%. As expected, the performance improved as more references were used to calibrate the translators (Figure 5). What's particularly important about this experiment is that it shows the added benefit of the other features: We would have to use 30%–40% of the references to get the same benefit obtained from combining the non-calibration features and only 10% for the calibration feature (dashed line in the Figure; BLEU = 39.06).

6.1 Cost Reduction

While the combined cost of our data collection effort ($2,500; see 3.4) is quite low considering the amount of collected data, it would be more attractive if the cost could be reduced further without losing much in translation quality. To that end, we investigated lowering cost along two dimensions: eliminating the need for professional translations, and decreasing the amount of edited translations.

Selection Method	Pearson's r^2
Reference (ave.)	0.81 ± 0.07
Joshua (syntax)	0.08 ± 0.09
Turker (ave.)	0.60 ± 0.17
Oracle (segment)	0.81 ± 0.09
Oracle (Turker)	0.79 ± 0.10
Lowest TER	0.50 ± 0.26
Best rank	0.74 ± 0.17
Sentence features	0.56 ± 0.21
Turker features	0.59 ± 0.19
Rank features	0.75 ± 0.14
Calibration feature	0.76 ± 0.13
All features	0.77 ± 0.11

Table 1: Correlation ($\pm$ std. dev.) for different selection methods, compared against the reference sets.

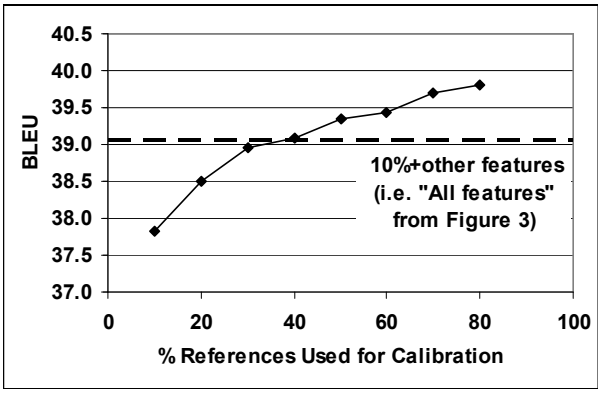


Figure 5: The effect of varying the amount of calibration data (and using only the calibration feature). The 10% point (BLEU = 37.82) and the dashed line (BLEU = 39.06) correspond to the two right-most bars of Figure 3.

The professional translations are used in our approach for computing the worker calibration feature (subsection 4.3) and for tuning the weights of the other features. We use a relatively small amount for this purpose, but we investigate a different setup whereby no professional translations are used at all. This eliminates the worker calibration feature, but, perhaps more critically, the feature weights must be set in a different fashion, since we cannot optimize BLEU on reference data anymore. Instead, we use the rank labels (from 3.3) as a proxy for BLEU, and set the weights so that better ranked translations receive higher scores.

Note that the rank features will also be excluded in this setup, since they are perfect predictors of rank labels. On the one hand, this means no rank labels need to be collected, other than for a small set used for weight tuning, further reducing the cost of data collection. However, this leads to a significant drop in performance, yielding a BLEU score of 34.86.

Another alternative for cost reduction would be to reduce the number of collected edited translations. To that end, we first investigate completely eliminating the editing phase, and considering only unedited translations. In other words, the selection will be over a group of four English sentences rather than 14 sentences. Completely eliminating the edited translations has an adverse effect, as expected (Figure 4). Another option, rather than eliminating the editing phase altogether, would be to consider the edited translations of only the translation receiving the best rank labels. This would reflect a data collection process whereby the editing task is delayed until after the rank labels are collected, with the rank labels used to determine which translations are most promising to post-edit (in addition to using the rank labels for the ranking features). Using this approach enables us to greatly reduce the number of edited translations collected, while maintaining good performance, obtaining a BLEU score of 38.67.

It is therefore our recommendation that crowdsourced translation efforts adhere to the following pipeline: collect multiple translations for each source sentence, collect rank labels for the translations, and finally collect edited versions of the top ranked translations.

7 Related Work

Dawid and Skene (1979) investigated filtering annotations using the EM algorithm, estimating annotator-specific error rates in the context of patient medical records. Snow et al. (2008) were among the first to use MTurk to obtain data for several NLP tasks, such as textual entailment and word sense disambiguation. Their approach, based on majority voting, had a component for annotator bias correction. They showed that for such tasks, a few non-expert labels usually suffice.

Whitehill et al. (2009) proposed a probabilistic model to filter labels from non-experts, in the context of an image labeling task. Their system generatively models image difficulty, as well as noisy, even

adversarial, annotators. They apply their method to simulated labels rather than real-life labels.

Callison-Burch (2009) proposed several ways to evaluate MT output on MTurk. One such method was to collect reference translations to score MT output. It was only a pilot study (50 sentences in each of several languages), but it showed the possibility of obtaining high-quality translations from non-professionals. As a followup, Bloodgood and Callison-Burch (2010) solicited a single translation of the NIST Urdu-to-English dataset we used. Their evaluation was similar to our correlation experiments, examining how well the collected translations agreed with the professional translations when evaluating three MT systems.

That paper appeared in a NAACL 2010 workshop organized by Callison-Burch and Dredze (2010), focusing on MTurk as a source of data for speech and language tasks. Two relevant papers from that workshop were by Ambati and Vogel (2010), focusing on the design of the translation HIT, and by Irvine and Klementiev (2010), who created translation lexicons between English and 42 rare languages.

Resnik et al. (2010) explore a very interesting way of creating translations on MTurk, relying only on *monolingual* speakers. Speakers of the target language iteratively identified problems in machine translation output, and speakers of the source language paraphrased the corresponding source portion. The paraphrased source would then be retranslated to produce a different translation, hopefully more coherent than the original.

8 Conclusion and Future Work

We have demonstrated that it is possible to obtain high-quality translations from non-professional translators, and that the cost is an order of magnitude cheaper than professional translation. We believe that crowdsourcing can play a pivotal role in future efforts to create parallel translation datasets. Beyond the cost and scalability, crowdsourcing provides access to languages that currently fall outside the scope of statistical machine translation research. We have begun an ongoing effort to collect translations for several low resource languages, including Tamil, Yoruba, and dialectal Arabic. We plan to:

- Investigate improvements from system combination techniques to the redundant translations.
- Modify our editing step to collect an annotated corpus of English as a second language errors.
- Calibrate against good Turkers, instead of professionals, once they have been identified.
- Predict whether it is necessary to solicit another translation instead of collecting a fixed number.
- Analyze how much quality matters if our goal is to train a statistical translation system.

Acknowledgments

This research was supported by the Human Language Technology Center of Excellence, by gifts from Google and Microsoft, and by the DARPA GALE program under Contract No. HR0011-06-2-0001. The views and findings are the authors' alone.

We would like to thank Ben Bederson, Philip Resnik, and Alain Désilets for organizing workshops focused on crowdsourcing translation (Bederson and Resnik, 2010; Désilets, 2010). We are grateful for the feedback of workshop participants, which helped shape this research.

References

Yaser Al-Onaizan, Ulrich Germann, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Daniel Marcu, and Kenji Yamada. 2002. Translation with scarce bilingual resources. *Machine Translation*, 17(1), March.

Vamshi Ambati and Stephan Vogel. 2010. Can crowds build parallel corpora for machine translation systems? In *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data With Amazon's Mechanical Turk*, pages 62–65.

Ben Bederson and Philip Resnik. 2010. Workshop on crowdsourcing and translation. `http://www.cs.umd.edu/hcil/monotrans/workshop/`.

Michael Bloodgood and Chris Callison-Burch. 2010. Using Mechanical Turk to build machine translation evaluation sets. In *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data With Amazon's Mechanical Turk*, pages 208–211.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with Amazon's Mechanical Turk. In *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data With Amazon's Mechanical Turk*, pages 1–12.

Chris Callison-Burch. 2009. Fast, cheap, and creative: Evaluating translation quality using Amazon's Me-

chanical Turk. In *Proceedings of EMNLP*, pages 286–295.

A. P. Dawid and A. M. Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, 28(1):20–28.

Alain Désilets. 2010. AMTA 2010 workshop on collaborative translation: technology, crowdsourcing, and the translator perspective. `http://bit.ly/gPnqR2`.

Pascale Fung and Lo Yuen Yee. 1998. An ir approach for translating new words from nonparallel, comparable texts. In *Proceedings of ACL/CoLing*.

Ulrich Germann. 2001. Building a statistical machine translation system from scratch: How much bang for the buck can we expect? In *ACL 2001 Workshop on Data-Driven Machine Translation*, Toulouse, France.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL/HLT*.

Panos Ipeirotis. 2010. New demographics of Mechanical Turk. `http://behind-the-enemy-lines.blogspot.com/2010/03/new-demographics-of-mechanical-turk.html`.

Ann Irvine and Alexandre Klementiev. 2010. Using Mechanical Turk to annotate lexicons for less commonly used languages. In *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data With Amazon's Mechanical Turk*, pages 108–113.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Ann Irvine, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Ziyuan Wang, Jonathan Weese, and Omar Zaidan. 2010. Joshua 2.0: A toolkit for parsing-based machine translation with syntax, semirings, discriminative training and other goodies. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 133–137.

Dragos Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting comparable corpora. *Computational Linguistics*, 31(4):477–504, December.

Sonja Niessen and Hermann Ney. 2004. Statistical machine translation with scarce resources using morpho-syntatic analysis. *Computational Linguistics*, 30(2):181–204.

Doug Oard, David Doermann, Bonnie Dorr, Daqing He, Phillip Resnik, William Byrne, Sanjeeve Khudanpur, David Yarowsky, Anton Leuski, Philipp Koehn, and Kevin Knight. 2003. Desperately seeking Cebuano. In *Proceedings of HLT/NAACL*.

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

Kishore Papineni, Salim Poukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Katharina Probst, Lori Levin, Erik Peterson, Alon Lavie, and Jamie Carbonell. 2002. MT for minority languages using elicitation-based learning of syntactic transfer rules. *Machine Translation*, 17(4).

Reinhard Rapp. 1995. Identifying word translations in non-parallel texts. In *Proceedings of ACL*.

Philip Resnik and Noah Smith. 2003. The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380, September.

Philip Resnik, Olivia Buzek, Chang Hu, Yakov Kronrod, Alex Quinn, and Benjamin Bederson. 2010. Improving translation via targeted paraphrasing. In *Proceedings of EMNLP*, pages 127–137.

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *Conference on Natural Language Learning-2002*, pages 146–152.

Jason R. Smith, Chris Quirk, and Kristina Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 403–411, Los Angeles, California, June. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas (AMTA)*.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast – but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP*, pages 254–263.

Jakob Uszkoreit, Jay M. Ponte, Ashok C. Popat, and Moshe Dubiner. 2010. Large scale parallel document mining for machine translation. In *Proc. of the International Conference on Computational Linguistics (COLING)*.

Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier Movellan. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Proceedings of NIPS*, pages 2035–2043.

Omar F. Zaidan. 2009. Z-MERT: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.

A Statistical Tree Annotator and Its Applications

Xiaoqiang Luo and Bing Zhao
IBM T.J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598
{xiaoluo,zhaob}@us.ibm.com

Abstract

In many natural language applications, there is a need to enrich syntactical parse trees. We present a statistical tree annotator augmenting nodes with additional information. The annotator is generic and can be applied to a variety of applications. We report 3 such applications in this paper: predicting function tags; predicting null elements; and predicting whether a tree constituent is projectable in machine translation. Our function tag prediction system outperforms significantly published results.

1 Introduction

Syntactic parsing has made tremendous progress in the past 2 decades (Magerman, 1994; Ratnaparkhi, 1997; Collins, 1997; Charniak, 2000; Klein and Manning, 2003; Carreras et al., 2008), and accurate syntactic parsing is often assumed when developing other natural language applications. On the other hand, there are plenty of language applications where basic syntactic information is insufficient. For instance, in question answering, it is highly desirable to have the semantic information of a syntactic constituent, e.g., a noun-phrase (NP) is a person or an organization; an adverbial phrase is locative or temporal. As syntactic information has been widely used in machine translation systems (Yamada and Knight, 2001; Xiong et al., 2010; Shen et al., 2008; Chiang, 2010; Shen et al., 2010), an interesting question is to predict whether or not a syntactic constituent is *projectable*[1] across a language pair.

[1]A constituent in the source language is projectable if it can be aligned to a contiguous span in the target language.

Such problems can be abstracted as adding additional annotations to an existing tree structure. For example, the English Penn treebank (Marcus et al., 1993) contains function tags and many carry semantic information. To add semantic information to the basic syntactic trees, a logical step is to predict these function tags after syntactic parsing. For the problem of predicting projectable syntactic constituent, one can use a sentence alignment tool and syntactic trees on source sentences to create training data by annotating a tree node as projectable or not. A generic tree annotator can also open the door of solving other natural language problems so long as the problem can be cast as annotating tree nodes. As one such example, we will present how to predict empty elements for the Chinese language.

Some of the above-mentioned problems have been studied before: predicting function tags were studied in (Blaheta and Charniak, 2000; Blaheta, 2003; Lintean and Rus, 2007a), and results of predicting and recovering empty elements can be found in (Dienes et al., 2003; Schmid, 2006; Campbell, 2004). In this work, we will show that these seemingly unrelated problems can be treated uniformly as adding annotations to an existing tree structure, which is the first goal of this work. Second, the proposed generic tree annotator can also be used to solve new problems: we will show how it can be used to predict projectable syntactic constituents. Third, the uniform treatment not only simplifies the model building process, but also affords us to concentrate on discovering most useful features for a particular application which often leads to improved performances, e.g, we find some features are very effective in predicting function tags and our system

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1230–1238,
Portland, Oregon, June 19-24, 2011.

has significant lower error rate than (Blaheta and Charniak, 2000; Lintean and Rus, 2007a).

The rest of the paper is organized as follows. Section 2 describes our tree annotator, which is a conditional log-linear model. Section 3 describes the features used in our system. Next, three applications of the proposed tree annotator are presented in Section 4: predicting English function tags, predicting Chinese empty elements and predicting Arabic projectable constituents. Section 5 compares our work with some related prior arts.

2 A MaxEnt Tree Annotator Model

The input to the tree annotator is a tree T. While T can be of any type, we concentrate on the syntactic parse tree in this paper. The non-terminal nodes, $N = \{n : n \in T\}$ of T are associated with an order by which they are visited so that they can be indexed as n_1, n_2, $\cdots$, $n_{|T|}$, where $|T|$ is the number of non-terminal nodes in T. As an example, Figure 1 shows a syntactic parse tree with the prefix order (i.e., the number at the up-right corner of each non-terminal node), where child nodes are visited recursively from left to right before the parent node is visited. Thus, the `NP-SBJ` node is visited first, followed by the `NP` spanning `duo action`, followed by the `PP-CLR` node etc.

With a prescribed tree visit order, our tree annotator model predicts a symbol l_i, where l_i takes value from a predefined finite set $\mathcal{L}$, for each non-terminal node n_i in a sequential fashion:

$$P(l_1, \cdots, l_{|T|}|T) = \prod_{i=1}^{|T|} P(l_i|l_1, \cdots, l_{i-1}, T) \quad (1)$$

The visit order is important since it determines what are in the conditioning of Eq. (1).

$P(l_i|l_1, \cdots, l_{i-1}, T)$ in this work is a conditional log linear (or MaxEnt) model (Berger et al., 1996):

$$P(l_i|l_1, \cdots, l_{i-1}, T) = \frac{exp\big(\sum_k \lambda_k g_k(l_1^{i-1}, T, l_i)\big)}{Z(l_1^{i-1}, T)} \quad (2)$$

where

$$Z(l_1^{i-1}, T) = \sum_{x \in \mathcal{L}} exp\big(\sum_k \lambda_k g_k(l_1^{i-1}, T, x)\big)$$

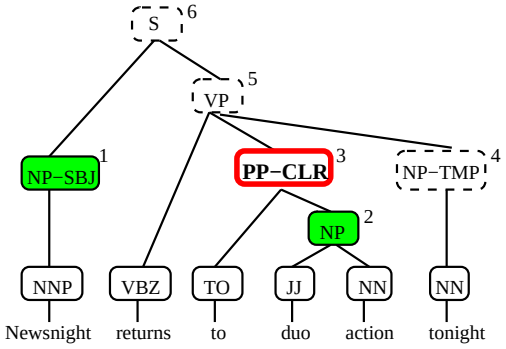


Figure 1: A sample tree: the number on the upright corner of each non-terminal node is the visit order.

is the normalizing factor to ensure that $P(l_i|l_1, \cdots, l_{i-1}, T)$ in Equation (2) is a probability and $\{g_k(l_1^{i-1}, T, l_i)\}$ are feature functions.

There are efficient training algorithms to find optimal weights relative to a labeled training data set once the feature functions $\{g_k(l_1^{i-1}, T, l_i)\}$ are selected (Berger et al., 1996; Goodman, 2002; Malouf, 2002). In our work, we use the SCGIS training algorithm (Goodman, 2002), and the features used in our systems are detailed in the next section.

Once a model is trained, at testing time it is applied to input tree nodes by the same order. Figure 1 highlights the prediction of the function tag for node 3(i.e., `PP-CLR`-node in the thickened box) after 2 shaded nodes (`NP-SBJ` node and `NP` node) are predicted. Note that by this time the predicted values are available to the system, while unvisited nodes (nodes in dashed boxes in Figure 1) can not provide such information.

3 Features

The features used in our systems are tabulated in Table 1. Numbers in the first column are the feature indices. The second column contains a brief description of each feature, and the third column contains the feature value when the feature at the same row is applied to the `PP`-node of Figure 1 for the task of predicting function tags.

Feature 1 through 8 are non-lexical features in that all of them are computed based on the labels or POS tags of neighboring nodes (e.g., Feature 4 computes the label or POS tag of the right most child), or the structure information (e.g., Feature 5 computes the number of child nodes).

Feature 9 and 10 are computed from past predicted values. When predicting the function tag for the PP-node in Figure 1, there is no predicted value for its left-sibling and any of its child node. That's why both feature values are NONE, a special symbol signifying that a node does not carry any function tag. If we were to predict the function tag for the VP-node, the value of Feature 9 would be SBJ, while Feature 10 will be instantiated twice with one value being CLR, another being TMP.

No.	Description	Value
1	current node label	PP
2	parent node label	VP
3	left-most child label/tag	TO
4	right-most child label/tag	NP
5	number of child nodes	2
6	CFG rule	PP->TO NP
7	label/tag of left sibling	VBZ
8	label/tag of right sibling	NP
9	predicted value of left-sibling	NONE
10	predicted value of child nodes	NONE
11	left-most internal word	to
12	right-most internal word	action
13	left neighboring external word	returns
14	right neighboring external word	tonight
15	head word of current node	to
16	head word of parent node	returns
17	is current node the head child	false
18	label/tag of head child	TO
19	predicted value of the head child	NONE

Table 1: Feature functions: the 2nd column contains the descriptions of each feature, and the 3rd column the feature value when it is applied to the PP-node in Figure 1.

Feature 11 to 19 are lexical features or computed from head nodes. Feature 11 and 12 compute the node-internal boundary words, while Feature 13 and 14 compute the immediate node-external boundary words. Feature 15 to 19 rely on the head information. For instance, Feature 15 computes the head word of the current node, which is to for the PP-node in Figure 1. Feature 16 computes the same for the parent node. Feature 17 tests if the current node is the head of its parent. Feature 18 and 19 compute the label or POS tag and the predicted value of the head child, respectively.

Besides the basic feature presented in Table 1, we also use conjunction features. For instance, applying the conjunction of Feature 1 and 18 to the PP-node in Figure 1 would yield a feature instance that captures the fact that the current node is a PP node and its head child's POS tag is TO.

4 Applications and Results

A wide variety of language problems can be treated as or cast into a tree annotating problem. In this section, we present three applications of the statistical tree annotator. The first application is to predict function tags of an input syntactic parse tree; the second one is to predict Chinese empty elements; and the third one is to predict whether a syntactic constituent of a source sentence is *projectable*, meaning if the constituent will have a contiguous translation on the target language.

4.1 Predicting Function Tags

In the English Penn Treebank (Marcus et al., 1993) and more recent OntoNotes data (Hovy et al., 2006), some tree nodes are assigned a function tag, which is of one of the four types: grammatical, form/function, topicalization and miscellaneous. Table 2 contains a list of function tags used in the English Penn Treebank (Bies et al., 1995). The "Grammatical" row contains function tags marking the grammatical role of a constituent, e.g., DTV for dative objects, LGS for logical subjects etc. Many tags in the "Form/function" row carry semantic information, e.g., LOC is for locative expressions, and TMP for temporal expressions.

Type	Function Tags
Grammatical (52.2%)	DTV LGS PRD PUT SBJ VOC
Form/function (36.2%)	ADV BNF DIR EXT LOC MNR NOM PRP TMP
Topicalization (2.2%)	TPC
Miscellaneous (9.4%)	CLF CLR HLN TTL

Table 2: Four types of function tags and their relative frequency

4.1.1 Comparison with Prior Arts

In order to have a direct comparison with (Blaheta and Charniak, 2000; Lintean and Rus, 2007a), we use the same English Penn Treebank (Marcus et al., 1993) and partition the data set identically: Section

2-21 of Wall Street Journal (WSJ) data for training and Section 23 as the test set. We use all features in Table 1 and build four models, each of which predicting one type of function tags. The results are tabulated in Table 3.

As can be seen, our system performs much better than both (Blaheta and Charniak, 2000) and (Lintean and Rus, 2007a). For two major categories, namely grammatical and form/function which account for 96.84% non-null function tags in the test set, our system achieves a relative error reduction of 77.1% (from (Blaheta and Charniak, 2000)'s 1.09% to 0.25%) and 46.9%(from (Blaheta and Charniak, 2000)'s 2.90% to 1.54%) , respectively. The performance improvements result from a clean learning framework and some new features we introduced: e.g., the node-external features, i.e., Feature 13 and 14 in Table 1, can capture long-range statistical dependencies in the conditional model (2) and are proved very useful (cf. Section 4.1.2). As far as we can tell, they are not used in previous work.

Type	Blaheta00	Lintean07	Ours
Grammar	98.91%	98.45%	99.75%
Form/Func	97.10%	95.15%	98.46%
topic	99.92%	99.87%	99.98%
Misc	98.65%	98.54%	99.41%

Table 3: Function tag prediction accuracies on gold parse trees: breakdown by types of function tags. The 2nd column is due to (Blaheta and Charniak, 2000) and 3rd column due to (Lintean and Rus, 2007a). Our results on the 4th column compare favorably with theirs.

4.1.2 Relative Contributions of Features

Since the English WSJ data set contains newswire text, the most recent OntoNotes (Hovy et al., 2006) contains text from a more diversified genres such as broadcast news and broadcast conversation, we decide to test our system on this data set as well. WSJ Section 24 is used for development and Section 23 for test, and the rest is used as the training data. Note that some WSJ files were not included in the OntoNotes release and Section 23 in OntoNotes contains only 1640 sentences. The OntoNotes data statistics is tabulated in Table 4. Less than 2% of nodes with non-empty function tags were assigned multiple function tags. To simplify the system building, we take the first tag in training and testing and report the aggregated accuracy only in this section.

	#-sents	#-nodes	#-funcNodes
training	71,186	1,242,747	280,755
test	1,640	31,117	6,778

Table 4: Statistics of OntoNotes: #-sents – number of sentences; #-nodes – number of non-terminal nodes; #-funcNodes – number of nodes containing non-empty function tags.

We use this data set to test relative contributions of different feature groups by incrementally adding features into the system, and the results are reported in Table 5. The dummy baseline is predicting the most likely prior – the empty function tag, which indicates that there are 78.21% of nodes without a function tag. The next line reflects the performance of a system with non-lexical features only (Feature 1 to 8 in Table 1), and the result is fairly poor with an accuracy 91.51%. The past predictions (Feature 8 and 9) helps a bit by improving the accuracy to 92.04%. Node internal lexical features (Feature 11 and 12) are extremely useful: it added more than 3 points to the accuracy. So does the node external lexical features (Feature 13 and 14) which added an additional 1.52 points. Features computed from head words (Feature 15 to 19) carry information complementary to the lexical features and it helps quite a bit by improving the accuracy by 0.64%. When all features are used, the system reached an accuracy of 97.34%.

From these results, we can conclude that, unlike syntactic parsing (Bikel, 2004), lexical information is extremely important for predicting and recovering function tags. This is not surprising since many function tags carry semantic information, and more often than not, the ambiguity can only be resolved by lexical information. E.g., whether a PP is locative or temporal PP is heavily influenced by the lexical choice of the NP argument.

4.2 Predicting Chinese Empty Elements

As is well known, Chinese is a pro-drop language. This and its lack of subordinate conjunction complementizers lead to the ubiquitous use of empty elements in the Chinese treebank (Xue et al., 2005). Predicting or recovering these empty elements is therefore important for the Chinese language pro-

Feature Set	Accuracy
prior (guess NONE)	78.21%
Non-lexical labels only	91.52%
+past prediction	92.04%
+node-internal lexical	95.17%
+node-external lexical	96.70%
+head word	**97.34%**

Table 5: Effects of feature sets: the second row contains the baseline result when always predicting NONE; Row 3 through 8 contain results by incrementally adding feature sets.

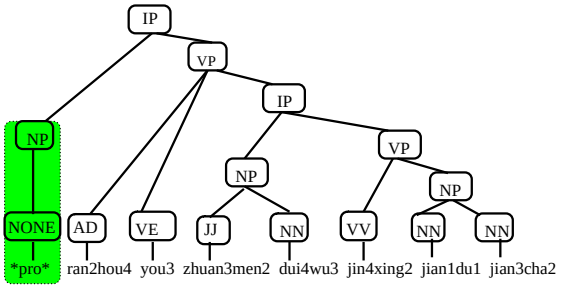
cessing. Recently, Chung and Gildea (2010) has found it useful to recover empty elements in machine translation.

Since empty elements do not have any surface string representation, we tackle the problem by attaching a pseudo function tag to an empty element's lowest non-empty parent and then removing the subtree spanning it. Figure 2 contains an example tree before and after removing the empty element *pro* and annotating the non-empty parent with a pseudo function tag NoneL. The transformation procedure is summarized in Algorithm 1.

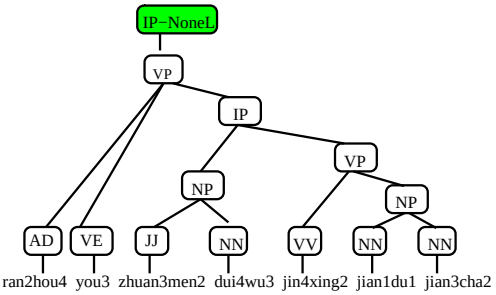
In particular, line 2 of Algorithm 1 find the lowest parent of an empty element that spans at least one non-trace word. In the example in Figure 2, it would find the top IP-node. Since *pro* is the left-most child, line 4 of Algorithm 1 adds the pseudo function tag NoneL to the top IP-node. Line 9 then removes its NP child node and all lower children (i.e., shaded subtree in Figure 2(1)), resulting in the tree in Figure 2(2).

Line 4 to 8 of Algorithm 1 indicate that there are 3 types of pseudo function tags: NoneL, NoneM, and NoneR, encoding a trace found in the left, middle or right position of its lowest non-empty parent. It's trivial to recover a trace's position in a sentence from NoneL, and NoneR, but it may be ambiguous for NoneM. The problem could be solved either using heuristics to determine the position of a middle empty element, or encoding the positional information in the pseudo function tag. Since here we just want to show that predicting empty elements can be cast as a tree annotation problem, we leave this option to future research.

With this transform, the problem of predicting a trace is cast into predicting the corresponding



(1) Original tree with a trace (the left−most child of the top IP−node)



(2) After removing trace and its parent node (shaded subtree in (1))

Figure 2: Transform of traces in a Chinese parse tree by adding pseudo function tags.

Algorithm 1 Procedure to remove empty elements and add pseudo function tags.

Input: An input tree
Output: a tree after removing traces (and their empty parents) and adding pseudo function tags to its lowest non-empty parent node
1:Foreach trace t
2: Find its lowest ancestor node p spanning at least one non-trace word
3: if t is p's left-most child
4: add pseudo tag NoneL to p
5: else if t is p's right-most child
6: add pseudo tag NoneR to p
7: else
8: add pseudo tag NoneM to p
9: Remove p's child spanning the trace t and all its children

pseudo function tag and the statistical tree annotator can thus be used to solve this problem.

4.2.1 Results

We use Chinese Treebank v6.0 (Xue et al., 2005) and the broadcast conversation data from CTB v7.0 [2]. The data set is partitioned into training, development and blind test as shown in Table 6. The partition is created so that different genres are well represented in different subsets. The training, development and test set have 32925, 3297 and 3033 sentences, respectively.

Subset	File IDs
Training	0001-0325, 0400-0454, 0600-0840 0500-0542, 2000-3000, 0590-0596 1001-1120, cctv,cnn,msnbc, phoenix 00-06
Dev	0841-0885, 0543-0548, 3001-3075 1121-1135, phoenix 07-09
Test	0900-0931,0549-0554, 3076-3145 1136-1151, phoenix 10-11

Table 6: Data partition for CTB6 and CTB 7's broadcast conversation portion

We then apply Algorithm 1 to transform trees and predict pseudo function tags. Out of 1,100,506 non-terminal nodes in the training data, 80,212 of them contain pseudo function tags. There are 94 nodes containing 2 pseudo function tags. The vast majority of pseudo tags – more then 99.7% – are attached to either `IP`, `CP`, or `VP`: 50971, 20113, 8900, respectively.

We used all features in Table 1 and achieved an accuracy of 99.70% on the development data, and 99.71% on the test data on gold trees.

To understand why the accuracies are so high, we look into the 5 most frequent labels carrying pseudo tags in the development set, and tabulate their performance in Table 7. The 2nd column contains the number of nodes in the reference; the 3rd column the number of nodes of system output; the 4th column the number of nodes with correct prediction; and the 5th column F-measure for each label.

From Table 7, it is clear that `CP-NoneL` and `IP-NoneL` are easy to predict. This is not surprising, given that the Chinese language lacks of complementizers for subordinate clauses. In other words, left-most empty elements under `CP` are almost unambiguous: if a `CP` node has an immediate `IP` child, it almost always has a left-most empty element; similarly, if an `IP` node has a `VP` node as the left-most child (i.e., without a subject), it almost always should have a left empty element (e.g., marking the dropped pro). Another way to interpret these results is as follows: when developing the Chinese treebank, there is really no point to annotate left-most traces for `CP` and `IP` when tree structures are available.

Label	numRef	numSys	numCorr	F1
CP-NoneL	1723	1724	1715	0.995
IP-NoneL	3874	3875	3844	0.992
VP-NoneR	660	633	597	0.923
IP-NoneM	440	432	408	0.936
VP-NoneL	135	107	105	0.868

Table 7: 5 most frequent labels carrying pseudo tags and their performances

On the other hand, predicting the left-most empty elements for `VP` is a lot harder: the F-measure is only 86.8% for `VP-NoneL`. Predicting the right-most empty elements under `VP` and middle empty elements under `IP` is somewhat easier: `VP-NoneR` and `IP-NoneM`'s F-measures are 92.3% and 93.6%, respectively.

4.3 Predicting Projectable Constituents

The third application is predicting projectable constituents for machine translation. State-of-the-art machine translation systems (Yamada and Knight, 2001; Xiong et al., 2010; Shen et al., 2008; Chiang, 2010; Shen et al., 2010) rely heavily on syntactic analysis. Projectable structures are important in that it is assumed in CFG-style translation rules that a source span can be translated contiguously. Clearly, not all source constituents can be translated this way, but if we can predict whether a non-terminal source node is projectable, we can avoid translation errors by bypassing or discouraging the derivation paths relying on non-projectable constituents, or using phrase-based approaches for non-projectable constituents.

We start from LDC's bilingual Arabic-English treebank with source human parse trees and alignments, and mark source constituents as either pro-

[2] Many files are missing in LDC's early 2010 release of CTB 7.0, but broadcast conversation portion is new and is used in our system.

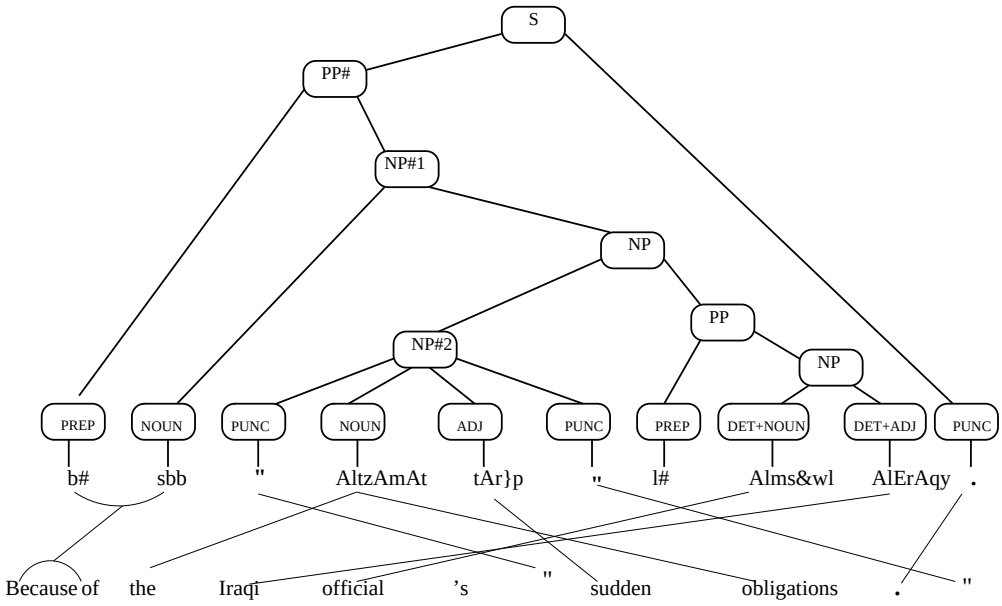


Figure 3: An example to show how a source tree is annotated with its alignment with the target sentence.

jectable or non-projectable. The binary annotations can again be treated as pseudo function tags and the proposed tree annotator can be readily applied to this problem.

As an example, the top half of Figure 3 contains an Arabic sentence with its parse tree; the bottom is its English translation with the human word-alignment. There are three non-projectable constituents marked with "#": the top `PP#` spanning the whole sentence except the final stop, and `NP#1` and `NP#2`. The `PP#` node is not projectable due to an inserted stop from outside; `NP#1` is not projectable because it is involved in a 2-to-2 alignment with the token `b#` outside `NP#1`; `NP#2` is aligned to a span `the Iraqi official 's sudden obligations .`, in which `Iraqi official` breaks the contiguity of the translation. It is clear that a CFG-like grammar will not be able to generate the translation for `NP#2`.

The LDC's Arabic-English bilingual treebank does not mark if a source node is projectable or not, but the information can be computed from word alignment. In our experiments, we processed 16,125 sentence pairs with human source trees for training, and 1,151 sentence pairs for testing. The statistics of the training and test data can be found in Table 8, where the number of sentences, the number of non-terminal nodes and the number of non-projectable nodes are listed in Column 2 through 4, respectively.

Data Set	#Sents	#nodes	#NonProj
Training	16,125	558,365	121,201
Test	1,151	40,674	8,671

Table 8: Statistics of the data for predicting projectable constituents

We get a 94.6% accuracy for predicting projectable constituents on the gold trees, and an 84.7% F-measure on the machine-generated parse trees. This component has been integrated into our machine translation system (Zhao et al., 2011).

5 Related Work

Blaheta and Charniak (2000) used a feature tree model to predict function tags. The work was later extended to use the voted perceptron (Blaheta, 2003). There are considerable overlap in terms of features used in (Blaheta and Charniak, 2000; Blaheta, 2003) and our system: for example, the label of current node, parent node and sibling nodes. However, there are some features that are unique in our work, e.g., lexical features at a constituent boundaries (node-internal and node-external words). Table 2 of (Blaheta and Charniak, 2000) contains the ac-

curacies for 4 types of function tags, and our results in Table 3 compare favorably with those in (Blaheta and Charniak, 2000). Lintean and Rus (2007a; Lintean and Rus (2007b) also studied the function tagging problem and applied naive Bayes and decision tree to it. Their accuracy results are worse than (Blaheta and Charniak, 2000). Neither (Blaheta and Charniak, 2000) nor (Lintean and Rus, 2007a; Lintean and Rus, 2007b) reported the relative usefulness of different features, while we found that the lexical features are extremely useful.

Campbell (2004) and Schmid (2006) studied the problem of predicting and recovering empty categories, but they used very different approaches: in (Campbell, 2004), a rule-based approach is used while (Schmid, 2006) used a non-lexical PCFG similar to (Klein and Manning, 2003). Chung and Gildea (2010) studied the effects of empty categories on machine translation and they found that even with noisy machine predictions, empty categories still helped machine translation. In this paper, we showed that empty categories can be encoded as pseudo function tags and thus predicting and recovering empty categories can be cast as a tree annotating problem. Our results also shed light on some empty categories can almost be determined unambiguously, given a gold tree structure, which suggests that these empty elements do not need to be annotated.

Gabbard et al. (2006) modified Collins' parser to output function tags. Since their results for predicting function tags are on system parses, they are not comparable with ours. (Gabbard et al., 2006) also contains a second stage employing multiple classifiers to recover empty categories and resolve coindexations between an empty element and its antecedent.

As for predicting projectable constituent, it is related to the work described in (Xiong et al., 2010), where they were predicting translation boundaries. A major difference is that (Xiong et al., 2010) defines projectable spans on a left-branching derivation tree solely for their phrase decoder and models, while translation boundaries in our work are defined from source parse trees. Our work uses more resources, but the prediction accuracy is higher (modulated on a different test data): we get a F-measure 84.7%, in contrast with (Xiong et al., 2010)'s 71%.

6 Conclusions and Future Work

We proposed a generic statistical tree annotator in the paper. We have shown that a variety of natural language problems can be tackled with the proposed tree annotator, from predicting function tags, predicting empty categories, to predicting projectable syntactic constituents for machine translation. Our results of predicting function tags compare favorably with published results on the same data set, possibly due to new features employed in the system. We showed that empty categories can be represented as pseudo function tags, and thus predicting empty categories can be solved with the proposed tree annotator. The same technique can be used to predict projectable syntactic constituents for machine translation.

There are several directions to expand the work described in this paper. First, the results for predicting function tags and Chinese empty elements were obtained on human-annotated trees and it would be interesting to do it on parse trees generated by system. Second, predicting projectable constituents is for improving machine translation and we are integrating the component into a syntax-based machine translation system.

Acknowledgments

This work was partially supported by the Defense Advanced Research Projects Agency under contract No. HR0011-08-C-0110. The views and findings contained in this material are those of the authors and do not necessarily reflect the position or policy of the U.S. government and no official endorsement should be inferred.

We are also grateful to three anonymous reviewers for their suggestions and comments for improving the paper.

References

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, March.

Ann Bies, Mark Ferguson, and karen Katz. 1995. Bracketing guidelines for treebank II-style penn treebank project. Technical report, Linguistic Data Consortium.

Daniel M. Bikel. 2004. A distributional analysis of a lexicalized statistical parsing model. In Dekang Lin

and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 182–189, Barcelona, Spain, July. Association for Computational Linguistics.

Don Blaheta and Eugene Charniak. 2000. Assigning function tags to parsed text. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 234–240.

Don Blaheta. 2003. *Function Tagging*. Ph.D. thesis, Brown University.

Richard Campbell. 2004. Using linguistic principles to recover empty categories. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 645–652, Barcelona, Spain, July.

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

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*, Seattle.

David Chiang. 2010. Learning to translate with source and target syntax. In *Proc. ACL*, pages 1443–1452.

Tagyoung Chung and Daniel Gildea. 2010. Effects of empty categories on machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 636–645, Cambridge, MA, October. Association for Computational Linguistics.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. Annual Meeting of ACL*, pages 16–23.

Peter Dienes, P Eter Dienes, and Amit Dubey. 2003. Antecedent recovery: Experiments with a trace tagger. In *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 33–40.

Ryan Gabbard, Mitchell Marcus, and Seth Kulick. 2006. Fully parsing the Penn Treebank. In *Proceedings of Human Language Technology Conference of the North Amer- ican Chapter of the Association of Computational Linguistics*.

Joshua Goodman. 2002. Sequential conditional generalized iterative scaling. In *Pro. of the 40th ACL*.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, New York City, USA, June. Association for Computational Linguistics.

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

Mihai Lintean and V. Rus. 2007a. Large scale experiments with function tagging. In *Proceedings of the International Conference on Knowledge Engineering*, pages 1–7.

Mihai Lintean and V. Rus. 2007b. Naive Bayes and decision trees for function tagging. In *Proceedings of the International Conference of the FLAIRS-2007*.

David M. Magerman. 1994. *Natural Language Parsing As Statistical Pattern Recognition*. Ph.D. thesis, Stanford University.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *the Sixth Conference on Natural Language Learning (CoNLL-2002)*, pages 49–55.

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

Adwait Ratnaparkhi. 1997. A Linear Observed Time Statistical Parser Based on Maximum Entropy Models. In *Second Conference on Empirical Methods in Natural Language Processing*, pages 1 – 10.

Helmut Schmid. 2006. Trace prediction and recovery with unlexicalized PCFGs and slash features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 177–184, Sydney, Australia, July. Association for Computational Linguistics.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL*.

Libin Shen, Bing Zhang, Spyros Matsoukas, Jinxi Xu, and Ralph Weischedel. 2010. Statistical machine translation with a factorized grammar. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 616–625, Cambridge, MA, October. Association for Computational Linguistics.

Deyi Xiong, Min Zhang, and Haizhou Li. 2010. Learning translation boundaries for phrase-based decoding. In *NAACL-HLT 2010*.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. Annual Meeting of the Association for Computational Linguistics*.

Bing Zhao, , Young-Suk Lee, Xiaoqiang Luo, and Liu Li. 2011. Learning to transform and select elementary trees for improved syntax-based machine translations. In *Proc. of ACL*.

Consistent Translation using Discriminative Learning: A Translation Memory-inspired Approach*

Yanjun Ma[†] **Yifan He**[‡] **Andy Way**[‡] **Josef van Genabith**[‡]
[†] Baidu Inc., Beijing, China
yma@baidu.com
[‡]Centre for Next Generation Localisation
School of Computing, Dublin City University
{yhe,away,josef}@computing.dcu.ie

Abstract

We present a discriminative learning method to improve the consistency of translations in phrase-based Statistical Machine Translation (SMT) systems. Our method is inspired by Translation Memory (TM) systems which are widely used by human translators in industrial settings. We constrain the translation of an input sentence using the most similar 'translation example' retrieved from the TM. Differently from previous research which used simple fuzzy match thresholds, these constraints are imposed using discriminative learning to optimise the translation performance. We observe that using this method can benefit the SMT system by not only producing consistent translations, but also improved translation outputs. We report a 0.9 point improvement in terms of BLEU score on English–Chinese technical documents.

1 Introduction

Translation consistency is an important factor for large-scale translation, especially for domain-specific translations in an industrial environment. For example, in the translation of technical documents, lexical as well as structural consistency is essential to produce a fluent target-language sentence. Moreover, even in the case of translation errors, consistency in the errors (e.g. repetitive error patterns) are easier to diagnose and subsequently correct by translators.

In phrase-based SMT, translation models and language models are automatically learned and/or generalised from the training data, and a translation is produced by maximising a weighted combination of these models. Given that global contextual information is not normally incorporated, and that training data is usually noisy in nature, there is no guarantee that an SMT system can produce translations in a consistent manner.

On the other hand, TM systems – widely used by translators in industrial environments for enterprise localisation by translators – can shed some light on mitigating this limitation. TM systems can assist translators by retrieving and displaying previously translated similar 'example' sentences (displayed as source-target pairs, widely called 'fuzzy matches' in the localisation industry (Sikes, 2007)). In TM systems, fuzzy matches are retrieved by calculating the similarity or the so-called 'fuzzy match score' (ranging from 0 to 1 with 0 indicating no matches and 1 indicating a full match) between the input sentence and sentences in the source side of the translation memory.

When presented with fuzzy matches, translators can then avail of useful chunks in previous translations while composing the translation of a new sentence. Most translators only consider a few sentences that are most similar to the current input sentence; this process can inherently improve the consistency of translation, given that the new translations produced by translators are likely to be similar to the target side of the fuzzy match they have consulted.

Previous research as discussed in detail in Sec-

*This work was done while the first author was in the Centre for Next Generation Localisation at Dublin City University.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1239–1248,
Portland, Oregon, June 19-24, 2011.

tion 2 has focused on using fuzzy match score as a threshold when using the target side of the fuzzy matches to constrain the translation of the input sentence. In our approach, we use a more fine-grained discriminative learning method to determine whether the target side of the fuzzy matches should be used as a constraint in translating the input sentence. We demonstrate that our method can consistently improve translation quality.

The rest of the paper is organized as follows: we begin by briefly introducing related research in Section 2. We present our discriminative learning method for consistent translation in Section 3 and our feature design in Section 4. We report the experimental results in Section 5 and conclude the paper and point out avenues for future research in Section 6.

2 Related Research

Despite the fact that TM and MT integration has long existed as a major challenge in the localisation industry, it has only recently received attention in main-stream MT research. One can loosely combine TM and MT at sentence (called segments in TMs) level by choosing one of them (or both) to recommend to the translators using automatic classifiers (He et al., 2010), or simply using fuzzy match score or MT confidence measures (Specia et al., 2009).

One can also tightly integrate TM with MT at the sub-sentence level. The basic idea is as follows: given a source sentence to translate, we firstly use a TM system to retrieve the most similar 'example' source sentences together with their translations. If matched chunks between input sentence and fuzzy matches can be detected, we can directly re-use the corresponding parts of the translation in the fuzzy matches, and use an MT system to translate the remaining chunks.

As a matter of fact, implementing this idea is pretty straightforward: a TM system can easily detect the word alignment between the input sentence and the source side of the fuzzy match by retracing the paths used in calculating the fuzzy match score. To obtain the translation for the matched chunks, we just require the word alignment between source and target TM matches, which can be addressed using state-of-the-art word alignment techniques. More importantly, albeit not explicitly spelled out in previous work, this method can potentially increase the consistency of translation, as the translation of new input sentences is closely informed and guided (or constrained) by previously translated sentences.

There are several different ways of using the translation information derived from fuzzy matches, with the following two being the most widely adopted: 1) to add these translations into a phrase table as in (Biçici and Dymetman, 2008; Simard and Isabelle, 2009), or 2) to mark up the input sentence using the relevant chunk translations in the fuzzy match, and to use an MT system to translate the parts that are not marked up, as in (Smith and Clark, 2009; Koehn and Senellart, 2010; Zhechev and van Genabith, 2010). It is worth mentioning that translation consistency was not explicitly regarded as their primary motivation in this previous work. Our research follows the direction of the second strand given that consistency can no longer be guaranteed by constructing another phrase table.

However, to categorically reuse the translations of matched chunks without any differentiation could generate inferior translations given the fact that the context of these matched chunks in the input sentence could be completely different from the source side of the fuzzy match. To address this problem, both (Koehn and Senellart, 2010) and (Zhechev and van Genabith, 2010) used fuzzy match score as a threshold to determine whether to reuse the translations of the matched chunks. For example, (Koehn and Senellart, 2010) showed that reusing these translations as large rules in a hierarchical system (Chiang, 2005) can be beneficial when the fuzzy match score is above 70%, while (Zhechev and van Genabith, 2010) reported that it is only beneficial to a phrase-based system when the fuzzy match score is above 90%.

Despite being an informative measure, using fuzzy match score as a threshold has a number of limitations. Given the fact that fuzzy match score is normally calculated based on Edit Distance (Levenshtein, 1966), a low score does not necessarily imply that the fuzzy match is harmful when used to constrain an input sentence. For example, in longer sentences where fuzzy match scores tend to be low, some chunks and the corresponding translations within the sentences can still be useful. On

the other hand, a high score cannot fully guarantee the usefulness of a particular translation. We address this problem using discriminative learning.

3 Constrained Translation with Discriminative Learning

3.1 Formulation of the Problem

Given a sentence $\mathbf{e}$ to translate, we retrieve the most similar sentence $\mathbf{e}'$ from the translation memory associated with target translation $\mathbf{f}'$. The m common "phrases" $\bar{\mathbf{e}}_{\mathbf{1}}^{\mathbf{m}}$ between $\mathbf{e}$ and $\mathbf{e}'$ can be identified. Given the word alignment information between $\mathbf{e}'$ and $\mathbf{f}'$, one can easily obtain the corresponding translations $\bar{\mathbf{f}}'^{\mathbf{m}}_{\mathbf{1}}$ for each of the phrases in $\bar{\mathbf{e}}_{\mathbf{1}}^{\mathbf{m}}$. This process can derive a number of "phrase pairs" $< \bar{e}_m, \bar{f}'_m >$, which can be used to specify the translations of the matched phrases in the input sentence. The remaining words without specified translations will be translated by an MT system.

For example, given an input sentence $e_1 e_2 \cdots e_i e_{i+1} \cdots e_I$, and a phrase pair $< \bar{e}, \bar{f}' >$, $\bar{e} = e_i e_{i+1}$, $\bar{f}' = f'_j f'_{j+1}$ derived from the fuzzy match, we can mark up the input sentence as:

$e_1 e_2 \cdots$ <tm="$f'_j f'_{j+1}$"> $e_i e_{i+1}$ < /tm> $\cdots e_I$.

Our method to constrain the translations using TM fuzzy matches is similar to (Koehn and Senellart, 2010), except that the word alignment between $\mathbf{e}'$ and $\mathbf{f}'$ is the intersection of bidirectional GIZA++ (Och and Ney, 2003) posterior alignments. We use the intersected word alignment to minimise the noise introduced by word alignment of only one direction in marking up the input sentence.

3.2 Discriminative Learning

Whether the translation information from the fuzzy matches should be used or not (i.e. whether the input sentence should be marked up) is determined using a discriminative learning procedure. The translation information refers to the "phrase pairs" derived using the method described in Section 3.1. We cast this problem as a binary classification problem.

3.2.1 Support Vector Machines

SVMs (Cortes and Vapnik, 1995) are binary classifiers that classify an input instance based on decision rules which minimise the regularised error function in (1):

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{s. t.} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geqslant 1 - \xi_i \\ & \xi_i \geqslant 0 \end{aligned} \tag{1}$$

where $(\mathbf{x}_i, y_i) \in R^n \times \{+1, -1\}$ are l training instances that are mapped by the function ϕ to a higher dimensional space. $\mathbf{w}$ is the weight vector, ξ is the relaxation variable and $C > 0$ is the penalty parameter.

Solving SVMs is viable using a kernel function K in (1) with $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j)$. We perform our experiments with the Radial Basis Function (RBF) kernel, as in (2):

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2), \gamma > 0 \tag{2}$$

When using SVMs with the RBF kernel, we have two free parameters to tune on: the cost parameter C in (1) and the radius parameter γ in (2).

In each of our experimental settings, the parameters C and γ are optimised by a brute-force grid search. The classification result of each set of parameters is evaluated by cross validation on the training set.

The SVM classifier will thus be able to predict the usefulness of the TM fuzzy match, and determine whether the input sentence should be marked up using relevant phrase pairs derived from the fuzzy match before sending it to the SMT system for translation. The classifier uses features such as the fuzzy match score, the phrase and lexical translation probabilities of these relevant phrase pairs, and additional syntactic dependency features. Ideally the classifier will decide to mark up the input sentence if the translations of the marked phrases are accurate when taken contextual information into account. As large-scale manually annotated data is not available for this task, we use automatic TER scores (Snover et al., 2006) as the measure for training data annotation.

We label the training examples as in (3):

$$y = \begin{cases} +1 & \text{if } TER(\text{w. markup}) < TER(\text{w/o markup}) \\ -1 & \text{if } TER(\text{w/o markup}) \geq TER(\text{w. markup}) \end{cases} \tag{3}$$

Each instance is associated with a set of features which are discussed in more detail in Section 4.

3.2.2 Classification Confidence Estimation

We use the techniques proposed by (Platt, 1999) and improved by (Lin et al., 2007) to convert classification margin to posterior probability, so that we can easily threshold our classifier (cf. Section 5.4.2).

Platt's method estimates the posterior probability with a sigmoid function, as in (4):

$$Pr(y=1|\mathbf{x}) \approx P_{A,B}(f) \equiv \frac{1}{1+exp(Af+B)} \quad (4)$$

where $f = f(\mathbf{x})$ is the decision function of the estimated SVM. A and B are parameters that minimise the cross-entropy error function F on the training data, as in (5):

$$\min_{z=(A,B)} F(z) = -\sum_{i=1}^{l}(t_i log(p_i) + (1-t_i) log(1-p_i)),$$
$$\text{where } p_i = P_{A,B}(f_i), \text{and } t_i = \begin{cases} \frac{N_+ +1}{N_+ +2} \text{ if } y_i = +1 \\ \frac{1}{N_- +2} \text{ if } y_i = -1 \end{cases} \quad (5)$$

where $z = (A, B)$ is a parameter setting, and N_+ and N_- are the numbers of observed positive and negative examples, respectively, for the label y_i. These numbers are obtained using an internal cross-validation on the training set.

4 Feature Set

The features used to train the discriminative classifier, all on the sentence level, are described in the following sections.

4.1 The TM Feature

The TM feature is the fuzzy match score, which indicates the overall similarity between the input sentence and the source side of the TM output. If the input sentence is similar to the source side of the matching segment, it is more likely that the matching segment can be used to mark up the input sentence.

The calculation of the fuzzy match score itself is one of the core technologies in TM systems, and varies among different vendors. We compute fuzzy match cost as the minimum Edit Distance (Levenshtein, 1966) between the source and TM entry, normalised by the length of the source as in (6), as most of the current implementations are based on edit distance while allowing some additional flexible matching.

$$h_{fm}(\mathbf{e}) = \min_{\mathbf{s}} \frac{EditDistance(\mathbf{e}, \mathbf{s})}{Len(\mathbf{e})} \quad (6)$$

where $\mathbf{e}$ is the sentence to translate, and $\mathbf{s}$ is the source side of an entry in the TM. For fuzzy match scores F, h_{fm} roughly corresponds to $1 - F$.

4.2 Translation Features

We use four features related to translation probabilities, i.e. the phrase translation and lexical probabilities for the phrase pairs $< \bar{e}_m, \bar{f'}_m >$ derived using the method in Section 3.1. Specifically, we use the phrase translation probabilities $p(\bar{f'}_m|\bar{e}_m)$ and $p(\bar{e}_m|\bar{f'}_m)$, as well as the lexical translation probabilities $p_{lex}(\bar{f'}_m|\bar{e}_m)$ and $p_{lex}(\bar{e}_m|\bar{f'}_m)$ as calculated in (Koehn et al., 2003). In cases where multiple phrase pairs are used to mark up one single input sentence $\mathbf{e}$, we use a unified score for each of the four features, which is an average over the corresponding feature in each phrase pair. The intuition behind these features is as follows: phrase pairs $< \bar{e}_m, \bar{f'}_m >$ derived from the fuzzy match should also be reliable with respect to statistically produced models.

We also have a count feature, i.e. the number of phrases used to mark up the input sentence, and a binary feature, i.e. whether the phrase table contains at least one phrase pair $< \bar{e}_m, \bar{f'}_m >$ that is used to mark up the input sentence.

4.3 Dependency Features

Given the phrase pairs $< \bar{e}_m, \bar{f'}_m >$ derived from the fuzzy match, and used to translate the corresponding chunks of the input sentence (cf. Section 3.1), these translations are more likely to be coherent in the context of the particular input sentence if the matched parts on the input side are syntactically and semantically related.

For matched phrases $\bar{e}_m$ between the input sentence and the source side of the fuzzy match, we define the contextual information of the input side using dependency relations between words e_m in $\bar{e}_m$ and the remaining words e_j in the input sentence $\mathbf{e}$.

We use the Stanford parser to obtain the dependency structure of the input sentence. We add a pseudo-label SYS_PUNCT to punctuation marks, whose governor and dependent are both the punctuation mark. The dependency features designed to capture the context of the matched input phrases $\bar{e}_m$ are as follows:

Coverage features measure the coverage of dependency labels on the input sentence in order to obtain a bigger picture of the matched parts in the input. For each dependency label *L*, we consider its head or modifier as *covered* if the corresponding input word e_m is covered by a matched phrase $\bar{e}_m$. Our coverage features are the frequencies of governor and dependent coverage calculated separately for each dependency label.

Position features identify whether the head and the tail of a sentence are matched, as these are the cases in which the matched translation is not affected by the preceding words (when it is the head) or following words (when it is the tail), and is therefore more reliable. The feature is set to 1 if this happens, and to 0 otherwise. We distinguish among the possible dependency labels, the head or the tail of the sentence, and whether the aligned word is the governor or the dependent. As a result, each permutation of these possibilities constitutes a distinct binary feature.

The consistency feature is a single feature which determines whether matched phrases $\bar{e}_m$ belong to a consistent dependency structure, instead of being distributed discontinuously around in the input sentence. We assume that a consistent structure is less influenced by its surrounding context. We set this feature to 1 if every word in $\bar{e}_m$ is dependent on another word in $\bar{e}_m$, and to 0 otherwise.

5 Experiments

5.1 Experimental Setup

Our data set is an English–Chinese translation memory with technical translation from Symantec, consisting of 87K sentence pairs. The average sentence length of the English training set is 13.3 words and the size of the training set is comparable to the larger TMs used in the industry. Detailed corpus statistics about the training, development and test sets for the SMT system are shown in Table 1.

The composition of test subsets based on fuzzy match scores is shown in Table 2. We can see that sentences in the test sets are longer than those in the training data, implying a relatively difficult translation task. We train the SVM classifier using the libSVM (Chang and Lin, 2001) toolkit. The SVM-

	Train	Develop	Test
SENTENCES	86,602	762	943
ENG. TOKENS	1,148,126	13,955	20,786
ENG. VOC.	13,074	3,212	3,115
CHI. TOKENS	1,171,322	10,791	16,375
CHI. VOC.	12,823	3,212	1,431

Table 1: Corpus Statistics

Scores	Sentences	Words	W/S
(0.9, 1.0)	80	1526	19.0750
(0.8, 0.9]	96	1430	14.8958
(0.7, 0.8]	110	1596	14.5091
(0.6, 0.7]	74	1031	13.9324
(0.5, 0.6]	104	1811	17.4135
(0, 0.5]	479	8972	18.7307

Table 2: Composition of test subsets based on fuzzy match scores

training and validation is on the same training sentences[1] as the SMT system with 5-fold cross validation.

The SVM hyper-parameters are tuned using the training data of the first fold in the 5-fold cross validation via a brute force grid search. More specifically, for parameter C in (1), we search in the range $[2^{-5}, 2^{15}]$, while for parameter γ (2) we search in the range $[2^{-15}, 2^{3}]$. The step size is 2 on the exponent.

We conducted experiments using a standard log-linear PB-SMT model: GIZA++ implementation of IBM word alignment model 4 (Och and Ney, 2003), the refinement and phrase-extraction heuristics described in (Koehn et al., 2003), minimum-error-rate training (Och, 2003), a 5-gram language model with Kneser-Ney smoothing (Kneser and Ney, 1995) trained with SRILM (Stolcke, 2002) on the Chinese side of the training data, and Moses (Koehn et al., 2007) which is capable of handling user-specified translations for some portions of the input during decoding. The maximum phrase length is set to 7.

5.2 Evaluation

The performance of the phrase-based SMT system is measured by BLEU score (Papineni et al., 2002) and TER (Snover et al., 2006). Significance test-

[1]We have around 87K sentence pairs in our training data. However, for 67.5% of the input sentences, our MT system produces the same translation irrespective of whether the input sentence is marked up or not.

ing is carried out using approximate randomisation (Noreen, 1989) with a 95% confidence level.

We also measure the quality of the classification by precision and recall. Let A be the set of predicted markup input sentences, and B be the set of input sentences where the markup version has a lower TER score than the plain version. We standardly define precision P and recall R as in (7):

$$P = \frac{|A \bigcap B|}{|A|}, R = \frac{|A \bigcap B|}{|B|} \quad (7)$$

5.3 Cross-fold translation

In order to obtain training samples for the classifier, we need to label each sentence in the SMT training data as to whether marking up the sentence can produce better translations. To achieve this, we translate both the marked-up versions and plain versions of the sentence and compare the two translations using the sentence-level evaluation metric TER.

We do not make use of additional training data to translate the sentences for SMT training, but instead use cross-fold translation. We create a new training corpus T by keeping 95% of the sentences in the original training corpus, and creating a new test corpus H by using the remaining 5% of the sentences. Using this scheme we make 20 different pairs of corpora (T_i, H_i) in such a way that each sentence from the original training corpus is in exactly one H_i for some $1 \leq i \leq 20$. We train 20 different systems using each T_i, and use each system to translate the corresponding H_i as well as the marked-up version of H_i using the procedure described in Section 3.1. The development set is kept the same for all systems.

5.4 Experimental Results

5.4.1 Translation Results

Table 3 contains the translation results of the SMT system when we use discriminative learning to mark up the input sentence (MARKUP-DL). The first row (BASELINE) is the result of translating plain test sets without any markup, while the second row is the result when all the test sentences are marked up. We also report the oracle scores, i.e. the upperbound of using our discriminative learning approach. As we can see from this table, we obtain significantly inferior results compared to the the Baseline system if we categorically mark up all the input sentences using phrase pairs derived from fuzzy matches. This is reflected by an absolute 1.4 point drop in BLEU score and a 1.8 point increase in TER. On the other hand, both the oracle BLEU and TER scores represent as much as a 2.5 point improvement over the baseline. Our discriminative learning method (MARKUP-DL), which automatically classifies whether an input sentence should be marked up, leads to an increase of 0.7 absolute BLEU points over the BASELINE, which is statistically significant. We also observe a slight decrease in TER compared to the BASELINE. Despite there being much room for further improvement when compared to the Oracle score, the discriminative learning method appears to be effective not only in maintaining translation consistency, but also a statistically significant improvement in translation quality.

	TER	BLEU
BASELINE	39.82	45.80
MARKUP	41.62	44.41
MARKUP-DL	39.61	46.46
ORACLE	37.27	48.32

Table 3: Performance of Discriminative Learning (%)

5.4.2 Classification Confidence Thresholding

To further analyse our discriminative learning approach, we report the classification results on the test set using the SVM classifier. We also investigate the use of classification confidence, as described in Section 3.2.2, as a threshold to boost classification precision if required. Table 4 shows the classification and translation results when we use different confidence thresholds. The default classification confidence is 0.50, and the corresponding translation results were described in Section 5.4.1. We investigate the impact of increasing classification confidence on the performance of the classifier and the translation results. As can be seen from Table 4, increasing the classification confidence up to 0.70 leads to a steady increase in classification precision with a corresponding sacrifice in recall. The fluctuation in classification performance has an impact on the translation results as measured by BLEU and TER. We can see that the best BLEU as well as TER scores are achieved when we set the classification confidence to 0.60, representing a modest improve-

	Classification Confidence						
	0.50	0.55	0.60	0.65	0.70	0.75	0.80
BLEU	46.46	46.65	**46.69**	46.59	46.34	46.06	46.00
TER	39.61	39.46	**39.32**	39.36	39.52	39.71	39.71
P	60.00	68.67	70.31	**74.47**	72.97	64.28	88.89
R	32.14	29.08	22.96	**17.86**	13.78	9.18	4.08

Table 4: The impact of classification confidence thresholding

ment over the default setting (0.50). Despite the higher precision when the confidence is set to 0.7, the dramatic decrease in recall cannot be compensated for by the increase in precision.

We can also observe from Table 4 that the recall is quite low across the board, and the classification results become unstable when we further increase the level of confidence to above 0.70. This indicates the degree of difficulty of this classification task, and suggests some directions for future research as discussed at the end of this paper.

5.4.3 Comparison with Previous Work

As discussed in Section 2, both (Koehn and Senellart, 2010) and (Zhechev and van Genabith, 2010) used fuzzy match score to determine whether the input sentences should be marked up. The input sentences are only marked up when the fuzzy match score is above a certain threshold. We present the results using this method in Table 5. From this table, we can see an inferior performance compared to the BASELINE results (cf. Table 3) when the fuzzy match score is below 0.70. A modest gain can only be achieved when the fuzzy match score is above 0.8. This is slightly different from the conclusions drawn in (Koehn and Senellart, 2010), where gains are observed when the fuzzy match score is above 0.7, and in (Zhechev and van Genabith, 2010) where gains are only observed when the score is above 0.9. Comparing Table 5 with Table 4, we can see that our classification method is more effective. This confirms our argument in the last paragraph of Section 2, namely that fuzzy match score is not informative enough to determine the usefulness of the subsentences in a fuzzy match, and that a more comprehensive set of features, as we have explored in this paper, is essential for the discriminative learning-based method to work.

	Fuzzy Match Scores				
	0.50	0.60	0.70	0.80	0.90
BLEU	45.13	45.55	45.58	45.84	45.82
TER	40.99	40.62	40.56	40.29	40.07

Table 5: Performance using fuzzy match score for classification

FM Scores	w. markup	w/o markup
[0,0.5]	37.75	62.24
(0.5,0.6]	40.64	59.36
(0.6,0.7]	40.94	59.06
(0.7,0.8]	46.67	53.33
(0.8,0.9]	54.28	45.72
(0.9,1.0]	44.14	55.86

Table 6: Percentage of training sentences with markup vs without markup grouped by fuzzy match (FM) score ranges

To further validate our assumption, we analyse the training sentences by grouping them according to their fuzzy match score ranges. For each group of sentences, we calculate the percentage of sentences where markup (and respectively without markup) can produce better translations. The statistics are shown in Table 6. We can see that for sentences with fuzzy match scores lower than 0.8, more sentences can be better translated without markup. For sentences where fuzzy match scores are within the range $(0.8, 0.9]$, more sentences can be better translated with markup. However, within the range $(0.9, 1.0]$, surprisingly, actually more sentences receive better translation without markup. This indicates that fuzzy match score is not a good measure to predict whether fuzzy matches are beneficial when used to constrain the translation of an input sentence.

5.5 Contribution of Features

We also investigated the contribution of our different feature sets. We are especially interested in the contribution of dependency features, as they re-

Example 1	
w/o markup	after policy name , type the name of the policy (it shows new host integrity policy by default) .
Translation	在 “ 策略 ” 名称 后面 ， 键入 策略 的 名称 (名称 显示 为 “ 新 主机 完整性 策略 默认 ） 。
w. markup	after policy name <tm translation=“， 键入 策略 名称 （ 默认 显示 “ 新 主机 完整性 策略 ” ） 。”>, type the name of the policy (it shows new host integrity policy by default) .< /tm>
Translation	在 “ 策略 ” 名称 后面 ， 键入 策略 名称 （ 默认 显示 “ 新 主机 完整性 策略 ” ） 。
Reference	在 “ 策略 名称 ” 后面 ， 键入 策略 名称 （ 默认 显示 “ 新 主机 完整性 策略 ” ） 。
Example 2	
w/o markup	changes apply only to the specific scan that you select .
Translation	更改 仅 适用于 特定 扫描 的 规则 。
w. markup	changes apply only to the specific scan that you select <tm translation=“。”>.< /tm>
Translation	更改 仅 适用于 您 选择 的 特定 扫描 。
Reference	更改 只 应用于 您 选择 的 特定 扫描 。

flect whether translation consistency can be captured using syntactic knowledge. The classification and translation results using different features are reported in Table 7. We observe a significant improvement in both classification precision and recall by adding dependency (DEP) features on top of TM and translation features. As a result, the translation quality also significantly improves. This indicates that dependency features which can capture structural and semantic similarities are effective in gauging the usefulness of the phrase pairs derived from the fuzzy matches. Note also that without including the dependency features, our discriminative learning method cannot outperform the BASELINE (cf. Table 3) in terms of translation quality.

	TER	BLEU	P	R
TM+TRANS	40.57	45.51	52.48	27.04
+DEP	39.61	46.46	60.00	32.14

Table 7: Contribution of Features (%)

5.6 Improved Translations

In order to pinpoint the sources of improvements by marking up the input sentence, we performed some manual analysis of the output. We observe that the improvements can broadly be attributed to two reasons: 1) the use of long phrase pairs which are missing in the phrase table, and 2) deterministically using highly reliable phrase pairs.

Phrase-based SMT systems normally impose a limit on the length of phrase pairs for storage and speed considerations. Our method can overcome this limitation by retrieving and reusing long phrase pairs on the fly. A similar idea, albeit from a different perspective, was explored by (Lopez, 2008), where he proposed to construct a phrase table on the fly for each sentence to be translated. Differently from his approach, our method directly translates part of the input sentence using fuzzy matches retrieved on the fly, with the rest of the sentence translated by the pre-trained MT system. We offer some more insights into the advantages of our method by means of a few examples.

Example 1 shows translation improvements by using long phrase pairs. Compared to the reference translation, we can see that for the underlined phrase, the translation without markup contains (i) word ordering errors and (ii) a missing right quotation mark. In Example 2, by specifying the translation of the final punctuation mark, the system correctly translates the relative clause ‘that you select’. The translation of this relative clause is missing when translating the input without markup. This improvement can be partly attributed to the reduction in search errors by specifying the highly reliable translations for phrases in an input sentence.

6 Conclusions and Future Work

In this paper, we introduced a discriminative learning method to tightly integrate fuzzy matches retrieved using translation memory technologies with phrase-based SMT systems to improve translation consistency. We used an SVM classifier to predict whether phrase pairs derived from fuzzy matches could be used to constrain the translation of an in-

put sentence. A number of feature functions including a series of novel dependency features were used to train the classifier. Experiments demonstrated that discriminative learning is effective in improving translation quality and is more informative than the fuzzy match score used in previous research. We report a statistically significant 0.9 absolute improvement in BLEU score using a procedure to promote translation consistency.

As mentioned in Section 2, the potential improvement in sentence-level translation consistency using our method can be attributed to the fact that the translation of new input sentences is closely informed and guided (or constrained) by previously translated sentences using global features such as dependencies. However, it is worth noting that the level of gains in translation consistency is also dependent on the nature of the TM itself; a self-contained coherent TM would facilitate consistent translations. In the future, we plan to investigate the impact of TM quality on translation consistency when using our approach. Furthermore, we will explore methods to promote translation consistency at document level.

Moreover, we also plan to experiment with phrase-by-phrase classification instead of sentence-by-sentence classification presented in this paper, in order to obtain more stable classification results. We also plan to label the training examples using other sentence-level evaluation metrics such as Meteor (Banerjee and Lavie, 2005), and to incorporate features that can measure syntactic similarities in training the classifier, in the spirit of (Owczarzak et al., 2007). Currently, only a standard phrase-based SMT system is used, so we plan to test our method on a hierarchical system (Chiang, 2005) to facilitate direct comparison with (Koehn and Senellart, 2010). We will also carry out experiments on other data sets and for more language pairs.

Acknowledgments

This work is supported by Science Foundation Ireland (Grant No 07/CE/I1142) and part funded under FP7 of the EC within the EuroMatrix+ project (grant No 231720). The authors would like to thank the reviewers for their insightful comments and suggestions.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, MI.

Ergun Biçici and Marc Dymetman. 2008. Dynamic translation memory: Using statistical machine translation to improve translation memory. In *Proceedings of the 9th Internation Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, pages 454–465, Haifa, Israel.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

David Chiang. 2005. A hierarchical Phrase-Based model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, MI.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Yifan He, Yanjun Ma, Josef van Genabith, and Andy Way. 2010. Bridging SMT and TM with translation recommendation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 622–630, Uppsala, Sweden.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 181–184, Detroit, MI.

Philipp Koehn and Jean Senellart. 2010. Convergence of translation memory and statistical machine translation. In *Proceedings of AMTA Workshop on MT Research and the Translation Industry*, pages 21–31, Denver, CO.

Philipp Koehn, Franz Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of the 2003 Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics*, pages 48–54, Edmonton, AB, Canada.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Vol-*

ume Proceedings of the Demo and Poster Sessions, pages 177–180, Prague, Czech Republic.

Vladimir Iosifovich Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C. Weng. 2007. A note on platt's probabilistic outputs for support vector machines. *Machine Learning*, 68(3):267–276.

Adam Lopez. 2008. Tera-scale translation models via pattern matching. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 505–512, Manchester, UK, August.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. Wiley-Interscience, New York, NY.

Franz Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.

Karolina Owczarzak, Josef van Genabith, and Andy Way. 2007. Labelled dependencies in machine translation evaluation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 104–111, Prague, Czech Republic.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of Machine Translation. In *40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA.

John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, pages 61–74.

Richard Sikes. 2007. Fuzzy matching in theory and practice. *Multilingual*, 18(6):39–43.

Michel Simard and Pierre Isabelle. 2009. Phrase-based machine translation in a computer-assisted translation environment. In *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)*, pages 120 – 127, Ottawa, Ontario, Canada.

James Smith and Stephen Clark. 2009. EBMT for SMT: A new EBMT-SMT hybrid. In *Proceedings of the 3rd International Workshop on Example-Based Machine Translation*, pages 3–10, Dublin, Ireland.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas (AMTA-2006)*, pages 223–231, Cambridge, MA, USA.

Lucia Specia, Craig Saunders, Marco Turchi, Zhuoran Wang, and John Shawe-Taylor. 2009. Improving the confidence of machine translation quality estimates. In *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)*, pages 136 – 143, Ottawa, Ontario, Canada.

Andreas Stolcke. 2002. SRILM – An extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 901–904, Denver, CO.

Ventsislav Zhechev and Josef van Genabith. 2010. Seeding statistical machine translation with translation memory output through tree-based structural alignment. In *Proceedings of the Fourth Workshop on Syntax and Structure in Statistical Translation*, pages 43–51, Beijing, China.

Machine Translation System Combination by Confusion Forest

Taro Watanabe and **Eiichiro Sumita**
National Institute of Information and Communications Technology
3-5 Hikaridai, Keihanna Science City, 619-0289 JAPAN
{taro.watanabe,eiichiro.sumita}@nict.go.jp

Abstract

The state-of-the-art system combination method for machine translation (MT) is based on confusion networks constructed by aligning hypotheses with regard to word similarities. We introduce a novel system combination framework in which hypotheses are encoded as a confusion forest, a packed forest representing alternative trees. The forest is generated using syntactic consensus among parsed hypotheses: First, MT outputs are parsed. Second, a context free grammar is learned by extracting a set of rules that constitute the parse trees. Third, a packed forest is generated starting from the root symbol of the extracted grammar through non-terminal rewriting. The new hypothesis is produced by searching the best derivation in the forest. Experimental results on the WMT10 system combination shared task yield comparable performance to the conventional confusion network based method with smaller space.

1 Introduction

System combination techniques take the advantages of consensus among multiple systems and have been widely used in fields, such as speech recognition (Fiscus, 1997; Mangu et al., 2000) or parsing (Henderson and Brill, 1999). One of the state-of-the-art system combination methods for MT is based on confusion networks, which are compact graph-based structures representing multiple hypotheses (Bangalore et al., 2001).

Confusion networks are constructed based on string similarity information. First, one skeleton or backbone sentence is selected. Then, other hypotheses are aligned against the skeleton, forming a lattice with each arc representing alternative word candidates. The alignment method is either model-based (Matusov et al., 2006; He et al., 2008) in which a statistical word aligner is used to compute hypothesis alignment, or edit-based (Jayaraman and Lavie, 2005; Sim et al., 2007) in which alignment is measured by an evaluation metric, such as translation error rate (TER) (Snover et al., 2006). The new translation hypothesis is generated by selecting the best path through the network.

We present a novel method for system combination which exploits the syntactic similarity of system outputs. Instead of constructing a string-based confusion network, we generate a packed forest (Billot and Lang, 1989; Mi et al., 2008) which encodes exponentially many parse trees in a polynomial space. The packed forest, or *confusion forest*, is constructed by merging the MT outputs with regard to their syntactic consensus. We employ a grammar-based method to generate the confusion forest: First, system outputs are parsed. Second, a set of rules are extracted from the parse trees. Third, a packed forest is generated using a variant of Earley's algorithm (Earley, 1970) starting from the unique root symbol. New hypotheses are selected by searching the best derivation in the forest. The grammar, a set of rules, is limited to those found in the parse trees. Spurious ambiguity during the generation step is further reduced by encoding the tree local contextual information in each non-terminal symbol, such as parent and sibling labels, using the state representation in Earley's algorithm.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1249–1257,
Portland, Oregon, June 19-24, 2011.

Experiments were carried out for the system combination task of the fifth workshop on statistical machine translation (WMT10) in four directions, {Czech, French, German, Spanish}-to-English (Callison-Burch et al., 2010), and we found comparable performance to the conventional confusion network based system combination in two language pairs, and statistically significant improvements in the others.

First, we will review the state-of-the-art method which is a system combination framework based on confusion networks (§2). Then, we will introduce a novel system combination method based on confusion forest (§3) and present related work in consensus translations (§4). Experiments are presented in Section 5 followed by discussion and our conclusion.

2 Combination by Confusion Network

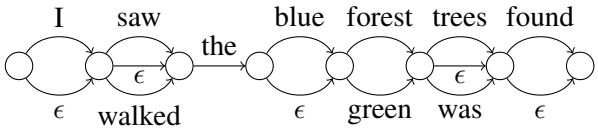
The system combination framework based on confusion network starts from computing pairwise alignment between hypotheses by taking one hypothesis as a reference. Matusov et al. (2006) employs a model based approach in which a statistical word aligner, such as GIZA++ (Och and Ney, 2003), is used to align the hypotheses. Sim et al. (2007) introduced TER (Snover et al., 2006) to measure the edit-based alignment.

Then, one hypothesis is selected, for example by employing a minimum Bayes risk criterion (Sim et al., 2007), as a skeleton, or a backbone, which serves as a building block for aligning the rest of the hypotheses. Other hypotheses are aligned against the skeleton using the pairwise alignment. Figure 1(b) illustrates an example of a confusion network constructed from the four hypotheses in Figure 1(a), assuming the first hypothesis is selected as our skeleton. The network consists of several arcs, each of which represents an alternative word at that position, including the empty symbol, ϵ.

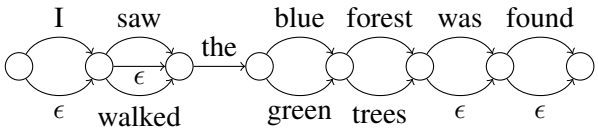
This pairwise alignment strategy is prone to spurious insertions and repetitions due to alignment errors such as in Figure 1(a) in which "green" in the third hypothesis is aligned with "forest" in the skeleton. Rosti et al. (2008) introduces an incremental method so that hypotheses are aligned incrementally to the growing confusion network, not only the skeleton hypothesis. In our example, "green trees" is aligned with "blue forest" in Figure 1(c).

*	I	saw	the		forest		
	I	walked	the	blue	forest		
	I	saw	the		green	trees	
			the		forest	was	found

(a) Pairwise alignment using the first starred hypothesis as a skeleton.



(b) Confusion network from (a)



(c) Incrementally constructed confusion network

Figure 1: An example confusion network construction

The confusion network construction is largely influenced by the skeleton selection, which determines the global word reordering of a new hypothesis. For example, the last hypothesis in Figure 1(a) has a passive voice grammatical construction while the others are active voice. This large grammatical difference may produce a longer sentence with spuriously inserted words, as in "I saw the blue trees was found" in Figure 1(c). Rosti et al. (2007b) partially resolved the problem by constructing a large network in which each hypothesis was treated as a skeleton and the multiple networks were merged into a single network.

3 Combination by Confusion Forest

The confusion network approach to system combination encodes multiple hypotheses into a compact lattice structure by using word-level consensus. Likewise, we propose to encode multiple hypotheses into a confusion forest, which is a packed forest which represents multiple parse trees in a polynomial space (Billot and Lang, 1989; Mi et al., 2008) Syntactic consensus is realized by sharing tree frag-

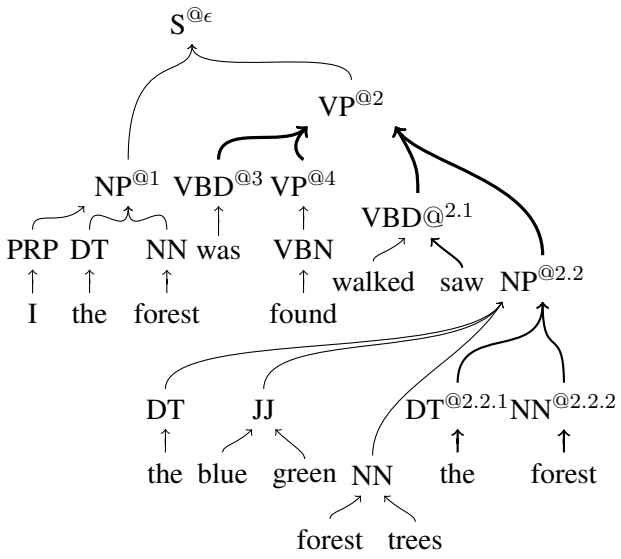


Figure 2: An example packed forest representing hypotheses in Figure 1(a).

ments among parse trees. The forest is represented as a hypergraph which is exploited in parsing (Klein and Manning, 2001; Huang and Chiang, 2005) and machine translation (Chiang, 2007; Huang and Chiang, 2007).

More formally, a hypergraph is a pair $\langle V, E\rangle$ where V is the set of nodes and E is the set of hyperedges. Each node in V is represented as $X^{@p}$ where $\mathrm{X} \in \mathcal{N}$ is a non-terminal symbol and p is an address (Shieber et al., 1995) that encapsulates each node id relative to its parent. The root node is given the address ϵ and the address of the first child of node p is given $p.1$. Each hyperedge $e \in E$ is represented as a pair $\langle head(e), tails(e)\rangle$ where $head(e) \in V$ is a head node and $tails(e) \in V^*$ is a list of tail nodes, corresponding to the left-hand side and the right-hand side of an instance of a rule in a CFG, respectively. Figure 2 presents an example packed forest for the parsed hypotheses in Figure 1(a). For example, $\mathrm{VP}^{@2}$ has two hyperedges, $\langle \mathrm{VP}^{@2}, (\mathrm{VBD}^{@3}, \mathrm{VP}^{@4})\rangle$ and $\langle \mathrm{VP}^{@2}, (\mathrm{VBD}^{@2.1}, \mathrm{NP}^{@2.2})\rangle$, leading to different derivations where the former takes the grammatical construction in passive voice while the latter in active voice.

Given system outputs, we employ the following grammar based approach for constructing a confusion forest: First, MT outputs are parsed. Second, a grammar is learned by treating each hyperedge as an instance of a CFG rule. Third, a forest is generated from the unique root symbol of the extracted grammar through non-terminal rewriting.

Initialization:

$$\frac{}{[\mathrm{TOP} \rightarrow \bullet \mathrm{S}, 0] : \bar{1}}$$

Scan:

$$\frac{[\mathrm{X} \rightarrow \alpha \bullet x\beta, h] : u}{[\mathrm{X} \rightarrow \alpha x \bullet \beta, h] : u}$$

Predict:

$$\frac{[\mathrm{X} \rightarrow \alpha \bullet \mathrm{Y}\beta, h]}{[\mathrm{Y} \rightarrow \bullet\gamma, h+1] : u} \quad \mathrm{Y} \xrightarrow{u} \gamma \in \mathcal{G}, h < H$$

Complete:

$$\frac{[\mathrm{X} \rightarrow \alpha \bullet \mathrm{Y}\beta, h] : u \quad [\mathrm{Y} \rightarrow \gamma\bullet, h+1] : v}{[\mathrm{X} \rightarrow \alpha\mathrm{Y} \bullet \beta, h] : u \otimes v}$$

Goal:

$$[\mathrm{TOP} \rightarrow \mathrm{S}\bullet, 0]$$

Figure 3: The deductive system for Earley's generation algorithm

3.1 Forest Generation

Given the extracted grammar, we apply a variant of Earley's algorithm (Earley, 1970) which can generate strings in a left-to-right manner from the unique root symbol, TOP. Figure 3 presents the deductive inference rules (Goodman, 1999) for our generation algorithm. We use capital letters $\mathrm{X} \in \mathcal{N}$ to denote non-terminals and $x \in \mathcal{T}$ for terminals. Lowercase Greek letters α, β and γ are strings of terminals and non-terminals $(\mathcal{T} \cup \mathcal{N})^*$. u and v are weights associated with each item.

The major difference compared to Earley's parsing algorithm is that we ignore the terminal span information each non-terminal covers and keep track of the height of derivations by h. The scanning step will always succeed by moving the dot to the right. Combined with the prediction and completion steps, our algorithm may potentially generate a spuriously deep forest. Thus, the height of the forest is constrained in the prediction step not to exceed H, which is empirically set to 1.5 times the maximum

height of the parsed system outputs.

3.2 Tree Annotation

The grammar compiled from the parsed trees is local in that it can represent a finite number of sentences translated from a specific input sentence. Although its coverage is limited, our generation algorithm may yield a spuriously large forest. As a way to reduce spurious ambiguities, we relabel the non-terminal symbols assigned to each parse tree before extracting rules.

Here, we replace each non-terminal symbol by the state representation of Earley's algorithm corresponding to the sequence of prediction steps starting from TOP. Figure 4(a) presents an example parse tree with each symbol replaced by the Earley's state in Figure 4(b). For example, the label for VBD is replaced by •S + NP : •VP + •VBD : NP which corresponds to the prediction steps of TOP → •S, S → NP • VP and VP → •VBD NP. The context represented in the Earley's state is further limited by the vertical and horizontal Markovization (Klein and Manning, 2003). We define the vertical order v in which the label is limited to memorize only v previous prediction steps. For instance, setting $v = 1$ yields NP : •VP + •VBD : NP in our example. Likewise, we introduce the horizontal order h which limits the number of sibling labels memorized on the left and the right of the dotted label. Limiting $h = 1$ implies that each deductive step is encoded with at most three symbols.

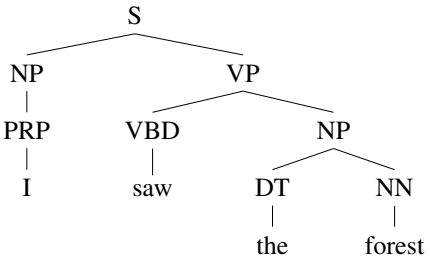
No limits in the horizontal and vertical Markovization orders implies memorizing of all the deductions and yields a confusion forest representing the union of parse trees through the grammar collection and the generation processes. More relaxed horizontal orders allow more reordering of subtrees in a confusion forest by discarding the sibling context in each prediction step. Likewise, constraining the vertical order generates a deeper forest by ignoring the sequence of symbols leading to a particular node.

3.3 Forest Rescoring

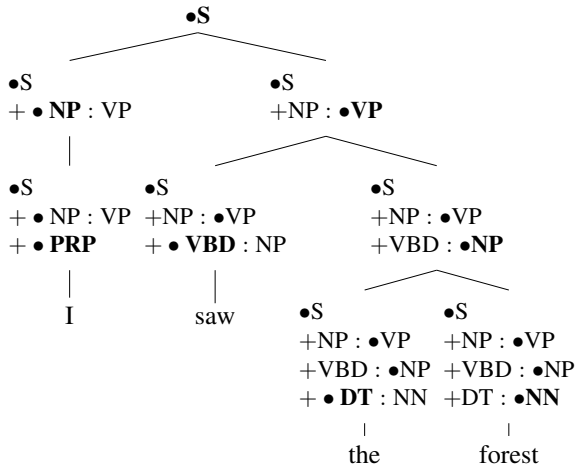
From the packed forest F, new k-best derivations are extracted from all possible derivations D by efficient forest-based algorithms for k-best parsing (Huang and Chiang, 2005). We use a linear combination of features as our objective function to seek for the best derivation $\hat{d}$:

$$\hat{d} = \arg\max_{d \in D} \mathbf{w}^\top \cdot \mathbf{h}(d, F) \quad (1)$$

where $\mathbf{h}(d, F)$ is a set of feature functions scaled by weight vector $\mathbf{w}$. We use cube-pruning (Chiang, 2007; Huang and Chiang, 2007) to approximately intersect with non-local features, such as n-gram language models. Then, k-best derivations are extracted from the rescored forest using algorithm 3 of Huang and Chiang (2005).



(a) A parse tree for "I saw the forest"



(b) Earley's state annotated tree for (a). The sub-labels in boldface indicate the original labels.

Figure 4: Label annotation by Earley's alsogirhtm state

4 Related Work

Consensus translations have been extensively studied with many granularities. One of the simplest forms is a sentence-based combination in which hypotheses are simply reranked without merging (Nomoto, 2004). Frederking and Nirenburg (1994)

proposed a phrasal combination by merging hypotheses in a chart structure, while others depended on confusion networks, or similar structures, as a building block for merging hypotheses at the word level (Bangalore et al., 2001; Matusov et al., 2006; He et al., 2008; Jayaraman and Lavie, 2005; Sim et al., 2007). Our work is the first to explicitly exploit syntactic similarity for system combination by merging hypotheses into a syntactic packed forest. The confusion forest approach may suffer from parsing errors such as the confusion network construction influenced by alignment errors. Even with parsing errors, we can still take a tree fragment-level consensus as long as a parser is consistent in that similar syntactic mistakes would be made for similar hypotheses.

Rosti et al. (2007a) describe a re-generation approach to consensus translation in which a phrasal translation table is constructed from the MT outputs aligned with an input source sentence. New translations are generated by decoding the source sentence again using the newly extracted phrase table. Our grammar-based approach can be regarded as a re-generation approach in which an off-the-shelf monolingual parser, instead of a word aligner, is used to annotate syntactic information to each hypothesis, then, a new translation is generated from the merged forest, not from the input source sentence through decoding. In terms of generation, our approach is an instance of statistical generation (Langkilde and Knight, 1998; Langkilde, 2000). Instead of generating forests from semantic representations (Langkilde, 2000), we generate forests from a CFG encoding the consensus among parsed hypotheses.

Liu et al. (2009) present joint decoding in which a translation forest is constructed from two distinct MT systems, tree-to-string and string-to-string, by merging forest outputs. Their merging method is either translation-level in which no new translation is generated, or derivation-level in that the rules sharing the same left-hand-side are used in both systems. While our work is similar in that a new forest is constructed by sharing rules among systems, although their work involves no consensus translation and requires structures internal to each system such as model combinations (DeNero et al., 2010).

		cz-en	de-en	es-en	fr-en
# of systems		6	16	8	14
avg. words	tune	10.6K	10.9K	10.9K	11.0K
	test	50.5K	52.1K	52.1K	52.4K
sentences	tune	455			
	test	2,034			

Table 1: WMT10 system combination tuning/testing data

5 Experiments

5.1 Setup

We ran our experiments for the WMT10 system combination task usinge four language pairs, {Czech, French, German, Spanish}-to-English (Callison-Burch et al., 2010). The data is summarized in Table 1. The system outputs are retokenized to match the Penn-treebank standard, parsed by the Stanford Parser (Klein and Manning, 2003), and lower-cased.

We implemented our confusion forest system combination using an in-house developed hypergraph-based toolkit *cicada* which is motivated by generic weighted logic programming (Lopez, 2009), originally developed for a synchronous-CFG based machine translation system (Chiang, 2007). Input to our system is a collection of hypergraphs, a set of parsed hypotheses, from which rules are extracted and a new forest is generated as described in Section 3. Our baseline, also implemented in *cicada*, is a confusion network-based system combination method (§2) which incrementally aligns hypotheses to the growing network using TER (Rosti et al., 2008) and merges multiple networks into a large single network. After performing epsilon removal, the network is transformed into a forest by parsing with monotone rules of S $\rightarrow$ X, S $\rightarrow$ S X and X $\rightarrow$ x. k-best translations are extracted from the forest using the forest-based algorithms in Section 3.3.

5.2 Features

The feature weight vector $\mathbf{w}$ in Equation 1 is tuned by MERT over hypergraphs (Kumar et al., 2009).

We use three lower-cased 5-gram language mod-

els $h^i_{lm}(d)$: English Gigaword Fourth edition[1], the English side of French-English 10^9 corpus and the news commentary English data[2]. The count based features $h_t(d)$ and $h_e(d)$ count the number of terminals and the number of hyperedges in d, respectively. We employ M confidence measures $h^m_s(d)$ for M systems, which basically count the number of rules used in d originally extracted from mth system hypothesis (Rosti et al., 2007a).

Following Macherey and Och (2007), BLEU (Papineni et al., 2002) correlations are also incorporated in our system combination. Given M system outputs $\mathbf{e}_1 ... \mathbf{e}_M$, M BLEU scores are computed for d using each of the system outputs $\mathbf{e}_m$ as a reference

$$h^m_b(d) = BP(\mathbf{e}, \mathbf{e}_m) \cdot \exp\left(\frac{1}{4}\sum_{n=1}^{4} \log \rho_n(\mathbf{e}, \mathbf{e}_m)\right)$$

where $\mathbf{e} = \text{yield}(d)$ is a terminal yield of d, $BP(\cdot)$ and $\rho_n(\cdot)$ respectively denote brevity penalty and n-gram precision. Here, we use approximated unclipped n-gram counts (Dreyer et al., 2007) for computing $\rho_n(\cdot)$ with a compact state representation (Li and Khudanpur, 2009).

Our baseline confusion network system has an additional penalty feature, $h_p(m)$, which is the total edits required to construct a confusion network using the mth system hypothesis as a skeleton, normalized by the number of nodes in the network (Rosti et al., 2007b).

5.3 Results

Table 2 compares our confusion forest approach (CF) with different orders, a confusion network (CN) and max/min systems measured by BLEU (Papineni et al., 2002). We vary the horizontal orders, $h = 1, 2, \infty$ with vertical orders of $v = 3, 4, \infty$. Systems without statistically significant differences from the best result ($p < 0.05$) are indicated by bold face. Setting $v = \infty$ and $h = \infty$ achieves comparable performance to CN. Our best results in three languages come from setting $v = \infty$ and $h = 2$, which favors little reordering of phrasal structures. In general, lower horizontal and vertical order leads to lower BLEU.

[1]LDC catalog No. LDC2009T13

[2]Those data are available from `http://www.statmt.org/wmt10/`.

language	cz-en	de-en	es-en	fr-en
system min	14.09	15.62	21.79	16.79
max	23.44	24.10	29.97	**29.17**
CN	23.70	24.09	**30.45**	**29.15**
CF$_{v=\infty,h=\infty}$	**24.13**	24.18	**30.41**	**29.57**
CF$_{v=\infty,h=2}$	**24.14**	**24.58**	**30.52**	28.84
CF$_{v=\infty,h=1}$	**24.01**	23.91	**30.46**	**29.32**
CF$_{v=4,h=\infty}$	**23.93**	23.57	29.88	28.71
CF$_{v=4,h=2}$	**23.82**	22.68	29.92	28.83
CF$_{v=4,h=1}$	**23.77**	21.42	30.10	28.32
CF$_{v=3,h=\infty}$	23.38	23.34	29.81	27.34
CF$_{v=3,h=2}$	23.30	23.95	30.02	28.19
CF$_{v=3,h=1}$	23.23	21.43	29.27	26.53

Table 2: Translation results in lower-case BLEU. CN for confusion network and CF for confusion forest with different vertical (v) and horizontal (h) Markovization order.

language	cz-en	de-en	es-en	fr-en
rerank	29.40	32.32	36.83	36.59
CN	38.52	34.97	47.65	46.37
CF$_{v=\infty,h=\infty}$	30.51	34.07	38.69	38.94
CF$_{v=\infty,h=2}$	30.61	34.25	38.87	39.10
CF$_{v=\infty,h=1}$	31.09	34.65	39.27	39.51
CF$_{v=4,h=\infty}$	30.86	34.19	39.17	39.39
CF$_{v=4,h=2}$	30.96	34.32	39.35	39.57
CF$_{v=4,h=1}$	31.44	34.62	39.69	39.90
CF$_{v=3,h=\infty}$	31.03	34.30	39.29	39.57
CF$_{v=3,h=2}$	31.25	34.97	39.61	40.00
CF$_{v=3,h=1}$	31.55	34.60	39.72	39.97

Table 3: Oracle lower-case BLEU

Table 3 presents oracle BLEU achievable by each combination method. The gains achievable by the CF over simple reranking are small, at most 2-3 points, indicating that small variations are encoded in confusion forests. We also observed that a lower horizontal and vertical order leads to better BLEU potentials. As briefly pointed out in Section 3.2, the higher horizontal and vertical order implies more faithfulness to the original parse trees. Introducing new tree fragments to confusion forests leads to new phrasal translations with enlarged forests, as presented in Table 4, measured by the average number

lang	cz-en	de-en	es-en	fr-en
CN	2,222.68	47,231.20	2,932.24	11,969.40
lattice	1,723.91	41,403.90	2,330.04	10,119.10
$CF_{v=\infty}$	230.08	540.03	262.30	386.79
$CF_{v=4}$	254.45	651.10	302.01	477.51
$CF_{v=3}$	286.01	802.79	349.21	575.17

Table 4: Hypegraph size measured by the average number of hyperedges ($h = 1$ for CF). "lattice" is the average number of edges in the original CN.

of hyperedges[3]. The larger potentials do not imply better translations, probably due to the larger search space with increased search errors. We also conjecture that syntactic variations were not captured by the n-gram like string-based features in Section 5.2, therefore resulting in BLEU loss, which will be investigated in future work.

In contrast, CN has more potential for generating better translations, with the exception of the German-to-English direction, with scores that are usually 10 points better than simple sentence-wise reranking. The low potential in German should be interpreted in the light of the extremely large confusion network in Table 4. We postulate that the divergence in German hypotheses yields wrong alignments, and therefore amounts to larger networks with incorrect hypotheses. Table 4 also shows that CN produces a forest that is an order of magnitude larger than those created by CFs. Although we cannot directly relate the runtime and the number of hyperedges in CN and CFs, since the shape of the forests are different, CN requires more space to encode the hypotheses than those by CFs.

Table 5 compares the average length of the minimum/maximum hypothesis that each method can produce. CN may generate shorter hypotheses, whereby CF prefers longer hypotheses as we decrease the vertical order. Large divergence is also observed for German, such as for hypergraph size.

6 Conclusion

We presented a confusion forest based method for system combination in which system outputs are merged into a packed forest using their syntactic similarity. The forest construction is treated as a generation from a CFG compiled from the parsed outputs. Our experiments indicate comparable performance to a strong confusion network baseline with smaller space, and statistically significant gains in some language pairs.

To our knowledge, this is the first work to directly introduce syntactic consensus to system combination by encoding multiple system outputs into a single forest structure. We believe that the confusion forest based approach to system combination has future exploration potential. For instance, we did not employ syntactic features in Section 5.2 which would be helpful in discriminating hypotheses in larger forests. We would also like to analyze the trade-offs, if any, between parsing errors and confusion forest constructions by controlling the parsing qualities. As an alternative to the grammar-based forest generation, we are investigating an edit distance measure for tree alignment, such as tree edit distance (Bille, 2005) which basically computes insertion/deletion/replacement of nodes in trees.

[3]We measure the hypergraph size before intersecting with non-local features, like n-gram language models.

language		cz-en	de-en	es-en	fr-en
system	avg.	24.84	25.62	25.63	25.75
CN	min	11.09	3.39	12.27	7.94
	max	33.69	40.65	33.22	36.27
$CF_{v=\infty}$	min	15.97	10.88	17.67	16.62
	max	35.20	47.20	35.28	37.94
$CF_{v=4}$	min	15.52	10.58	17.02	15.85
	max	37.11	53.67	38.56	42.64
$CF_{v=3}$	min	15.15	10.34	16.54	15.30
	max	39.88	68.45	42.85	49.55

Table 5: Average min/max hypothesis length producible by each method ($h = 1$ for CF).

Acknowledgments

We would like to thank anonymous reviewers and our colleagues for helpful comments and discussion.

References

Srinivas Bangalore, German Bordel, and Giuseppe Riccardi. 2001. Computing consensus translation from multiple machine translation systems. In *Proceedings of Automatic Speech Recognition and Understanding (ASRU), 2001*, pages 351 – 354.

Philip Bille. 2005. A survey on tree edit distance and related problems. *Theor. Comput. Sci.*, 337:217–239, June.

Sylvie Billot and Bernard Lang. 1989. The structure of shared forests in ambiguous parsing. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 143–151, Vancouver, British Columbia, Canada, June.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar Zaidan. 2010. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53, Uppsala, Sweden, July. Revised August 2010.

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

John DeNero, Shankar Kumar, Ciprian Chelba, and Franz Och. 2010. Model combination for machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 975–983, Los Angeles, California, June.

Markus Dreyer, Keith Hall, and Sanjeev Khudanpur. 2007. Comparing reordering constraints for smt using efficient bleu oracle computation. In *Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 103–110, Rochester, New York, April.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the Association for Computing Machinery*, 13:94–102, February.

J.G. Fiscus. 1997. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover). In *Proceedings of Automatic Speech Recognition and Understanding (ASRU), 1997*, pages 347 –354, December.

Robert Frederking and Sergei Nirenburg. 1994. Three heads are better than one. In *Proceedings of the fourth conference on Applied natural language processing*, pages 95–100, Morristown, NJ, USA.

Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25:573–605, December.

Xiaodong He, Mei Yang, Jianfeng Gao, Patrick Nguyen, and Robert Moore. 2008. Indirect-HMM-based hypothesis alignment for combining outputs from machine translation systems. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 98–107, Honolulu, Hawaii, October.

John C. Henderson and Eric Brill. 1999. Exploiting diversity in natural language processing: Combining parsers. In *Proceedings of the Fourth Conference on Empirical Methods in Natural Language Processing*, pages 187–194.

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

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151, Prague, Czech Republic, June.

Shyamsundar Jayaraman and Alon Lavie. 2005. Multi-engine machine translation guided by explicit word matching. In *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, ACL '05, pages 101–104, Morristown, NJ, USA.

Dan Klein and Christopher D. Manning. 2001. Parsing and hypergraphs. In *Proceedings of the Seventh International Workshop on Parsing Technologies (IWPT-2001)*, pages 123–134.

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

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

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL-36, pages 704–710, Morristown, NJ, USA.

Irene Langkilde. 2000. Forest-based statistical sentence generation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 170–177, San Francisco, CA, USA.

Zhifei Li and Sanjeev Khudanpur. 2009. Efficient extraction of oracle-best translations from hypergraphs. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 9–12, Boulder, Colorado, June.

Yang Liu, Haitao Mi, Yang Feng, and Qun Liu. 2009. Joint decoding with multiple translation models. In

Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pages 576–584, Suntec, Singapore, August.

Adam Lopez. 2009. Translation as weighted deduction. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 532–540, Athens, Greece, March.

Wolfgang Macherey and Franz J. Och. 2007. An empirical study on computing consensus translations from multiple machine translation systems. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 986–995, Prague, Czech Republic, June.

Lidia Mangu, Eric Brill, and Andreas Stolcke. 2000. Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech & Language*, 14(4):373 – 400.

Evgeny Matusov, Nicola Ueffing, and Hermann Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 33–40.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL-08: HLT*, pages 192–199, Columbus, Ohio, June.

Tadashi Nomoto. 2004. Multi-engine machine translation with voted language model. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 494–501, Barcelona, Spain, July.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July.

Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie Dorr. 2007a. Combining outputs from multiple machine translation systems. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 228–235, Rochester, New York, April.

Antti-Veikko Rosti, Spyros Matsoukas, and Richard Schwartz. 2007b. Improved word-level system combination for machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 312–319, Prague, Czech Republic, June.

Antti-Veikko Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz. 2008. Incremental hypothesis alignment for building confusion networks with application to machine translation system combination. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 183–186, Columbus, Ohio, June.

Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1–2):3–36, July–August.

K.C. Sim, W.J. Byrne, M.J.F. Gales, H. Sahbi, and P.C. Woodland. 2007. Consensus network decoding for statistical machine translation system combination. In *Proceedings of Acoustics, Speech and Signal Processing (ICASSP), 2007*, volume 4, pages IV–105 –IV–108, April.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Hypothesis Mixture Decoding for Statistical Machine Translation

Nan Duan,
School of Computer Science and Technology
Tianjin University
Tianjin, China
v-naduan@microsoft.com

Mu Li, and **Ming Zhou**
Natural Language Computing Group
Microsoft Research Asia
Beijing, China
{muli,mingzhou}@microsoft.com

Abstract

This paper presents *hypothesis mixture decoding* (HM decoding), a new decoding scheme that performs translation reconstruction using hypotheses generated by multiple translation systems. HM decoding involves two decoding stages: first, each component system decodes independently, with the explored search space kept for use in the next step; second, a new search space is constructed by composing existing hypotheses produced by all component systems using a set of rules provided by the HM decoder itself, and a new set of model independent features are used to seek the final best translation from this new search space. Few assumptions are made by our approach about the underlying component systems, enabling us to leverage SMT models based on arbitrary paradigms. We compare our approach with several related techniques, and demonstrate significant BLEU improvements in large-scale Chinese-to-English translation tasks.

1 Introduction

Besides tremendous efforts on constructing more complicated and accurate models for statistical machine translation (SMT) (Och and Ney, 2004; Chiang, 2005; Galley et al., 2006; Shen et al., 2008; Chiang 2010), many researchers have concentrated on the approaches that improve translation quality using information between hypotheses from one or more SMT systems as well.

System combination is built on top of the *N*-best outputs generated by multiple component systems (Rosti et al., 2007; He et al., 2008; Li et al., 2009b) which aligns multiple hypotheses to build confusion networks as new search spaces, and outputs the highest scoring paths as the final translations. *Consensus decoding*, on the other hand, can be based on either single or multiple systems: single system based methods (Kumar and Byrne, 2004; Tromble et al., 2008; DeNero et al., 2009; Kumar et al., 2009) re-rank translations produced by a single SMT model using either *n*-gram posteriors or expected *n*-gram counts. Because hypotheses generated by a single model are highly correlated, improvements obtained are usually small; recently, dedicated efforts have been made to extend it from single system to multiple systems (Li et al., 2009a; DeNero et al., 2010; Duan et al., 2010). Such methods select translations by optimizing consensus models over the combined hypotheses using all component systems' posterior distributions.

Although these two types of approaches have shown consistent improvements over the standard Maximum a Posteriori (MAP) decoding scheme, most of them are implemented as post-processing procedures over translations generated by MAP decoders. In this sense, the work of Li et al. (2009a) is different in that both partial and full hypotheses are re-ranked during the decoding phase directly using consensus between translations from different SMT systems. However, their method does not change component systems' search spaces.

This paper presents *hypothesis mixture decoding* (HM decoding), a new decoding scheme that performs translation reconstruction using hypotheses generated by multiple component systems. HM decoding involves two decoding stages: first, each component system decodes the source sentence independently, with the explored search space kept for use in the next step; second, a new search space is constructed by composing existing hypo-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1258–1267,
Portland, Oregon, June 19-24, 2011.

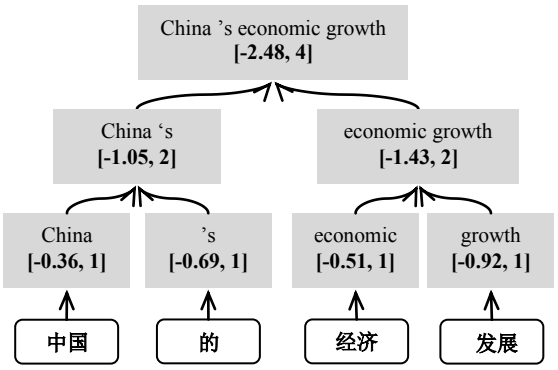


Figure 1: A decoding example of a phrase-based SMT system. Each hypothesis is annotated with a feature vector, which includes a logarithmic probability feature and a word count feature.

theses produced by all component systems using a set of rules provided by the HM decoder itself, and a new set of component model independent features are used to seek the final best translation from this new constructed search space.

We evaluate by combining two SMT models with state-of-the-art performances on the NIST Chinese-to-English translation tasks. Experimental results show that our approach outperforms the best component SMT system by up to 2.11 BLEU points. Consistent improvements can be observed over several related decoding techniques as well, including word-level system combination, collaborative decoding and model combination.

2 Hypothesis Mixture Decoding

2.1 Motivation and Overview

SMT models based on different paradigms have emerged in the last decade using fairly different levels of linguistic knowledge. Motivated by the success of system combination research, the key contribution of this work is to make more effective use of the extended search spaces from different SMT models in decoding phase directly, rather than just post-processing their final outputs. We first begin with a brief review of single system based SMT decoding, and then illustrate major challenges to this end.

Given a source sentence f, an SMT decoder seeks for a target translation e that best matches f as its translation by maximizing the following conditional probability:

$$P(e|f) = \frac{exp\{\sum_{r\in\mathcal{D}(e)}\theta\cdot\phi(r)\}}{\sum_{e'\in\mathcal{H}(f)}exp\{\sum_{r\in\mathcal{D}(e')}\theta\cdot\phi(r)\}}$$

where $\phi(\cdot)$ is the feature vector that includes a set of system specific features, θ is the weight vector, $\mathcal{D}(e)$ is a derivation that can yield e and is defined as a sequence of translation rule applications $\{r\}$. Figure 1 illustrates a decoding example, in which the final translation is generated by recursively composing partial hypotheses that cover different ranges of the source sentence until the whole input sentence is fully covered, and the feature vector of the final translation is the aggregation of feature vectors of all partial hypotheses used.[1]

However, hypotheses generated by different SMT systems cannot be combined directly to form new translations because of two major issues:

The first one is the heterogeneous structures of different SMT models. For example, a string-to-tree system cannot use hypotheses generated by a phrase-based system in decoding procedure, as such hypotheses are based on flat structures, which cannot provide any additional information needed in the syntactic model.

The second one is the incompatible feature spaces of different SMT models. For example, even if a phrase-based system can use the lexical forms of hypotheses generated by a syntax-based system without considering syntactic structures, the feature vectors of these hypotheses still cannot be aggregated together in any trivial way, because the feature sets of SMT models based on different paradigms are usually inconsistent.

To address these two issues discussed above, we propose HM decoding that performs translation reconstruction using hypotheses generated by multiple component systems.[2] Our method involves two decoding stages depicted as follows:

1. *Independent decoding* stage, in which each component system decodes input sentences independently based on its own model and search algorithm, and the explored search spaces (translation forests) are kept for use in the next stage.

[1] There are also features independent of translation derivations, such as the language model feature.

[2] In this paper, we will constrain our discussions within CKY-style decoders, in which we find translations for all spans of the source sentence. Although standard implementations of phrase-based decoders fall out of this scope, they can be still re-written to work in the CKY-style bottom-up manner at the cost of 1) only BTG-style reordering allowed, and 2) higher time complexity. As a result, any phrase-based SMT system can be used as a component in our HM decoding method.

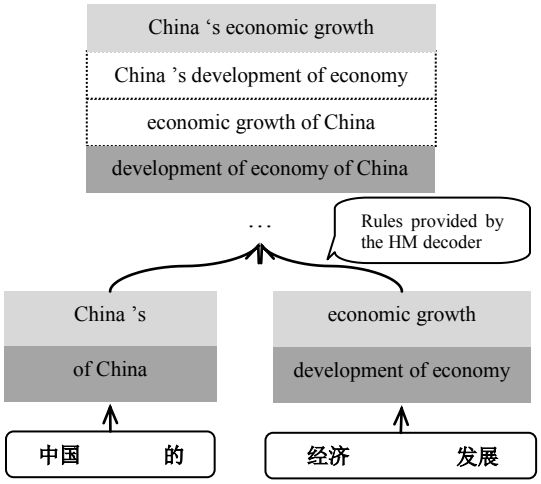


Figure 2: An example of HM decoding, in which the translations surrounded by the dotted lines are newly generated hypotheses. Hypotheses light-shaded come from a phrase-based system, and hypotheses dark-shaded come from a syntax-based system.

2. *HM decoding* stage, where a mixture search space is constructed for translation derivations by composing partial hypotheses generated by all component systems, and a new decoding model with a set of enriched feature functions are used to seek final translations from this newly generated search space.

HM decoding can use lexicalized hypotheses of arbitrary SMT models to derive translation, and a set of component model independent features are used to compute translation confidence. We discuss mixture search space construction, details of model and feature designs as well as HM decoding algorithms in Section 2.2, 2.3 and 2.4 respectively.

2.2 Mixture Search Space Construction

Let $\mathcal{M}_1, \dots, \mathcal{M}_N$ denote N component MT systems, f_i^j denote the span of a source sentence f starting at position i and ending at position j. We use $\mathcal{H}_n(f_i^j)$ denoting the search space of f_i^j predicted by $\mathcal{M}_n$, and $\mathcal{H}(f_i^j)$ denoting the mixture search space of f_i^j constructed by the HM decoder, which is defined recursively as follows:

- $\mathcal{H}_n(f_i^j) \subset \mathcal{H}(f_i^j)$. This rule adds all component systems' search spaces into the mixture search space for use in HM decoding. Thus hypotheses produced by all component systems are still available to the HM decoder.
- $r(e_{p_1}^{q_1}, \dots, e_{p_k}^{q_k}) \subset \mathcal{H}(f_i^j)$, in which $i \le p_k \le q_k \le j$ and $e_{p_k}^{q_k} \subset \mathcal{H}(f_{p_k}^{q_k})$. r is a translation rule provided by HM decoder that composes a new hypothesis using smaller hypotheses in the search spaces $\mathcal{H}(f_{p_1}^{q_1}), \dots, \mathcal{H}(f_{p_k}^{q_k})$. These rules further extend $\mathcal{H}(f_i^j)$ with hypotheses generated by the HM decoder itself.

Figure 2 shows an example of HM decoding, in which hypotheses generated by two SMT systems are used together to compose new translations. Since search space pruning is the indispensable procedure for all SMT systems, we will omit its explicit expression in the following descriptions and algorithms for convenience.

2.3 Models and Features

Following the common practice in SMT research, we use a linear model to formulate the preference of translation hypotheses in the mixture search space $\mathcal{H}(f)$. Formally, we are to find a translation $\hat{e}$ that maximizes the weighted linear combination of a set of real-valued features as follows:

$$\hat{e} = \underset{e \in \mathcal{H}(f)}{argmax} \left\{ \sum_i \lambda_i \cdot h_i(e, f) \right\}$$

where $h_i(e, f)$ is an HM decoding feature with its corresponding feature weight λ_i.

In this paper, the HM decoder does not assume the availability of any internal knowledge of the underlying component systems. The HM decoding features are independent of component models as well, which fall into two categories:

The first category contains a set of consensus-based features, which are inspired by the success of consensus decoding approaches. These features are described in details as follows:

1) $h_{\mathcal{H}_n(f)}(e, f)$: the n-gram posterior feature of e computed based on the component search space $\mathcal{H}_n(f)$ generated by $\mathcal{M}_n$:

$$h_{\mathcal{H}_n(f)}(e, f) = \sum_{\omega \in e} \#_\omega(e) p(\omega | \mathcal{H}_n(f))$$

$p(\omega | \mathcal{H}_n(f)) = \sum_{e' \in \mathcal{H}_n(f)} \delta_\omega(e') P_n(e' | f)$ is the posterior probability of an n-gram ω in $\mathcal{H}_n(f)$, $\#_\omega(e)$ is the number of times that ω occurs in e, $\delta_\omega(e)$ equals to 1 when ω occurs in e, and 0 otherwise.

2) $h_{\mathcal{H}_n^S(f)}(e^S, f)$: the stemmed *n*-gram posterior feature of e computed based on the stemmed component search space $\mathcal{H}_n^S(f)$. A word stem dictionary that includes 22,660 entries is used to convert e and $\mathcal{H}_n(f)$ into their stem forms e^S and $\mathcal{H}_n^S(f)$ by replacing each word into its stem form. This feature is computed similarly to that of $h_{\mathcal{H}_n(f)}(e, f)$.

3) $h_{\mathcal{H}(f)}(e, f)$: the *n*-gram posterior feature of e computed based on the mixture search space $\mathcal{H}(f)$ generated by the HM decoder:

$$h_{\mathcal{H}(f)}(e, f) = \sum_{\omega \in e} \#_\omega(e) p(\omega|\mathcal{H}(f))$$

$p(\omega|\mathcal{H}(f)) = \sum_{e' \in \mathcal{H}(f)} \delta_\omega(e') P(e'|f)$ is the posterior probability of an *n*-gram ω in $\mathcal{H}(f)$, $P(e'|f)$ is the posterior probability of one translation e' given f based on $\mathcal{H}(f)$.

4) $h_I(e, f)$: the length posterior feature of the specific target hypothesis with length *I* based on the mixture search space $\mathcal{H}(f)$ generated by the HM decoder:

$$h_I(e, f) = \sum_{e' \in \mathcal{H}(f), |e'| = |e| = I} P(e'|f)$$

Note here that features in $h_{\mathcal{H}(f)}(e, f)$ and $h_I(e, f)$ will be computed when the computations of all the remainder features in two categories have already finished for each e in $\mathcal{H}(f)$, and they will be used to update current HM decoding model scores.

Consensus features based on component search spaces have already shown effectiveness (Kumar et al., 2009; DeNero et al., 2010; Duan et al., 2010). We leverage consensus features based on the mixture search space newly generated in HM decoding as well. The length posterior feature (Zen and Ney, 2006) is used to adjust the preference of HM decoder for longer or shorter translations, and the stemmed *n*-gram posterior features are used to provide more discriminative power for HM decoding and to decrease the effects of morphological changes in words for more accurate computation of consensus statistics.

The second feature category contains a set of general features. Although there are more features that can be incorporated into HM decoding besides the ones we list below, we only utilize the most representative ones for convenience:

1) $h_{Length}(e, f)$: the word count feature.

2) $h_{LM}(e, f)$: the language model feature.

3) $h_{Dict}(e, f)$: the dictionary-based feature that counts how many lexicon pairs can be found in a given translation pair (e, f).

4) $h_{[\cdot]}(e, f)$ and $h_{<\cdot>}(e, f)$: reordering features that penalize the uses of straight and inverted BTG rules during the derivation of e in HM decoding. These two features are specific to BTG-based HM decoding (Section 2.4.1):

$$h_{[\cdot]}(e, f) = \sum_{r \in \mathcal{D}(e)} \delta_r([\cdot])$$

$$h_{<\cdot>}(e, f) = \sum_{r \in \mathcal{D}(e)} \delta_r(<\cdot>)$$

5) $h_{Rule}(e, f)$ and $h_{Glue}(e, f)$: reordering features that penalize the uses of hierarchical and glue rules during the derivation of e in HM decoding. These two features are specific to SCFG-based HM decoding (Section 2.4.2):

$$h_{Rule}(e, f) = \sum_{r \in \mathcal{D}(e)} \delta_r(R)$$

$$h_{Glue}(e, f) = \sum_{r \in \mathcal{D}(e)} \delta_r([\cdot])$$

R is the hierarchical rule set provided by the HM decoder itself, $\delta_r(R)$ equals to 1 when r is provided by R, and 0 otherwise.

6) $h_{New}(e, f)$: the feature that counts how many *n*-grams in e are newly generated by the HM decoder, which cannot be found in all existing component search spaces:

$$h_{New}(e, f) = \sum_{\omega \in e} \#_\omega(e) \bar{\delta}_\omega(\bigcup_{n=1}^{N} \mathcal{H}_n(f))$$

$\bar{\delta}_\omega(\cup_{n=1}^{N} \mathcal{H}_n(f))$ equals to 1 when ω does not exist in $\cup_{n=1}^{N} \mathcal{H}_n(f)$, and 0 otherwise.

The MERT algorithm (Och, 2003) is used to tune weights of HM decoding features.

2.4 Decoding Algorithms

Two CKY-style algorithms for HM decoding are presented in this subsection. The first one is based on BTG (Wu, 1997), and the second one is based on SCFG, similar to Chiang (2005).

2.4.1 BTG-based HM Decoding

The first algorithm, *BTG-HMD*, is presented in Algorithm 1, where hypotheses of two consecutive source spans are composed using two BTG rules:

- *Straight rule* $[\cdot]$. It combines translations of two consecutive blocks into a single larger block in a straight order.
- *Inverted rule* $\langle\cdot\rangle$. It combines translations of two consecutive blocks into a single larger block in an inverted order.

These two rules are used bottom-up until the whole source sentence is fully covered. We use two reordering rule penalty features, $h_{[\cdot]}(e,f)$ and $h_{\langle\cdot\rangle}(e,f)$, to penalize the uses of these two rules.

Algorithm 1: BTG-based HM Decoding

1: **for** each component model $\mathcal{M}_n$ **do**
2: output the search space $\mathcal{H}_n(f)$ for the input f
3: **end for**
4: **for** $l = 1$ to $|f| - 1$ **do**
5: **for** all i, j s.t. $j - i = l$ **do**
6: $\mathcal{H}(f_i^j) = \{\emptyset\}$
7: **for** all k s.t. $i \le k < j$ **do**
8: **for** $e_1 \in \mathcal{H}(f_i^k)$ and $e_2 \in \mathcal{H}(f_{k+1}^j)$ **do**
9: add $e = Comb_{[\cdot]}(e_1, e_2)$ to $\mathcal{H}(f_i^j)$
10: add $e = Comb_{\langle\cdot\rangle}(e_1, e_2)$ to $\mathcal{H}(f_i^j)$
11: **end for**
12: **end for**
13: **for** each hypothesis $e \in \bigcup_{n=1}^{N} \mathcal{H}_n(f_i^j)$ **do**
14: compute HM decoding features for e
15: add e to $\mathcal{H}(f_i^j)$
16: **end for**
17: **for** each hypothesis $e \in \mathcal{H}(f_i^j)$ **do**
18: compute the n-gram and length posterior features for e based on $\mathcal{H}(f_i^j)$
19: update current HM decoding score of e
20: **end for**
21: **end for**
22: **end for**
23: return $\hat{e} \in \mathcal{H}(f)$ with the maximum model score

In BTG-HMD, in order to derive translations for a source span f_i^j, we compose hypotheses of any two smaller spans f_i^k and f_{k+1}^j using two BTG rules in line 9 and 10, $Comb_r(e_1, e_2)$ denotes the operations that firstly combine e_1 and e_2 using one BTG rule r and secondly compute HM decoding features for the newly generated hypothesis e. We compute HM decoding features for hypotheses contained in all existing component search spaces $\bigcup_{n=1}^{N} \mathcal{H}_n(f_i^j)$ as well, and add them to $\mathcal{H}(f_i^j)$. From line 17 to 20, we update current HM decoding scores for all hypotheses in $\mathcal{H}(f_i^j)$ using the n-gram and length posterior features computed based on $\mathcal{H}(f_i^j)$. When the whole source sentence is fully covered, we return the hypothesis with the maximum model score as the final best translation.

2.4.2 SCFG-based HM Decoding

The second algorithm, *SCFG-HMD*, is presented in Algorithm 2. An additional rule set $\mathcal{R}$, which is provided by the HM decoder, is used to compose hypotheses. It includes hierarchical rules extracted using Chiang (2005)'s method and glue rules. Two reordering rule penalty features, $h_{Rule}(e,f)$ and $h_{Glue}(e,f)$, are used to adjust the preferences of using hierarchical rules and glue rules.

Algorithm 2: SCFG-based HM Decoding

1: **for** each component model $\mathcal{M}_n$ **do**
2: output the search space $\mathcal{H}_n(f)$ for the input f
3: **end for**
4: **for** $l = 1$ to $|f| - 1$ **do**
5: **for** all i, j s.t. $j - i = l$ **do**
6: $\mathcal{H}(f_i^j) = \{\emptyset\}$
7: **for** each rule $r \in \mathcal{R}$ that matches f_i^j **do**
8: **for** $e_1 \in \mathcal{H}(r_{\#_1})$ and $e_2 \in \mathcal{H}(r_{\#_2})$ **do**
9: add $e = Comb_r(e_1, e_2)$ to $\mathcal{H}(f_i^j)$
10: **end for**
11: **end for**
12: **for** each hypothesis $e \in \bigcup_{n=1}^{N} \mathcal{H}_n(f_i^j)$ **do**
13: compute HM decoding features for e
14: add e to $\mathcal{H}(f_i^j)$
15: **end for**
16: **for** each hypothesis $e \in \mathcal{H}(f_i^j)$ **do**
17: compute the n-gram and length posterior features for e based on $\mathcal{H}(f_i^j)$
18: update current HM decoding score of e
19: **end for**
20: **end for**
21: **end for**
22: return $\hat{e} \in \mathcal{H}(f)$ with the maximum model score

Compared to BTG-HMD, the key differences in SCFG-HMD are located from line 7 to 11, where the translation for a given span f_i^j is generated by replacing the non-terminals in a hierarchical rule $r \in \mathcal{R}$ with their corresponding target translations, $r_{\#_i}$ is the source span that is covered by the i^{th} non-terminal of r, $\mathcal{H}(r_{\#_i})$ is the search space for $r_{\#_i}$ predicted by the HM decoder.

3 Comparisons to Related Techniques

3.1 Model Combination and Mixture Model based MBR Decoding

Model combination (DeNero et al., 2010) is an approach that selects translations from a conjoint search space using information from multiple SMT component models; Duan et al. (2010) presents a similar method, which utilizes a mixture model to combine distributions of hypotheses from different systems for Bayes-risk computation, and selects final translations from the combined search spaces using MBR decoding. Both of these two methods share a common limitation: they only re-rank the combined search space, without the capability to generate new translations. In contrast, by reusing hypotheses generated by all component systems in HM decoding, translations beyond any existing search space can be generated.

3.2 Co-Decoding and Joint Decoding

Li et al. (2009a) proposes collaborative decoding, an approach that combines translation systems by re-ranking partial and full translations iteratively using *n*-gram features from the predictions of other member systems. However, in co-decoding, all member systems must work in a synchronous way, and hypotheses between different systems cannot be shared during decoding procedure; Liu et al. (2009) proposes joint-decoding, in which multiple SMT models are combined in either translation or derivation levels. However, their method relies on the correspondence between nodes in hypergraph outputs of different models. HM decoding, on the other hand, can use hypotheses from component search spaces directly without any restriction.

3.3 Hybrid Decoding

Hybrid decoding (Cui et al., 2010) resembles our approach in the motivation. This method uses the system combination technique in decoding directly to combine partial hypotheses from different SMT models. However, confusion network construction brings high computational complexity. What's more, partial hypotheses generated by confusion network decoding cannot be assigned exact feature values for future use in higher level decoding, and they only use feature values of 1-best hypothesis as an approximation. HM decoding, on the other hand, leverages a set of enriched features, which are computable for all the hypotheses generated by either component systems or the HM decoder.

4 Experiments

4.1 Data and Metric

Experiments are conducted on the NIST Chinese-to-English MT tasks. The NIST 2004 (MT04) data set is used as the development set, and evaluation results are reported on the NIST 2005 (MT05), the newswire portions of the NIST 2006 (MT06) and 2008 (MT08) data sets. All bilingual corpora available for the NIST 2008 constrained data track of Chinese-to-English MT task are used as training data, which contain 5.1M sentence pairs, 128M Chinese words and 147M English words after pre-processing. Word alignments are performed using GIZA++ with the intersect-diag-grow refinement. The English side of bilingual corpus plus Xinhua portion of the LDC English Gigaword Version 3.0 are used to train a 5-gram language model.

Translation performance is measured in terms of case-insensitive BLEU scores (Papineni et al., 2002), which compute the brevity penalty using the shortest reference translation for each segment. Statistical significance is computed using the bootstrap re-sampling approach proposed by Koehn (2004). Table 1 gives some data statistics.

Data Set	#Sentence	#Word
MT04(dev)	1,788	48,215
MT05	1,082	29,263
MT06	616	17,316
MT08	691	17,424

Table 1: Statistics on dev and test data sets

4.2 Component Systems

For convenience of comparing HM decoding with several related decoding techniques, we include two state-of-the-art SMT systems as component systems only:

- *PB*. A phrase-based system (Xiong et al., 2006) with one lexicalized reordering model based on the maximum entropy principle.
- *DHPB*. A string-to-dependency tree-based system (Shen et al., 2008), which translates source strings to target dependency trees. A target dependency language model is used as an additional feature.

Phrasal rules are extracted on all bilingual data, hierarchical rules used in DHPB and reordering rules used in SCFG-HMD are extracted from a selected data set[3]. Reordering model used in PB is trained on the same selected data set as well. A trigram dependency language model used in DHPB is trained with the outputs from Berkeley parser on all language model training data.

4.3 Contrastive Techniques

We compare HM decoding with three multiple-system based decoding techniques:

- *Word-Level System Combination* (SC). We re-implement an IHMM alignment based system combination method proposed by Li et al. (2009b). The setting of the *N*-best candidates used is the same as the original paper.
- *Co-decoding* (CD). We re-implement it based on Li et al. (2009a), with the only difference that only two models are included in our re-implementation, instead of three in theirs. For each test set, co-decoding outputs three results, two for two member systems, and one for the further system combination.
- *Model Combination* (MC). Different from co-decoding, MC produces single one output for each input sentence. We re-implement this method based on DeNero et al. (2010) with two component models included.

4.4 Comparison to Component Systems

We compared HM decoding with two component SMT systems first (in Table 2). 30 features are used to annotate each hypothesis in HM decoding, including: 8 *n*-gram posterior features computed from PB/DHPB forests for $1 \leq n \leq 4$; 8 stemmed *n*-gram posterior features computed from stemmed PB/DHPB forests for $1 \leq n \leq 4$; 4 *n*-gram posterior features and 1 length posterior feature computed from the mixture search space of HM decoder for $1 \leq n \leq 4$; 1 LM feature; 1 word count feature; 1 dictionary-based feature; 2 grammar-specified rule penalty features for either BTG-HMD or SCFG-HMD; 4 count features for newly generated *n*-grams in HM decoding for $1 \leq n \leq 4$. All *n*-gram posteriors are computed using the efficient algorithm proposed by Kumar et al. (2009).

[3] LDC2003E07, LDC2003E14, LDC2005T06, LDC2005T10, LDC2005E83, LDC2006E26, LDC2006E34, LDC2006E85 and LDC2006E92

Model	BLEU%			
	MT04	MT05	MT06	MT08
PB	38.93	38.21	33.59	29.62
DHPB	39.90	39.76	35.00	30.43
BTG-HMD	**41.24***	**41.26***	**36.76***	**31.69***
SCFG-HMD	**41.31***	**41.19***	**36.63***	**31.52***

Table 2: HM decoding vs. single component system decoding (*: significantly better than each component system with $\boldsymbol{p} < 0.01$)

From table 2 we can see, both BTG-HMD and SCFG-HMD outperform decoding results of the best component system (DHPB) with significant improvements: +1.50, +1.76, and +1.26 BLEU points on MT05, MT06, and MT08 for BTG-HMD; +1.43, +1.63 and +1.09 BLEU points on MT05, MT06, and MT08 for SCFG-HMD. We also notice that BTG-HMD performs slight better than SCFG-HMD on test sets. We think the potential reason is that more reordering rules are used in SCFG-HMD to handle phrase movements than BTG-HMD do; however, current HM decoding model lacks the ability to distinguish the qualities of different rules.

We also investigate on the effects of different HM-decoding features. For the convenience of comparison, we divide them into five categories:

- *Set-1*. 8 *n*-gram posterior features based on 2 component search spaces plus 3 commonly used features (1 LM feature, 1 word count feature and 1 dictionary-based feature).
- *Set-2*. 8 stemmed *n*-gram posterior features based on 2 stemmed component search spaces.
- *Set-3*. 4 *n*-gram posterior features and 1 length posterior feature based on the mixture search space of the HM decoder.
- *Set-4*. 2 grammar-specified reordering rule penalty features.
- *Set-5*. 4 count features for unseen *n*-grams generated by HM decoder itself.

Except for the dictionary-based feature, all the features contained in Set-1 are used by the latest multiple-system based consensus decoding techniques (DeNero et al., 2010; Duan et al., 2010). We use them as the starting point. Each time, we add one more feature set and describe the changes of performances by drawing two curves for each HM decoding algorithm on MT08 in Figure 3.

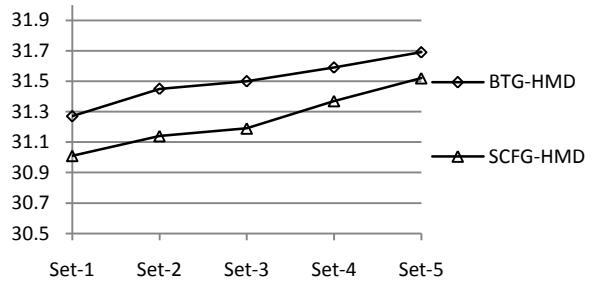


Figure 3: Effects of using different sets of HM decoding features on MT08

With Set-1 used only, HM-decoding has already outperformed the best component system, which shows the strong contributions of these features as proved in related work; small gains (+0.2 BLEU points) are achieved by using 8 stemmed *n*-gram posterior features in Set-2, which shows consensus statistics based on *n*-grams in their stem forms are also helpful; *n*-gram and length posterior features based on mixture search space bring improvements as well; reordering rule penalty features and count features for unseen *n*-grams boost newly generated hypotheses specific for HM decoding, and they contribute to the overall improvements.

4.5 Comparison to System Combination

Word-level system combination is state-of-the-art method to improve translation performance using outputs generated by multiple SMT systems. In this paper, we compare our HM decoding with the combination method proposed by Li et al. (2009b). Evaluation results are shown in Table 3.

Model	BLEU%			
	MT04	MT05	MT06	MT08
SC	41.14	40.70	36.04	31.16
BTG-HMD	**41.24**	**41.26**$^{+}$	**36.76**$^{+}$	**31.69**$^{+}$
SCFG-HMD	**41.31**$^{+}$	**41.19**$^{+}$	**36.63**$^{+}$	**31.52**$^{+}$

Table 3: HM decoding vs. system combination (+: significantly better than SC with $\boldsymbol{p} < 0.05$)

Compared to word-level system combination, both BTG-HMD and SCFG-HMD can provide significant improvements. We think the potential reason for these improvements is that, system combination can only use a small portion of the component systems' search spaces; HM decoding, on the other hand, can make full use of the entire translation spaces of all component systems.

4.6 Comparison to Consensus Decoding

Consensus decoding is another decoding technique that motivates our approach. We compare our HM decoding with two latest multiple-system based consensus decoding approaches, co-decoding and model combination. We list the comparison results in Table 4, in which CD-PB and CD-DHPB denote the translation results of two member systems in co-decoding respectively, CD-Comb denotes the results of further combination using outputs of CD-PB and CD-DHPB, MC denotes the results of model combination.

Model	BLEU%			
	MT04	MT05	MT06	MT08
CD-PB	40.39	40.34	35.20	30.39
CD-DHPB	40.81	40.56	35.73	30.87
CD-Comb	41.27	41.02	36.37	31.54
MC	41.19	40.96	36.30	31.43
BTG-HMD	**41.24**	**41.26**$^{+}$	**36.76**$^{+}$	**31.69**
SCFG-HMD	**41.31**	**41.19**	**36.63**$^{+}$	**31.52**

Table 4: HM decoding vs. consensus decoding (+: significantly better than the best result of consensus decoding methods with $\boldsymbol{p} < 0.05$)

Table 4 shows that after an additional system combination procedure, CD-Comb performs slight better than MC. Both BTG-HMD and SCFG-HMD perform consistent better than CD and MC on all blind test sets, due to its richer generative capability and usage of larger search spaces.

4.7 System Combination over BTG-HMD and SCFG-HMD Outputs

As BTG-HMD and SCFG-HMD are based on two different decoding grammars, we could perform system combination over the outputs of these two settings ($SC^{BTG+SCFG}$) for further improvements as well, just as Li et al. (2009a) did in co-decoding. We present evaluation results in Table 5.

Model	BLEU%			
	MT04	MT05	MT06	MT08
BTG-HMD	41.24	41.26	36.76	31.69
SCFG-HMD	41.31	41.19	36.63	31.52
$SC^{BTG+SCFG}$	**41.74**$^{+}$	**41.53**$^{+}$	**37.11**$^{+}$	**32.06**$^{+}$

Table 5: System combination based on the outputs of BTG-HMD and SCFG-HMD (+: significantly better than the best HM decoding algorithm (SCFG-HMD) with $\boldsymbol{p} < 0.05$)

After system combination, translation results are significantly better than all decoding approaches investigated in this paper: up to 2.11 BLEU points over the best component system (DHPB), up to 1.07 BLEU points over system combination, up to 0.74 BLEU points over co-decoding, and up to 0.81 BLEU points over model combination.

4.8 Evaluation of Oracle Translations

In the last part, we evaluate the quality of oracle translations on the *n*-best lists generated by HM decoding and all decoding approaches discussed in this paper. Oracle performances are obtained using the metric of sentence-level BLEU score proposed by Ye et al. (2007), and each decoding approach outputs its 1000-best hypotheses, which are used to extract oracle translations.

Model	BLEU%			
	MT04	MT05	MT06	MT08
PB	49.53	48.36	43.69	39.39
DHPB	50.66	49.59	44.68	40.47
SC	51.77	50.84	46.87	42.11
CD-PB	50.26	50.10	45.65	40.52
CD-DHPB	51.91	50.61	46.23	41.01
CD-Comb	52.10	51.00	46.95	42.20
MC	52.03	51.22	46.60	42.23
BTG-HMD	**52.69**$^{+}$	**51.75**$^{+}$	**47.08**	**42.71**$^{+}$
SCFG-HMD	**52.94**$^{+}$	**51.40**	**47.27**$^{+}$	**42.45**$^{+}$
$SC^{BTG+SCFG}$	**53.58**$^{+}$	**52.03**$^{+}$	**47.90**$^{+}$	**43.07**$^{+}$

Table 6: Oracle performances of different methods (+: significantly better than the best multiple-system based decoding method (CD-Comb) with $\boldsymbol{p} < 0.05$)

Results are shown in Table 6: compared to each single component system, decoding methods based on multiple SMT systems can provide significant improvements on oracle translations; word-level system combination, collaborative decoding and model combination show similar performances, in which CD-Comb performs best; BTG-HMD, SCFG-HMD and $SC^{BTG+SCFG}$ can obtain significant improvements than all the other approaches, and $SC^{BTG+SCFG}$ performs best on all evaluation sets.

5 Conclusion

In this paper, we have presented the hypothesis mixture decoding approach to combine multiple SMT models, in which hypotheses generated by multiple component systems are used to compose new translations. HM decoding method integrates the advantages of both system combination and consensus decoding techniques into a unified framework. Experimental results across different NIST Chinese-to-English MT evaluation data sets have validated the effectiveness of our approach.

In the future, we will include more SMT models and explore more features, such as syntax-based features, helping to improve the performance of HM decoding. We also plan to investigate more complicated reordering models in HM decoding.

References

David Chiang. 2005. A Hierarchical Phrase-based Model for Statistical Machine Translation. In *Proceedings of the Association for Computational Linguistics*, pages 263-270.

David Chiang. 2010. Learning to Translate with Source and Target Syntax. In *Proceedings of the Association for Computational Linguistics*, pages 1443-1452.

Lei Cui, Dongdong Zhang, Mu Li, Ming Zhou, and Tiejun Zhao. 2010. Hybrid Decoding: Decoding with Partial Hypotheses Combination over Multiple SMT Systems. In *Proceedings of the International Conference on Computational Linguistics*, pages 214-222.

John DeNero, David Chiang, and Kevin Knight. 2009. Fast Consensus Decoding over Translation Forests. In *Proceedings of the Association for Computational Linguistics*, pages 567-575.

John DeNero, Shankar Kumar, Ciprian Chelba and Franz Och. 2010. Model Combination for Machine Translation. In *Proceedings of the North American Association for Computational Linguistics*, pages 975-983.

Nan Duan, Mu Li, Dongdong Zhang, and Ming Zhou. 2010. Mixture Model-based Minimum Bayes Risk Decoding using Multiple Machine Translation Systems. In *Proceedings of the International Conference on Computational Linguistics*, pages 313-321.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable Inference and Training of Context-Rich Syntactic Translation Models. In *Proceedings of the Association for Computational Linguistics*, pages 961-968.

Xiaodong He, Mei Yang, Jianfeng Gao, Patrick Nguyen, and Robert Moore. 2008. Indirect-HMM-based Hypothesis Alignment for Combining Outputs from Machine Translation Systems. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, pages 98-107.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, pages 388-395.

Shankar Kumar and William Byrne. 2004. Minimum Bayes-Risk Decoding for Statistical Machine Translation. In *Proceedings of the North American Association for Computational Linguistics*, pages 169-176.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient Minimum Error Rate Training and Minimum Bayes-Risk Decoding for Translation Hypergraphs and Lattices. In *Proceedings of the Association for Computational Linguistics*, pages 163-171.

Mu Li, Nan Duan, Dongdong Zhang, Chi-Ho Li, and Ming Zhou. 2009a. Collaborative Decoding: Partial Hypothesis Re-Ranking Using Translation Consensus between Decoders. In *Proceedings of the Association for Computational Linguistics*, pages 585-592.

Chi-Ho Li, Xiaodong He, Yupeng Liu, and Ning Xi. 2009b. Incremental HMM Alignment for MT system Combination. In *Proceedings of the Association for Computational Linguistics*, pages 949-957.

Yang Liu, Haitao Mi, Yang Feng, and Qun Liu. 2009. Joint Decoding with Multiple Translation Models. In *Proceedings of the Association for Computational Linguistics*, pages 576-584.

Franz Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the Association for Computational Linguistics*, pages 160-167.

Franz Och and Hermann Ney. 2004. The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics*, 30(4): 417-449.

Kishore Papineni, Salim Roukos, Todd Ward, and Weijing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Association for Computational Linguistics*, pages 311-318.

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

Antti-Veikko Rosti, Spyros Matsoukas, and Richard Schwartz. 2007. Improved Word-Level System Combination for Machine Translation. In *Proceedings of the Association for Computational Linguistics*, pages 312-319.

Roy Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice Minimum Bayes-Risk Decoding for Statistical Machine Translation. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, pages 620-629.

Dekai Wu. 1997. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3): 377-404.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum Entropy based Phrase Reordering Model for Statistical Machine Translation. In *Proceedings of the Association for Computational Linguistics*, pages 521-528.

Yang Ye, Ming Zhou, and Chin-Yew Lin. 2007. Sentence Level Machine Translation Evaluation as a Ranking Problem: one step aside from BLEU. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 240-247.

Minimum Bayes-risk System Combination

Jesús González-Rubio
Instituto Tecnológico de Informática
U. Politècnica de València
46022 Valencia, Spain
jegonzalez@iti.upv.es

Alfons Juan **Francisco Casacuberta**
D. de Sistemas Informáticos y Computación
U. Politècnica de València
46022 Valencia, Spain
{ajuan,fcn}@dsic.upv.es

Abstract

We present *minimum Bayes-risk system combination*, a method that integrates consensus decoding and system combination into a unified multi-system minimum Bayes-risk (MBR) technique. Unlike other MBR methods that re-rank translations of a single SMT system, MBR system combination uses the MBR decision rule and a linear combination of the component systems' probability distributions to search for the minimum risk translation among all the finite-length strings over the output vocabulary. We introduce expected BLEU, an approximation to the BLEU score that allows to efficiently apply MBR in these conditions. MBR system combination is a general method that is independent of specific SMT models, enabling us to combine systems with heterogeneous structure. Experiments show that our approach bring significant improvements to single-system-based MBR decoding and achieves comparable results to different state-of-the-art system combination methods.

1 Introduction

Once statistical models are trained, a decoding approach determines what translations are finally selected. Two parallel lines of research have shown consistent improvements over the max–derivation decoding objective, which selects the highest probability derivation. *Consensus decoding* procedures select translations for a single system with a minimum Bayes risk (MBR) (Kumar and Byrne, 2004). *System combination* procedures, on the other hand, generate translations from the output of multiple component systems by combining the best fragments of these outputs (Frederking and Nirenburg, 1994). In this paper, we present minimum Bayes risk system combination, a technique that unifies these two approaches by learning a consensus translation over multiple underlying component systems.

MBR system combination operates directly on the outputs of the component models. We perform an MBR decoding using a linear combination of the component models' probability distributions. Instead of re-ranking the translations provided by the component systems, we search for the hypothesis with the minimum expected translation error among all the possible finite-length strings in the target language. By using a loss function based on BLEU (Papineni et al., 2002), we avoid the hypothesis alignment problem that is central to standard system combination approaches (Rosti et al., 2007). MBR system combination assumes only that each translation model can produce expectations of n-gram counts; the latent derivation structures of the component systems can differ arbitrary. This flexibility allows us to combine a great variety of SMT systems.

The key contributions of this paper are three: the usage of a linear combination of distributions within the MBR decoding, which allows multiple SMT models to be involved in, and makes the computation of n-grams statistics to be more accurate; the decoding in an extended search space, which allows to find better hypotheses than the evidences provided by the component models; and the use of an expected BLEU score instead of the sentence-wise BLEU, which allows to efficiently apply MBR decoding in the huge search space under consideration.

We evaluate in a multi-source translation task obtaining improvements of up to $+2.0$ BLEU abs. over the best single system max-derivation, and state-of-the-art performance in the system combination task of the ACL 2010 workshop on SMT.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1268–1277,
Portland, Oregon, June 19-24, 2011.

2 Related Work

MBR system combination is a multi-system generalization of MBR decoding where the space of hypotheses is not constrained to the space of evidences. We expand the space of hypotheses following some underlying ideas of system combination techniques.

2.1 Minimum Bayes risk

In SMT, MBR decoding allows to minimize the loss of the output for a single translation system. MBR is generally implemented by re-ranking an N-best list of translations produced by a first pass decoder (Kumar and Byrne, 2004). Different techniques to widen the search space have been described (Tromble et al., 2008; DeNero et al., 2009; Kumar et al., 2009; Li et al., 2009). These works extend the traditional MBR algorithms based on N-best lists to work with lattices.

The use of MBR to combine the outputs of various MT systems has also been explored previously. Duan et al. (2010) present an MBR decoding that makes use of a mixture of different SMT systems to improve translation accuracy. Our technique differs in that we use a linear combination instead of a mixture, which avoids the problem of component systems not sharing the same search space; perform the decoding in a search space larger than the outputs of the component models; and optimize an expected BLEU score instead of the linear approximation to it described in (Tromble et al., 2008).

DeNero et al. (2010) present *model combination*, a multi-system lattice MBR decoding on the conjoined evidences spaces of the component systems. Our technique differs in that we perform the search in an extended search space not restricted to the provided evidences, have fewer parameters to learn, and optimizes an expected BLEU score instead of the linear BLEU approximation.

Another MBR-related technique to combine the outputs of various MT systems was presented by González-Rubio and Casacuberta (2010). They use different median string (Fu, 1982) algorithms to combine various machine translation systems. Our approach differs in that we take into account the posterior distribution over translations instead of considering each translation equally likely, optimize the expected BLEU score instead of a sentence-wise measure such as the edit distance or the sentence-level BLEU, and take into account the quality differences by associating a tunable scaling factor to each system.

2.2 System Combination

System combination techniques in MT take as input the outputs $\{\mathbf{e}_1, \cdots, \mathbf{e}_N\}$ of N translation systems, where $\mathbf{e}_n$ is a structured translation object (or N-best lists thereof), typically viewed as a sequence of words. The dominant approach in the field chooses a primary translation $\mathbf{e}_p$ as a backbone, then finds an alignment $\mathbf{a}_n$ to the backbone for each $\mathbf{e}_n$. A new search space is constructed from these backbone-aligned outputs and then a voting procedure of feature-based model predicts a final consensus translation (Rosti et al., 2007). MBR system combination entirely avoids this alignment problem by considering hypotheses as n-gram occurrence vectors rather than word sequences. MBR system combination performs the decoding in a larger search space and includes statistics from the components' posteriors, whereas system combination techniques typically do not.

Despite these advantages, system combination may be more appropriate in some settings. In particular, MBR system combination is designed primarily for statistical systems that generate N-best or lattice outputs. MBR system combination can integrate non-statistical systems that generate either a single or an unweighted output. However, we would not expect the same strong performance from MBR system combination in these constrained settings.

3 Minimum Bayes risk Decoding

MBR decoding aims to find the candidate hypothesis that has the least expected loss under a probability model (Bickel and Doksum, 1977). We begin with a review of MBR for SMT.

SMT can be described as a mapping of a word sequence $\mathbf{f}$ in a source language to a word sequence $\mathbf{e}$ in a target language; this mapping is produced by the MT decoder $\mathcal{D}(\mathbf{f})$. If the reference translation $\mathbf{e}$ is known, the decoder performance can be measured by the loss function $\mathcal{L}(\mathbf{e}, \mathcal{D}(\mathbf{f}))$. Given such a loss function $\mathcal{L}(\mathbf{e}, \mathbf{e}')$ between an automatic translation $\mathbf{e}'$ and a reference $\mathbf{e}$, and an underlying proba-

bility model $P(\mathbf{e}|\mathbf{f})$, MBR decoding has the following form (Goel and Byrne, 2000; Kumar and Byrne, 2004):

$$\hat{\mathbf{e}} = \arg\min_{\mathbf{e}' \in E} \mathcal{R}(\mathbf{e}') \quad (1)$$

$$= \arg\min_{\mathbf{e}' \in E} \sum_{\mathbf{e} \in E} P(\mathbf{e}|\mathbf{f}) \cdot \mathcal{L}(\mathbf{e}, \mathbf{e}') \ , \quad (2)$$

where $\mathcal{R}(\mathbf{e}')$ denotes the Bayes risk of candidate translation $\mathbf{e}'$ under loss function $\mathcal{L}$, and E represents the space of translations.

If the loss function between any two hypotheses can be bounded: $\mathcal{L}(\mathbf{e}, \mathbf{e}') \leq \mathcal{L}_{max}$, the MBR decoder can be rewritten in term of a similarity function $\mathcal{S}(\mathbf{e}, \mathbf{e}') = \mathcal{L}_{max} - \mathcal{L}(\mathbf{e}, \mathbf{e}')$. In this case, instead of minimizing the Bayes risk, we maximize the Bayes gain $\mathcal{G}(\mathbf{e}')$:

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}' \in E} \mathcal{G}(\mathbf{e}') \quad (3)$$

$$= \arg\max_{\mathbf{e}' \in E} \sum_{\mathbf{e} \in E} P(\mathbf{e}|\mathbf{f}) \cdot \mathcal{S}(\mathbf{e}, \mathbf{e}') \ . \quad (4)$$

MBR decoding can use different spaces for hypothesis selection and gain computation ($\arg\max$ and summatory in Eq. (4)). Therefore, the MBR decoder can be more generally written as follows:

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}' \in E_h} \sum_{\mathbf{e} \in E_e} P(\mathbf{e}|\mathbf{f}) \cdot \mathcal{S}(\mathbf{e}, \mathbf{e}') \ , \quad (5)$$

where E_h refers to the hypotheses space form where the translations are chosen and E_e refers to the evidences space that is used to compute the Bayes gain. We will investigate the expansion of the hypotheses space while keeping the evidences space as provided by the decoder.

4 MBR System Combination

MBR system combination is a multi-system generalization of MBR decoding. It uses the MBR decision rule on a linear combination of the probability distributions of the component systems. Unlike existing MBR decoding methods that re-rank translation outputs, MBR system combination search for the minimum risk hypotheses on the complete set of finite-length hypotheses over the output vocabulary. We assume the component systems to be statistically independent and define the Bayes gain as a linear combination of the Bayes gains of the components. Each system provides its own space of evidences $\mathcal{D}_n(\mathbf{f})$ and its posterior distribution over translations $P_n(\mathbf{e}|\mathbf{f})$. Given a sentence $\mathbf{f}$ in the source language, MBR system combination is written as follows:

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}' \in E_h} \mathcal{G}(\mathbf{e}') \quad (6)$$

$$\approx \arg\max_{\mathbf{e}' \in E_h} \sum_{n=1}^{N} \alpha_n \cdot \mathcal{G}_n(\mathbf{e}') \quad (7)$$

$$= \arg\max_{\mathbf{e}' \in E_h} \sum_{n=1}^{N} \alpha_n \cdot \sum_{\mathbf{e} \in \mathcal{D}_n(\mathbf{f})} P_n(\mathbf{e}|\mathbf{f}) \cdot \mathcal{S}(\mathbf{e}, \mathbf{e}') \ , \quad (8)$$

where N is the total number of component systems, E_h represents the hypotheses space where the search is performed, $\mathcal{G}_n(\mathbf{e}')$ is the Bayes gain of hypothesis $\mathbf{e}'$ given by the n^{th} component system and α_n is a scaling factor introduced to take into account the differences in quality of the component models. It is worth mentioning that by using a linear combination instead of a mixture model, we avoid the problem of component systems not sharing the same search space (Duan et al., 2010).

MBR system combination parameters training and decoding in the extended hypotheses space are described below.

4.1 Model Training

We learn the scaling factors in Eq. (8) using minimum error rate training (MERT) (Och, 2003). MERT maximizes the translation quality of $\hat{\mathbf{e}}$ on a held-out set, according to an evaluation metric that compares to a reference set. We used BLEU, choosing the scaling factors to maximize BLEU score of the set of translations predicted by MBR system combination. We perform the maximization by means of the down-hill simplex algorithm (Nelder and Mead, 1965).

4.2 Model Decoding

In most MBR algorithms, the hypotheses space is equal to the evidences space. Following the underlying idea of system combination, we are interested in extend the hypotheses space by including new sentences created using fragments of the hypotheses in the evidences spaces of the component models. We perform the search ($argmax$ operation in Eq. (8))

Algorithm 1 MBR system combination decoding.

Require: Initial hypothesis $\mathbf{e}$
Require: Vocabulary the evidences Σ

1: $\hat{\mathbf{e}} \leftarrow \mathbf{e}$
2: **repeat**
3: $\quad \mathbf{e}_{cur} \leftarrow \hat{\mathbf{e}}$
4: $\quad$ **for** $j = 1$ **to** $|\mathbf{e}_{cur}|$ **do**
5: $\quad\quad \hat{\mathbf{e}}_s \leftarrow \mathbf{e}_{cur}$
6: $\quad\quad$ **for** $a \in \Sigma$ **do**
7: $\quad\quad\quad \mathbf{e}'_s \leftarrow Substitute(\mathbf{e}_{cur}, a, j)$
8: $\quad\quad\quad$ **if** $\mathcal{G}(\mathbf{e}'_s) > \mathcal{G}(\hat{\mathbf{e}}_s)$ **then**
9: $\quad\quad\quad\quad \hat{\mathbf{e}}_s \leftarrow \mathbf{e}'_s$
10: $\quad\quad \hat{\mathbf{e}}_d \leftarrow Delete(\mathbf{e}_{cur}, j)$
11: $\quad\quad \hat{\mathbf{e}}_i \leftarrow \mathbf{e}_{cur}$
12: $\quad\quad$ **for** $a \in \Sigma$ **do**
13: $\quad\quad\quad \mathbf{e}'_i \leftarrow Insert(\mathbf{e}_{cur}, a, j)$
14: $\quad\quad\quad$ **if** $\mathcal{G}(\mathbf{e}'_i) > \mathcal{G}(\hat{\mathbf{e}}_i)$ **then**
15: $\quad\quad\quad\quad \hat{\mathbf{e}}_i \leftarrow \mathbf{e}'_i$
16: $\quad\quad \hat{\mathbf{e}} \leftarrow \arg\max_{\mathbf{e}' \in \{\mathbf{e}_{cur}, \hat{\mathbf{e}}_s, \hat{\mathbf{e}}_d, \hat{\mathbf{e}}_i\}} \mathcal{G}(\mathbf{e}')$
17: **until** $\mathcal{G}(\hat{\mathbf{e}}) \not> \mathcal{G}(\mathbf{e}_{cur})$
18: **return** $\mathbf{e}_{cur}$

Ensure: $\mathcal{G}(\mathbf{e}_{cur}) \geq \mathcal{G}(\mathbf{e})$

using the approximate median string (AMS) algorithm (Martínez et al., 2000). AMS algorithm perform a search on a hypotheses space equal to the free monoid Σ^* of the vocabulary of the evidences $\Sigma = Voc(E_e)$.

The AMS algorithm is shown in Algorithm 1. AMS starts with an initial hypothesis $\mathbf{e}$ that is modified using edit operations until there is no improvement in the Bayes gain (Lines 3–16). On each position j of the current solution $\mathbf{e}_{cur}$, we apply all the possible single edit operations: substitution of the j^{th} word of $\mathbf{e}_{cur}$ by each word a in the vocabulary (Lines 5–9), deletion of the j^{th} word of $\mathbf{e}_{cur}$ (Line 10) and insertion of each word a in the vocabulary in the j^{th} position of $\mathbf{e}_{cur}$ (Lines 11–15). If the Bayes gain of any of the new edited hypotheses is higher than the Bayes gain of the current hypothesis (Line 17), we repeat the loop with this new hypotheses $\hat{\mathbf{e}}$, in other case, we return the current hypothesis.

AMS algorithm takes as input an initial hypothesis $\mathbf{e}$ and the combined vocabulary of the evidences spaces Σ. Its output is a possibly new hypothesis whose Bayes gain is assured to be higher or equal than the Bayes gain of the initial hypothesis.

The complexity of the main loop (lines 2-17) is $O(|\mathbf{e}_{cur}| \cdot |\Sigma| \cdot C_{\mathcal{G}})$, where $C_{\mathcal{G}}$ is the cost of computing the gain of a hypothesis, and usually only a moderate number of iterations (< 10) is needed to converge (Martínez et al., 2000).

5 Computing BLEU-based Gain

We are interested in performing MBR system combination under BLEU. BLEU behaves as a score function: its value ranges between 0 and 1 and a larger value reflects a higher similarity. Therefore, we rewrite the gain function $\mathcal{G}(\cdot)$ using single evidence (or reference) BLEU (Papineni et al., 2002) as the similarity function:

$$\mathcal{G}_n(\mathbf{e}') = \sum_{\mathbf{e} \in \mathcal{D}_n(\mathbf{f})} P_n(\mathbf{e}|\mathbf{f}) \cdot \text{BLEU}(\mathbf{e}, \mathbf{e}') \quad (9)$$

$$\text{BLEU} = \prod_{k=1}^{4} \left(\frac{m_k}{c_k} \right)^{\frac{1}{4}} \cdot \min\left(e^{1-\frac{r}{c}}, 1.0 \right) , \quad (10)$$

where r is the length of the evidence, c the length of the hypothesis, m_k the number of n-gram matches of size k, and c_k the count of n-grams of size k in the hypothesis.

The evidences space $\mathcal{D}_n(\mathbf{f})$ may contain a huge number of hypotheses[1] which often make impractical to compute Eq. (9) directly. To avoid this problem, Tromble et al. (2008) propose *linear BLEU*, an approximation to the BLEU score to efficiently perform MBR decoding when the search space is represented with lattices. However, our hypotheses space is the full set of finite-length strings in the target vocabulary and can not be represented in a lattice.

In Eq. (9), we have one hypothesis $\mathbf{e}'$ that is to be compared to a set of evidences $\mathbf{e} \in \mathcal{D}_n(\mathbf{f})$ which follow a probability distribution $P_n(\mathbf{e}|\mathbf{f})$. Instead of computing the expected BLEU score by calculating the BLEU score with respect to each of the evidences, our approach will be to use the expected n-gram counts and sentence length of the evidences to compute a single-reference BLEU score. We replace the reference statistics (r and m_n in Eq. (10)) by the expected statistics (r' and m'_n) given the pos-

[1]For example, in a lattice the number of hypotheses may be exponential in the size of its state set.

terior distribution $P_n(\mathbf{e}|\mathbf{f})$ over the evidences:

$$\mathcal{G}_n(\mathbf{e}') = \prod_{k=1}^{4} \left(\frac{m'_k}{c_k} \right)^{\frac{1}{4}} \cdot \min\left(e^{1-\frac{r'}{c}}, 1.0 \right) \quad (11)$$

$$r' = \sum_{\mathbf{e} \in \mathcal{D}_n(\mathbf{f})} |\mathbf{e}| \cdot P_n(\mathbf{e}|\mathbf{f}) \quad (12)$$

$$m'_k = \sum_{ng \in \mathcal{N}_k(\mathbf{e}')} \min(C_{\mathbf{e}'}(ng), C'(ng)) \quad (13)$$

$$C'(ng) = \sum_{\mathbf{e} \in \mathcal{D}_n(\mathbf{f})} C_{\mathbf{e}}(ng) \cdot P_n(\mathbf{e}|\mathbf{f}) \,, \quad (14)$$

where $\mathcal{N}_k(\mathbf{e}')$ is the set of n-grams of size k in the hypothesis, $C_{\mathbf{e}'}(ng)$ is the count of the n-gram ng in the hypothesis and $C'(ng)$ is the expected count of ng in the evidences. To compute the n-gram matchings m'_k, the count of each n-gram is truncated, if necessary, to not exceed the expected count for that n-gram in the evidences.

We have replaced a summation over a possibly exponential number of items ($\mathbf{e}' \in \mathcal{D}_n(\mathbf{f})$ in Eq. (9)) with a summation over a polynomial number of n-grams that occur in the evidences[2]. Both, the expected length of the evidences r' and their expected n-gram counts m'_k can be pre-computed efficiently from N-best lists and translation lattices (Kumar et al., 2009; DeNero et al., 2010).

6 Experiments

We report results on a multi-source translation task. From the Europarl corpus released for the ACL 2006 workshop on MT (WMT2006), we select those sentence pairs from the German–English (de–en), Spanish–English (es–en) and French–English (fr–en) sub-corpora that share the same English translation. We obtain a multi-source corpus with German, Spanish and French as source languages and English as target language. All the experiments were carried out with the lowercased and tokenized version of this corpus.

We report results using BLEU (Papineni et al., 2002) and translation edit rate (Snover et al., 2006) (TER). We measure statistical significance using 95% confidence intervals computed using paired bootstrap re-sampling (Zhang and Vogel, 2004). In all table cells (except for Table 3) systems without statistically significant differences are marked with the same superscript.

System		dev		test	
		BLEU	TER	BLEU	TER
de→en	MAX	25.3	60.5	25.6*	60.3
	MBR	25.1	60.7	25.4*	60.5
es→en	MAX	30.9*	53.3*	30.4*	53.9*
	MBR	31.0*	53.4*	30.4*	54.0*
fr→en	MAX	30.7*	53.9*	30.8*	53.4*
	MBR	30.7*	53.8*	30.9*	53.4*

Table 1: Performance of base systems.

Approach	dev		test	
	BLEU	TER	BLEU	TER
Best MAX	30.9*	53.3*	30.8*	53.4*
Best MBR	31.0*	53.4*	30.9*	53.4*
MBR-SC	**32.3**	**52.5**	**32.8**	**52.3**

Table 2: Performance from best single system max-derivation decoding (*Best MAX)*, the best single system minimum Bayes risk decoding (*Best MBR*) and minimum Bayes risk system combination (*MBR-SC*) combining three systems.

6.1 Base Systems

We combine outputs from three systems, each one translating from one source language (German, Spanish or French) into English. Each individual system is a phrase-based system trained using the Moses toolkit (Koehn et al., 2007). The parameters of the systems were tuned using MERT (Och, 2003) to optimize BLEU on the development set. Each base system yields state-of-the-art performance, summarized in Table 1. For each system, we report the performance of max-derivation decoding (MAX) and 1000-best[3] MBR decoding (Kumar and Byrne, 2004).

6.2 Experimental Results

Table 2 compares MBR system combination (MBR-SC) to the best MAX and MBR systems. Both Best

[2] If $\mathcal{D}_n(\mathbf{f})$ is represented by a lattice, the number of n-grams is polynomial in the number of edges in the lattice.

[3] Ehling et al. (2007) studied up to 10000-best and show that the use of 1000-best candidates is sufficient for MBR decoding.

Setup	BLEU	TER
Best MBR	30.9	53.4
MBR-SC Expected	30.9	53.5
MBR-SC E/Conjoin	32.4	52.1
MBR-SC E/C/evidences-best	30.9	53.5
MBR-SC E/C/hypotheses-best	31.8	52.5
MBR-SC E/C/Extended	32.7	52.3
MBR-SC E/C/Ex/MERT	**32.8**	**52.3**

Table 3: Results on the test set for different setups of minimum Bayes risk system combination.

MBR and MBR-SC were computed on 1000-best lists. MBR-SC uses expected BLEU as gain function using the conjoined evidences spaces of the three systems to compute expected BLEU statistics. It performs the search in the free monoid of the output vocabulary, and its model parameters were tuned using MERT on the development set. This is the standard setup for MBR system combination, and we refer to it as MBR-SC-E/C/Ex/MERT in Table 3.

MBR system combination improves single Best MAX system by $+2.0$ BLEU points in test, and always improves over MBR. This improvement could arise due to multiple reasons: the expected BLEU gain, the larger evidences space, the extended hypotheses space, or the MERT tuned scaling factor values. Table 3 teases apart these contributions.

We first apply MBR-SC to the best system (MBR-SC-Expected). Best MBR and MBR-SC-Expected differ only in the gain function: MBR uses sentence level BLEU while MBR-SC-Expected uses the expected BLEU gain described in Section 5. MBR-SC-Expected performance is comparable to MBR decoding on the 1000-best list from the single best system. The expected BLEU approximation performs as well as sentence-level BLEU and additionally requires less total computation.

We now extend the evidences space to the conjoined 1000-best lists (MBR-SC-E/Conjoin). MBR-SC-E/Conjoin is much better than the best MBR on a single system. This implies that either the expected BLEU statistics computed in the conjoined evidences space are stronger or the larger conjoined evidences spaces introduce better hypotheses.

When we restrict the BLEU statistics to be computed from only the best system's evidences space (MBR-SC-E/C/evidences-best), BLEU scores dramatically decrease relative to MBR-SC-E/Conjoin. This implies that the expected BLEU statistics computed over the conjoined 1000-best lists are stronger than the corresponding statistics from the single best system. On the other hand, if we restrict the search space to only the 1000-best list of the best system (MBR-SC-E/C/hypotheses-best), BLEU scores also decrease relative to MBR-SC-E/Conjoin. This implies that the conjoined search space also contains better hypotheses than the single best system's search space.

These results validate our approach. The linear combination of the probability distributions in the conjoined evidences spaces allows to compute much stronger statistics for the expected BLEU gain and also contains some better hypotheses than the single best system's search space does.

We next expand the conjoined evidences spaces using the decoding algorithm described in Section 4.2 (MBR-SC-E/C/Extended). In this case, the expected BLEU statistics are computed from the conjoined 1000-best lists of the three systems, but the hypotheses space where we perform the decoding is expanded to the set of all possible finite-length hypotheses over the vocabulary of the evidences. We take the output of MBR-SC-E/Conjoin as the initial hypotheses of the decoding (see Algorithm 1). MBR-SC-E/C/Extended improves BLEU score of MBR-SC-E/Conjoin but obtains a slightly worse TER score. Since these two systems are identical in their expected BLEU statistics, the improvements in BLEU imply that the extended search space has introduced better hypotheses. The degradation in TER performance can be explained by the use of a BLEU-based gain function in the decoding process.

We finally compute the optimum values for the scaling factors of the different system using MERT (MBR-SC-E/C/Ex/MERT). MBR-SC-E/C/Ex/MERT slightly improves BLEU score of MBR-SC-E/C/Extended. This implies that the optimal values of the scaling factors do not deviate much from 1.0; a similar result was reported in (Och and Ney, 2001). We hypothesize that this is because the three component systems share the same SMT model, pre-process and decoding. We expect to obtain larger improvements when combining systems implementing different MT paradigms.

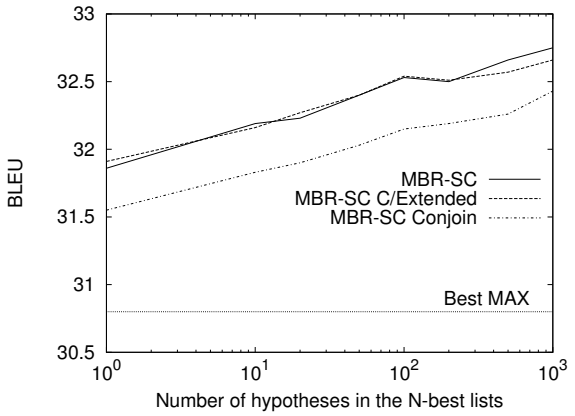


Figure 1: Performance of minimum Bayes risk system combination (MBR-SC) for different sizes of the evidences space in comparison to other MBR-SC setups.

MBR-SC-E/C/Ex/MERT is the standard setup for MBR system combination and, from now, on we will refer to it as MBR-SC.

We next evaluate performance of MBR system combination on N-best lists of increasing sizes, and compare it to MBR-SC-E/C/Extended and MBR-SC-E/Conjoin in the same N-best lists. We list the results of the Best MAX system for comparison.

Results in Figure 1 confirm the conclusions extracted from results displayed in Table 3. MBR-SC-Conjoin is consistently better than the Best MAX system, and differences in BLEU increase with the size of the evidences space. This implies that the linear combination of posterior probabilities allow to compute stronger statistics for the expected BLEU gain, and, in addition, the larger the evidences space is, the stronger the computed statistics are. MBR-SC-C/Extended is also consistently better than MBR-SC-Conjoin with an almost constant improvement of $+0.4$ BLEU points. This result show that the extended search space always contains better hypotheses than the conjoined evidences spaces; also confirms the soundness of Algorithm 1 that allows to reach them. Finally, MBR-SC also slightly improves MBR-SC-C/Extended. The optimization of the scaling factors allows only small improvements in BLEU.

Figure 2 display the MBR system combination translation and compare it to the max-derivation translations of the three component systems. Reference translation is also listed for comparison. MBR-SC adds word "*point*" to create a new translation equal to the reference. MBR-SC is able to detect that this is valuable word even though it does not appear in the max-derivation hypotheses.

MAX de→en	i will return later .
MAX es→en	i shall come back to that later .
MAX fr→en	i will return to this later .
MBR-SC	i will return to this point later .
Reference	i will return to this point later .

Figure 2: MBR system combination example.

6.3 Comparison to System Combination

Figure 3 compares MBR system combination (MBR-SC) with state-of-the-art system combination techniques presented to the system combination task of the ACL 2010 workshop on MT (WMT2010). All system combination techniques build a "word sausage" from the outputs of the different component systems and choose a path trough the sausage with the highest score under different models. A description of these systems can be found in (Callison-Burch et al., 2010).

In this task, the output of the component systems are single hypotheses or unweighted lists thereof. Therefore, we lack of the statistics of the components' posteriors which is one of the main advantages of MBR system combination over system combination techniques. However, we find that, even in these constrained setting, MBR system combination performance is similar to the best system combination techniques for all translation directions. These experiments validate our approach. MBR system combination yields state-of-the-art performance while avoiding the challenge of aligning translation hypotheses.

7 Conclusion

MBR system combination integrates consensus decoding and system combination into a unified multi-system MBR technique. MBR system combination uses the MBR decision rule on a linear combination of the component systems' probability distributions to search for the sentence with the minimum Bayes risk on the complete set of finite-length

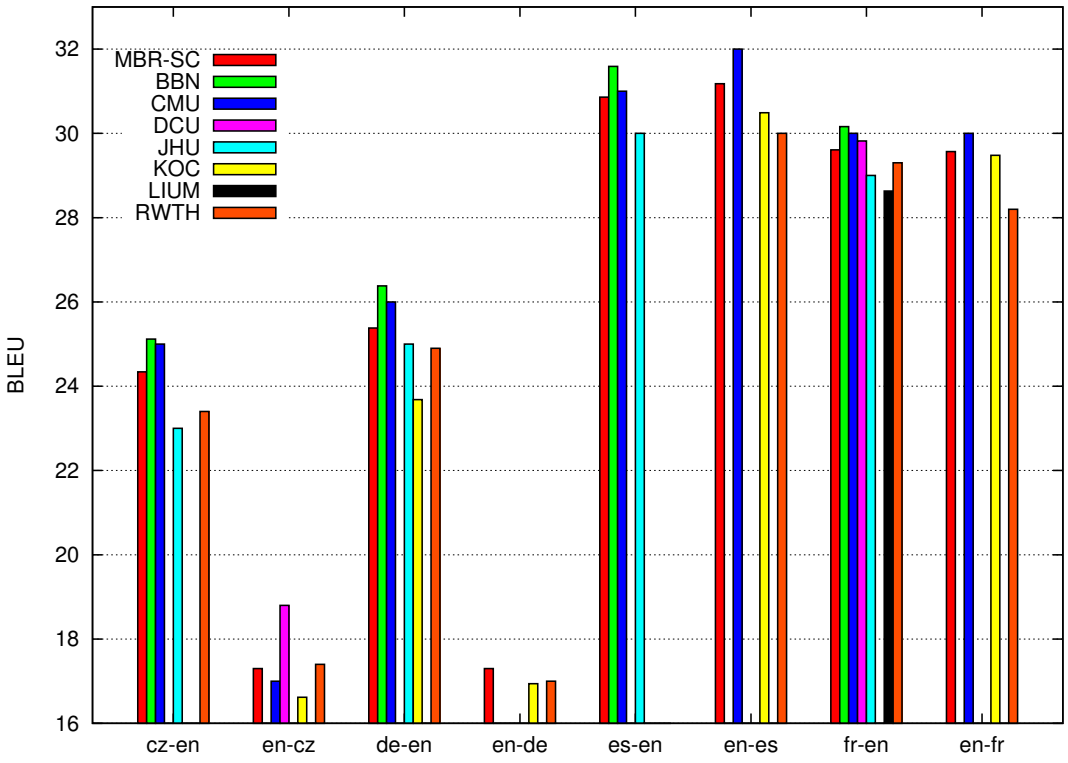


Figure 3: Performance of minimum Bayes risk system combination (MBR-SC) for different language directions in comparison to the rest of system combination techniques presented in the WMT2010 system combination task.

strings in the output vocabulary. Component systems can have varied decoding strategies; we only require that each system produce an N-best list (or a lattice) of translations. This flexibility allows the technique to be applied quite broadly. For instance, Leusch et al. (2010) generate intermediate translations in several pivot languages, translate them separately into the target language, and generate a consensus translation out of these using a system combination technique. Likewise, these pivot translations could be combined via MBR system combination.

MBR system combination has two significant advantages over current approaches to system combination. First, it does not rely on hypothesis alignment between outputs of individual systems. Aligning translation hypotheses can be challenging and has a substantial effect on combination performance (He et al., 2008). Instead of aligning the sentences, we view the sentences as vectors of n-gram counts and compute the expected statistics of the BLEU score to compute the Bayes gain. Second, we do not need to pick a backbone system for combination. Choosing a backbone system can also be challenging and also affects system combination performance (He and Toutanova, 2009). MBR system combination sidesteps this issue by working directly on the conjoined evidences space produced by the outputs of the component systems, and allows the consensus model to express system preferences via scaling factors.

Despite its simplicity, MBR system combination provides strong performance by leveraging different consensus, decoding and training techniques. It outperforms best MAX or MBR derivation on each of the component systems. In addition, it obtains state-of-the-art performance in a constrained setting better suited for dominant system combination techniques.

Acknowledgements

Work supported by the EC (FEDER/FSE) and the Spanish MEC/MICINN under the MIPRCV "Consolider Ingenio 2010" program (CSD2007-00018), the iTrans2 (TIN2009-14511) project, the UPV

under grant 20091027 and the FPU scholarship AP2006-00691. Also supported by the Spanish MITyC under the erudito.com (TSI-020110-2009-439) project and by the Generalitat Valenciana under grant Prometeo/2009/014.

References

Peter J. Bickel and Kjell A Doksum. 1977. *Mathematical statistics : basic ideas and selected topics*. Holden-Day, San Francisco.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar F. Zaidan. 2010. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53, Morristown, NJ, USA. Association for Computational Linguistics.

John DeNero, David Chiang, and Kevin Knight. 2009. Fast consensus decoding over translation forests. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, pages 567–575, Morristown, NJ, USA. Association for Computational Linguistics.

John DeNero, Shankar Kumar, Ciprian Chelba, and Franz Och. 2010. Model combination for machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 975–983, Morristown, NJ, USA. Association for Computational Linguistics.

Nan Duan, Mu Li, Dongdong Zhang, and Ming Zhou. 2010. Mixture model-based minimum bayes risk decoding using multiple machine translation systems. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 313–321, Beijing, China, August. Coling 2010 Organizing Committee.

Nicola Ehling, Richard Zens, and Hermann Ney. 2007. Minimum bayes risk decoding for bleu. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 101–104, Morristown, NJ, USA. Association for Computational Linguistics.

Robert Frederking and Sergei Nirenburg. 1994. Three heads are better than one. In *Proceedings of the fourth conference on Applied natural language processing*, pages 95–100, Morristown, NJ, USA. Association for Computational Linguistics.

K.S. Fu. 1982. *Syntactic Pattern Recognition and Applications*. Prentice Hall.

Vaibhava Goel and William J. Byrne. 2000. Minimum bayes-risk automatic speech recognition. *Computer Speech & Language*, 14(2):115–135.

Jesús González-Rubio and Francisco Casacuberta. 2010. On the use of median string for multi-source translation. In *In Proceedings of the International Conference on Pattern Recognition (ICPR2010)*, pages 4328–4331.

Xiaodong He and Kristina Toutanova. 2009. Joint optimization for machine translation system combination. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, pages 1202–1211, Morristown, NJ, USA. Association for Computational Linguistics.

Xiaodong He, Mei Yang, Jianfeng Gao, Patrick Nguyen, and Robert Moore. 2008. Indirect-hmm-based hypothesis alignment for combining outputs from machine translation systems. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 98–107, Morristown, NJ, USA. Association for Computational Linguistics.

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

Shankar Kumar and William J. Byrne. 2004. Minimum bayes-risk decoding for statistical machine translation. In *HLT-NAACL*, pages 169–176.

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

Gregor Leusch, Aurélien Max, Josep Maria Crego, and Hermann Ney. 2010. Multi-pivot translation by system combination. In *International Workshop on Spoken Language Translation*, Paris, France, December.

Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. 2009. Variational decoding for statistical machine translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Process-*

ing of the AFNLP: Volume 2 - Volume 2, pages 593–601, Morristown, NJ, USA. Association for Computational Linguistics.

C. D. Martínez, A. Juan, and F. Casacuberta. 2000. Use of Median String for Classification. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 2, pages 907–910, Barcelona (Spain), September.

John A. Nelder and Roger Mead. 1965. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, January.

Franz Josef Och and Hermann Ney. 2001. Statistical multi-source translation. In *In Machine Translation Summit*, pages 253–258.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pages 160–167, Morristown, NJ, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA. Association for Computational Linguistics.

Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie Dorr. 2007. Combining outputs from multiple machine translation systems. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 228–235, Rochester, New York, April. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and Ralph Weischedel. 2006. A study of translation error rate with targeted human annotation. In *In Proceedings of the Association for Machine Transaltion in the Americas*.

Roy W. Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice minimum bayes-risk decoding for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 620–629, Morristown, NJ, USA. Association for Computational Linguistics.

Ying Zhang and Stephan Vogel. 2004. Measuring confidence intervals for the machine translation evaluation metrics. In *In Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-2004*, pages 4–6.

Adjoining Tree-to-String Translation

Yang Liu, Qun Liu, and **Yajuan Lü**
Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100190, China
{yliu,liuqun,lvyajuan}@ict.ac.cn

Abstract

We introduce synchronous tree adjoining grammars (TAG) into tree-to-string translation, which converts a source tree to a target string. Without reconstructing TAG derivations explicitly, our rule extraction algorithm directly learns tree-to-string rules from aligned Treebank-style trees. As tree-to-string translation casts decoding as a tree parsing problem rather than parsing, the decoder still runs fast when adjoining is included. Less than 2 times slower, the adjoining tree-to-string system improves translation quality by +0.7 BLEU over the baseline system only allowing for tree substitution on NIST Chinese-English test sets.

1 Introduction

Syntax-based translation models, which exploit hierarchical structures of natural languages to guide machine translation, have become increasingly popular in recent years. So far, most of them have been based on synchronous context-free grammars (CFG) (Chiang, 2007), tree substitution grammars (TSG) (Eisner, 2003; Galley et al., 2006; Liu et al., 2006; Huang et al., 2006; Zhang et al., 2008), and inversion transduction grammars (ITG) (Wu, 1997; Xiong et al., 2006). Although these formalisms present simple and precise mechanisms for describing the basic recursive structure of sentences, they are not powerful enough to model some important features of natural language syntax. For example, Chiang (2006) points out that the translation of languages that can stack an unbounded number of clauses in an "inside-out" way (Wu, 1997) provably goes beyond the expressive power of synchronous CFG and TSG. Therefore, it is necessary to find ways to take advantage of more powerful synchronous grammars to improve machine translation.

Synchronous tree adjoining grammars (TAG) (Shieber and Schabes, 1990) are a good candidate. As a formal tree rewriting system, TAG (Joshi et al., 1975; Joshi, 1985) provides a larger domain of locality than CFG to state linguistic dependencies that are far apart since the formalism treats trees as basic building blocks. As a mildly context-sensitive grammar, TAG is conjectured to be powerful enough to model natural languages. Synchronous TAG generalizes TAG by allowing the construction of a pair of trees using the TAG operations of substitution and adjoining on tree pairs. The idea of using synchronous TAG in machine translation has been pursued by several researchers (Abeille et al., 1990; Prigent, 1994; Dras, 1999), but only recently in its probabilistic form (Nesson et al., 2006; DeNeefe and Knight, 2009). Shieber (2007) argues that probabilistic synchronous TAG possesses appealing properties such as expressivity and trainability for building a machine translation system.

However, one major challenge for applying synchronous TAG to machine translation is computational complexity. While TAG requires $O(n^6)$ time for monolingual parsing, synchronous TAG requires $O(n^{12})$ for bilingual parsing. One solution is to use tree insertion grammars (TIG) introduced by Schabes and Waters (1995). As a restricted form of TAG, TIG still allows for adjoining of unbounded trees but only requires $O(n^3)$ time for monolingual parsing. Nesson et al. (2006) firstly demonstrate

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1278–1287,
Portland, Oregon, June 19-24, 2011.

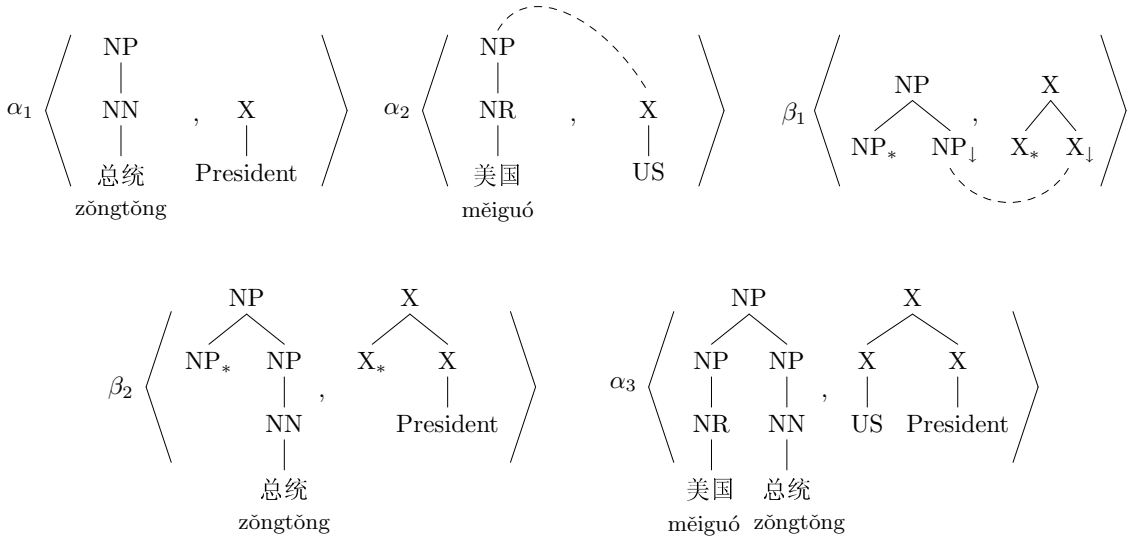


Figure 1: Initial and auxiliary tree pairs. The source side (Chinese) is a Treebank-style linguistic tree. The target side (English) is a purely structural tree using a single non-terminal (X). By convention, substitution and foot nodes are marked with a down arrow ($\downarrow$) and an asterisk ($*$), respectively. The dashed lines link substitution sites (e.g., $\text{NP}_\downarrow$ and $\text{X}_\downarrow$ in β_1) and adjoining sites (e.g., NP and X in α_2) in tree pairs. Substituting the initial tree pair α_1 at the $\text{NP}_\downarrow$-$\text{X}_\downarrow$ node pair in the auxiliary tree pair β_1 yields a derived tree pair β_2, which can be adjoined at NN-X in α_2 to generate α_3.

the use of synchronous TIG for machine translation and report promising results. DeNeefe and Knight (2009) prove that adjoining can improve translation quality significantly over a state-of-the-art string-to-tree system (Galley et al., 2006) that uses synchronous TSG with tractable computational complexity.

In this paper, we introduce synchronous TAG into tree-to-string translation (Liu et al., 2006; Huang et al., 2006), which is the simplest and fastest among syntax-based approaches (Section 2). We propose a new rule extraction algorithm based on GHKM (Galley et al., 2004) that directly induces a synchronous TAG from an aligned and parsed bilingual corpus without converting Treebank-style trees to TAG derivations explicitly (Section 3). As tree-to-string translation takes a source parse tree as input, the decoding can be cast as a tree parsing problem (Eisner, 2003): reconstructing TAG derivations from a derived tree using tree-to-string rules that allow for both substitution and adjoining. We describe how to convert TAG derivations to translation forest (Section 4). We evaluated the new tree-to-string system on NIST Chinese-English tests and obtained consistent improvements (+0.7 BLEU) over the STSG-based baseline system without significant loss in efficiency (1.6 times slower) (Section 5).

2 Model

A synchronous TAG consists of a set of linked elementary tree pairs: **initial** and **auxiliary**. An initial tree is a tree of which the interior nodes are all labeled with non-terminal symbols, and the nodes on the frontier are either words or non-terminal symbols marked with a down arrow ($\downarrow$). An auxiliary tree is defined as an initial tree, except that exactly one of its frontier nodes must be marked as foot node ($*$). The foot node must be labeled with a non-terminal symbol that is the same as the label of the root node.

Synchronous TAG defines two operations to build derived tree pairs from elementary tree pairs: **substitution** and **adjoining**. Nodes in initial and auxiliary tree pairs are linked to indicate the correspondence between substitution and adjoining sites. Figure 1 shows three initial tree pairs (i.e., α_1, α_2, and α_3) and two auxiliary tree pairs (i.e., β_1 and β_2). The dashed lines link substitution nodes (e.g., $\text{NP}_\downarrow$ and $\text{X}_\downarrow$ in β_1) and adjoining sites (e.g., NP and X in α_2) in tree pairs. Substituting the initial tree pair α_1 at

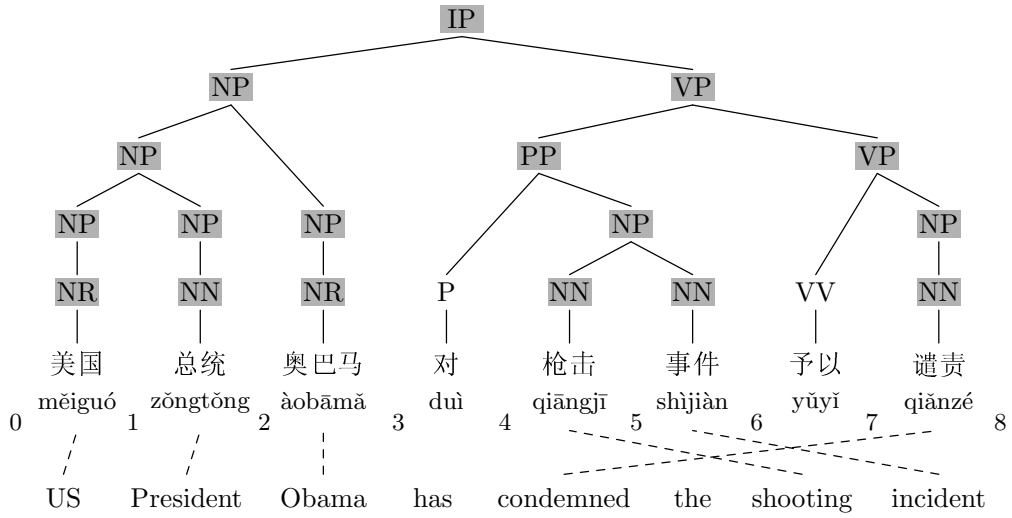


Figure 2: A training example. Tree-to-string rules can be extracted from shaded nodes.

node	minimal initial rule	minimal auxiliary rule
$NR_{0,1}$	[1] (NR měiguó) → US	
$NP_{0,1}$	[2] (NP (x_1:$NR_{\downarrow}$)) → x_1	
$NN_{1,2}$	[3] (NN zǒngtǒng) → President	
$NP_{1,2}$	[4] (NP (x_1:$NN_{\downarrow}$)) → x_1	
$NP_{0,2}$	[5] (NP (x_1:$NP_{\downarrow}$) (x_2:$NP_{\downarrow}$)) → x_1 x_2 [6] ($NP_{0:1}$ (x_1:$NR_{\downarrow}$)) → x_1 [9] ($NP_{0:1}$ (x_1:$NN_{\downarrow}$)) → x_1	[7] (NP (x_1:NP_{*}) (x_2:$NP_{\downarrow}$)) → x_1 x_2 [8] ($NP_{0:2}$ (x_1:NP_{*}) (x_2:$NP_{\downarrow}$)) → x_1 x_2 [10] (NP (x_1:$NP_{\downarrow}$) (x_2:NP_{*})) → x_1 x_2 [11] ($NP_{0:2}$ (x_1:$NP_{\downarrow}$) (x_2:NP_{*})) → x_1 x_2
$NR_{2,3}$	[12] (NR àobāmǎ) → Obama	
$NP_{2,3}$	[13] (NP (x_1:$NR_{\downarrow}$)) → x_1	
$NP_{0,3}$	[14] (NP (x_1:$NP_{\downarrow}$) (x_2:$NP_{\downarrow}$)) → x_1 x_2 [15] ($NP_{0:2}$ (x_1:$NP_{\downarrow}$) (x_2:$NP_{\downarrow}$)) → x_1 x_2 [17] ($NP_{0:1}$ (x_1:$NR_{\downarrow}$)) → x_1 [19] ($NP_{0:1}$ (x_1:$NN_{\downarrow}$)) → x_1 [20] ($NP_{0:1}$ (x_1:$NR_{\downarrow}$)) → x_1	[16] (NP (x_1:NP_{*}) (x_2:$NP_{\downarrow}$)) → x_1 x_2 [18] (NP (x_1:$NP_{\downarrow}$) (x_2:NP_{*})) → x_1 x_2
$NN_{4,5}$	[21] (NN qiāngjī) → shooting	
$NN_{5,6}$	[22] (NN shìjiàn) → incident	
$NP_{4,6}$	[23] (NP (x_1:$NN_{\downarrow}$) (x_2:$NN_{\downarrow}$)) → x_1 x_2	
$PP_{3,6}$	[24] (PP (duì) (x_1:$NP_{\downarrow}$)) → x_1	
$NN_{7,8}$	[25] (NN qiǎnzé) → condemned	
$NP_{7,8}$	[26] (NP (x_1:$NN_{\downarrow}$)) → x_1	
$VP_{6,8}$	[27] (VP (VV yǔyǐ) (x_1:$NP_{\downarrow}$)) → x_1	
$VP_{3,8}$	[28] (VP (x_1:$PP_{\downarrow}$) (x_2:$VP_{\downarrow}$)) → x_2 the x_1 [29] ($VP_{0:1}$ (VV yǔyǐ) (x_1:$NP_{\downarrow}$)) → x_1	[30] (VP (x_1:$PP_{\downarrow}$) (x_2:VP_{*})) → x_2 the x_1
$IP_{0,8}$	[31] (IP (x_1:$NP_{\downarrow}$) (x_2:$VP_{\downarrow}$)) → x_1 has x_2	

Table 1: Minimal initial and auxiliary rules extracted from Figure 2. Note that an adjoining site has a span as subscript. For example, $NP_{0:1}$ in rule 6 indicates that the node is an adjoining site linked to a target node dominating the target string spanning from position 0 to position 1 (i.e., x_1). The target tree is hidden because tree-to-string translation only considers the target surface string.

the $\text{NP}_\downarrow$-$\text{X}_\downarrow$ node pair in the auxiliary tree pair β_1 yields a derived tree pair β_2, which can be adjoined at NN-X in α_2 to generate α_3.

For simplicity, we represent α_2 as a tree-to-string rule:

$$\text{(} \text{NP}_{0:1} \text{ (NR měiguó)) } \rightarrow \text{US}$$

where $\text{NP}_{0:1}$ indicates that the node is an adjoining site linked to a target node dominating the target string spanning from position 0 to position 1 (i.e., "US"). The target tree is hidden because tree-to-string translation only considers the target surface string. Similarly, β_1 can be written as

$$\text{(NP (} x_1\text{:NP}_* \text{) (} x_2\text{:NP}_\downarrow \text{)) } \rightarrow x_1\ x_2$$

where x denotes a non-terminal and the subscripts indicate the correspondence between source and target non-terminals.

The parameters of a probabilistic synchronous TAG are

$$\sum_\alpha P_i(\alpha) = 1 \quad (1)$$

$$\sum_\alpha P_s(\alpha|\eta) = 1 \quad (2)$$

$$\sum_\beta P_a(\beta|\eta) + P_a(\text{NONE}|\eta) = 1 \quad (3)$$

where α ranges over initial tree pairs, β over auxiliary tree pairs, and η over node pairs. $P_i(\alpha)$ is the probability of beginning a derivation with α; $P_s(\alpha|\eta)$ is the probability of substituting α at η; $P_a(\beta|\eta)$ is the probability of adjoining β at η; finally, $P_a(\text{NONE}|\eta)$ is the probability of nothing adjoining at η.

For tree-to-string translation, these parameters can be treated as feature functions of a discriminative framework (Och, 2003) combined with other conventional features such as relative frequency, lexical weight, rule count, language model, and word count (Liu et al., 2006).

3 Rule Extraction

Inducing a synchronous TAG from training data often begins with converting Treebank-style parse trees to TAG derivations (Xia, 1999; Chen and Vijay-Shanker, 2000; Chiang, 2003). DeNeefe and Knight (2009) propose an algorithm to extract synchronous TIG rules from an aligned and parsed bilingual corpus. They first classify tree nodes into heads, arguments, and adjuncts using heuristics (Collins, 2003), then transform a Treebank-style tree into a TIG derivation, and finally extract minimally-sized rules from the derivation tree and the string on the other side, constrained by the alignments. Probabilistic models can be estimated by collecting counts over the derivation trees.

However, one challenge is that there are many TAG derivations that can yield the same derived tree, even with respect to a single grammar. It is difficult to choose appropriate single derivations that enable the resulting grammar to translate unseen data well. DeNeefe and Knight (2009) indicate that the way to reconstruct TIG derivations has a direct effect on final translation quality. They suggest that one possible solution is to use derivation forest rather than a single derivation tree for rule extraction.

Alternatively, we extend the GHKM algorithm (Galley et al., 2004) to *directly* extract tree-to-string rules that allow for both substitution and adjoining from aligned and parsed data. There is no need for transforming a parse tree into a TAG derivation explicitly before rule extraction and all derivations can be easily reconstructed using extracted rules. [1] Our rule extraction algorithm involves two steps: (1) extracting minimal rules and (2) composition.

3.1 Extracting Minimal Rules

Figure 2 shows a training example, which consists of a Chinese parse tree, an English string, and the word alignment between them. By convention, shaded nodes are called **frontier** nodes from which tree-to-string rules can be extracted. Note that the source phrase dominated by a frontier node and its corresponding target phrase are consistent with the word alignment: all words in the source phrase are aligned to all words in the corresponding target phrase and vice versa.

We distinguish between three categories of tree-

[1] Note that our algorithm does not take heads, complements, and adjuncts into consideration and extracts all possible rules with respect to word alignment. Our hope is that this treatment would make our system more robust in the presence of noisy data. It is possible to use the linguistic preferences as features. We leave this for future work.

to-string rules:

1. **substitution rules**, in which the source tree is an initial tree without adjoining sites.

2. **adjoining rules**, in which the source tree is an initial tree with at least one adjoining site.

3. **auxiliary rules**, in which the source tree is an auxiliary tree.

For example, in Figure 1, α_1 is a substitution rule, α_2 is an adjoining rule, and β_1 is an auxiliary rule.

Minimal substitution rules are the same with those in STSG (Galley et al., 2004; Liu et al., 2006) and therefore can be extracted directly using GHKM. By minimal, we mean that the interior nodes are not frontier and cannot be decomposed. For example, in Table 2, rule 1 (for short r_1) is a minimal substitution rule extracted from $\text{NR}_{0,1}$.

Minimal adjoining rules are defined as minimal substitution rules, except that each root node must be an adjoining site. In Table 2, r_2 is a minimal substitution rule extracted from $\text{NP}_{0,1}$. As $\text{NP}_{0,1}$ is a descendant of $\text{NP}_{0,2}$ with the same label, $\text{NP}_{0,1}$ is a possible adjoining site. Therefore, r_6 can be derived from r_2 and licensed as a minimal adjoining rule extracted from $\text{NP}_{0,2}$. Similarly, four minimal adjoining rules are extracted from $\text{NP}_{0,3}$ because it has four frontier descendants labeled with NP.

Minimal auxiliary rules are derived from minimal substitution and adjoining rules. For example, in Table 2, r_7 and r_{10} are derived from the minimal substitution rule r_5 while r_8 and r_{11} are derived from r_{15}. Note that a minimal auxiliary rule can have adjoining sites (e.g., r_8).

Table 1 lists 17 minimal substitution rules, 7 minimal adjoining rules, and 7 minimal auxiliary rules extracted from Figure 2.

3.2 Composition

We can obtain composed rules that capture rich contexts by substituting and adjoining minimal initial and auxiliary rules. For example, the composition of r_{12}, r_{17}, r_{25}, r_{26}, r_{29}, and r_{31} yields an initial rule with two adjoining sites:

(IP ($\text{NP}_{0:1}$ (NR àobāmǎ)) ($\text{VP}_{2:3}$ (VV yǔyǐ) (NP (NN qiǎnzé)))) → Obama has condemned

Note that the source phrase "àobāmǎ . . . yǔyǐ qiǎnzé" is discontinuous. Our model allows both the source and target phrases of an initial rule with adjoining sites to be discontinuous, which goes beyond the expressive power of synchronous CFG and TSG.

Similarly, the composition of two auxiliary rules r_8 and r_{16} yields a new auxiliary rule:

(NP (NP (x_1:NP_*) (x_2:$\text{NP}_\downarrow$)) (x_3:$\text{NP}_\downarrow$)) → $x_1 x_2 x_3$

We first compose initial rules and then compose auxiliary rules, both in a bottom-up way. To maintain a reasonable grammar size, we follow Liu (2006) to restrict that the tree height of a rule is no greater than 3 and the source surface string is no longer than 7.

To learn the probability models $P_i(\alpha)$, $P_s(\alpha|\eta)$, $P_a(\beta|\eta)$, and $P_a(\text{NONE}|\eta)$, we collect and normalize counts over these extracted rules following DeNeefe and Knight (2009).

4 Decoding

Given a synchronous TAG and a derived source tree π, a tree-to-string decoder finds the English yield of the best derivation of which the Chinese yield matches π:

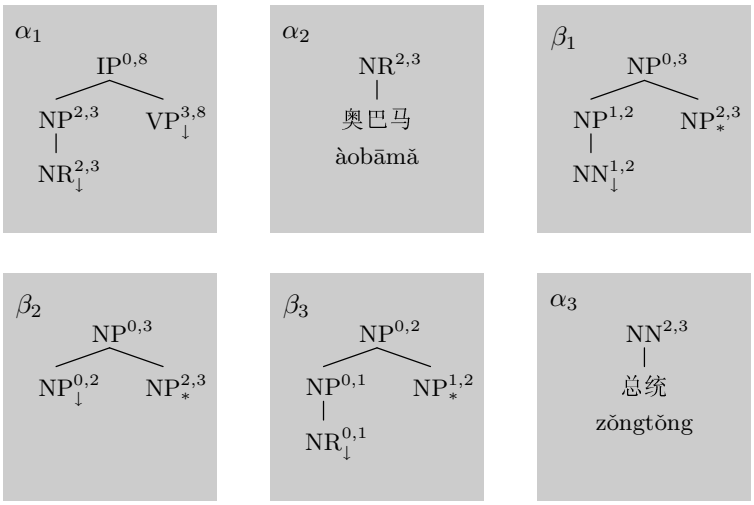
$$\hat{e} = e\left(\underset{D \text{ s.t. } f(D)=\pi}{\arg\max} P(D) \right) \quad (4)$$

This is called **tree parsing** (Eisner, 2003) as the decoder finds ways of decomposing π into elementary trees.

Tree-to-string decoding with STSG is usually treated as **forest rescoring** (Huang and Chiang, 2007) that involves two steps. The decoder first converts the input tree into a translation forest using a translation rule set by pattern matching. Huang et al. (2006) show that this step is a depth-first search with memorization in $O(n)$ time. Then, the decoder searches for the best derivation in the translation forest intersected with n-gram language models and outputs the target string. [2]

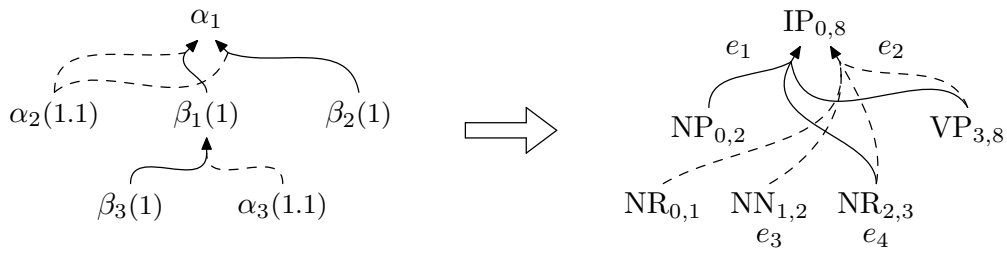
Decoding with STAG, however, poses one major challenge to forest rescoring. As translation forest only supports substitution, it is difficult to construct a translation forest for STAG derivations because of

[2] Mi et al. (2008) give a detailed description of the two-step decoding process. Huang and Mi (2010) systematically analyze the decoding complexity of tree-to-string translation.



elementary tree		translation rule
α_1	r_1	(IP ($\mathrm{NP}_{0:1}$ (x_1:$\mathrm{NR}_{\downarrow}$)) (x_2:$\mathrm{VP}_{\downarrow}$)) $\rightarrow x_1\ x_2$
α_2	r_2	(NR àobāmǎ) $\rightarrow$ Obama
β_1	r_3	(NP ($\mathrm{NP}_{0:1}$ (x_1:$\mathrm{NN}_{\downarrow}$)) (x_2:NP_{*})) $\rightarrow x_1\ x_2$
β_2	r_4	(NP (x_1:$\mathrm{NP}_{\downarrow}$) (x_2:NP_{*})) $\rightarrow x_1\ x_2$
β_3	r_5	(NP (NP (x_1:$\mathrm{NR}_{\downarrow}$)) (x_2:NP_{*})) $\rightarrow x_1\ x_2$
α_3	r_6	(NN zǒngtǒng) $\rightarrow$ President

Figure 3: Matched trees and corresponding rules. Each node in a matched tree is annotated with a span as superscript to facilitate identification. For example, $\mathrm{IP}^{0,8}$ in α_1 indicates that $\mathrm{IP}_{0,8}$ in Figure 2 is matched. Note that its left child $\mathrm{NP}^{2,3}$ is not its direct descendant in Figure 2, suggesting that adjoining is required at this site.



hyperedge		translation rule
e_1	$r_1 + r_4$	(IP (NP (x_1:$\mathrm{NP}_{\downarrow}$) (NP (x_2:$\mathrm{NR}_{\downarrow}$))) (x_3:$\mathrm{VP}_{\downarrow}$) $\rightarrow x_1\ x_2\ x_3$
e_2	$r_1 + r_3 + r_5$	(IP (NP (NP (x_1:$\mathrm{NP}_{\downarrow}$) (x_2:$\mathrm{NP}_{\downarrow}$)) (NP (x_3:$\mathrm{NR}_{\downarrow}$))) (x_4:$\mathrm{VP}_{\downarrow}$)) $\rightarrow x_1\ x_2\ x_3\ x_4$
e_3	r_6	(NN zǒngtǒng) $\rightarrow$ President
e_4	r_2	(NR àobāmǎ) $\rightarrow$ Obama

Figure 4: Converting a derivation forest to a translation forest. In a derivation forest, a node in a derivation forest is a matched elementary tree. A hyperedge corresponds to operations on related trees: substitution (dashed) or adjoining (solid). We use Gorn addresses as tree addresses. $\alpha_2(1.1)$ denotes that α_2 is substituted in the tree α_1 at the node $\mathrm{NR}^{2,3}_{\downarrow}$ of address 1.1 (i.e., the first child of the first child of the root node). As translation forest only supports substitution, we combine trees with adjoining sites to form an equivalent tree without adjoining sites. Rules are composed accordingly (e.g., $r_1 + r_4$).

adjoining. Therefore, we divide forest rescoring for STAG into three steps:

1. **matching**, matching STAG rules against the input tree to obtain a TAG derivation forest;

2. **conversion**, converting the TAG derivation forest into a translation forest;

3. **intersection**, intersecting the translation forest with an n-gram language model.

Given a tree-to-string rule, rule matching is to find a subtree of the input tree that is identical to the source side of the rule. While matching STSG rules against a derived tree is straightforward, it is somewhat non-trivial for STAG rules that move beyond nodes of a local tree. We follow Liu et al. (2006) to enumerate all elementary subtrees and match STAG rules against these subtrees. This can be done by first enumerating all minimal initial and auxiliary trees and then combining them to obtain composed trees, assuming that every node in the input tree is frontier (see Section 3). We impose the same restrictions on the tree height and length as in rule extraction. Figure 3 shows some matched trees and corresponding rules. Each node in a matched tree is annotated with a span as superscript to facilitate identification. For example, $\text{IP}^{0,8}$ in α_1 means that $\text{IP}_{0,8}$ in Figure 2 is matched. Note that its left child $\text{NP}^{2,3}$ is not its direct descendant in Figure 2, suggesting that adjoining is required at this site.

A **TAG derivation tree** specifies uniquely how a derived tree is constructed using elementary trees (Joshi, 1985). A node in a derivation tree is an elementary tree and an edge corresponds to operations on related elementary trees: substitution or adjoining. We introduce **TAG derivation forest**, a compact representation of multiple TAG derivation trees, to encodes all matched TAG derivation trees of the input derived tree.

Figure 4 shows part of a TAG derivation forest. The six matched elementary trees are nodes in the derivation forest. Dashed and solid lines represent substitution and adjoining, respectively. We use Gorn addresses as tree addresses: 0 is the address of the root node, p is the address of the p^{th} child of the root node, and $p \cdot q$ is the address of the q^{th} child of the node at the address p. The derivation forest should be interpreted as follows: α_2 is substituted in the tree α_1 at the node $\text{NR}^{2,3}_{\downarrow}$ of address 1.1 (i.e., the first child of the first child of the root node) and β_1 is adjoined in the tree α_1 at the node $\text{NP}^{2,3}$ of address 1.

To take advantage of existing decoding techniques, it is necessary to convert a derivation forest to a **translation forest**. A hyperedge in a translation forest corresponds to a translation rule. Mi et al. (2008) describe how to convert a derived tree to a translation forest using tree-to-string rules only allowing for substitution. Unfortunately, it is not straightforward to convert a derivation forest including adjoining to a translation forest. To alleviate this problem, we combine initial rules with adjoining sites and associated auxiliary rules to form *equivalent* initial rules without adjoining sites on the fly during decoding.

Consider α_1 in Figure 3. It has an adjoining site $\text{NP}^{2,3}$. Adjoining β_2 in α_1 at the node $\text{NP}^{2,3}$ produces an equivalent initial tree with only substitution sites:

$$(\text{IP}^{0,8} (\text{NP}^{0,3} (\text{NP}^{0,2}_{\downarrow}) (\text{NP}^{2,3} (\text{NR}^{2,3}_{\downarrow}))) (\text{VP}^{3,8}_{\downarrow}))$$

The corresponding composed rule $r_1 + r_4$ has no adjoining sites and can be added to translation forest.

We define that the elementary trees needed to be composed (e.g., α_1 and β_2) form a **composition tree** in a derivation forest. A node in a composition tree is a matched elementary tree and an edge corresponds to adjoining operations. The root node must be an initial tree with at least one adjoining site. The descendants of the root node must all be auxiliary trees. For example, (α_1 (β_2)) and (α_1 (β_1 (β_3))) are two composition trees in Figure 4. The number of children of a node in a composition tree depends on the number of adjoining sites in the node. We use **composition forest** to encode all possible composition trees.

Often, a node in a composition tree may have multiple matched rules. As a large amount of composition trees and composed rules can be identified and constructed on the fly during forest conversion, we used *cube pruning* (Chiang, 2007; Huang and Chiang, 2007) to achieve a balance between translation quality and decoding efficiency.

category	description	number
VP	verb phrase	12.40
NP	noun phrase	7.69
IP	simple clause	7.26
QP	quantifier phrase	0.14
CP	clause headed by C	0.10
PP	preposition phrase	0.09
CLP	classifier phrase	0.02
ADJP	adjective phrase	0.02
LCP	phrase formed by "XP+LC"	0.02
DNP	phrase formed by "XP+DEG"	0.01

Table 2: Top-10 phrase categories of foot nodes and their average occurrences in training corpus.

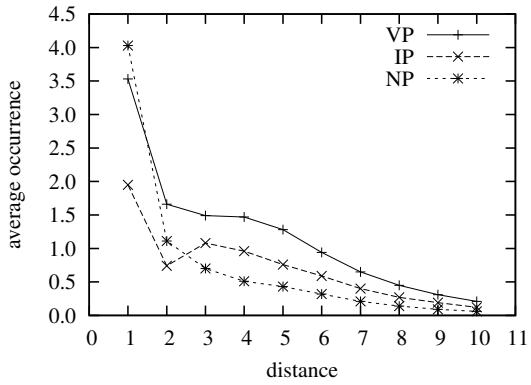


Figure 5: Average occurrences of foot node labels VP, NP, and IP over various distances.

5 Evaluation

We evaluated our adjoining tree-to-string translation system on Chinese-English translation. The bilingual corpus consists of 1.5M sentences with 42.1M Chinese words and 48.3M English words. The Chinese sentences in the bilingual corpus were parsed by an in-house parser. To maintain a reasonable grammar size, we follow Liu et al. (2006) to restrict that the height of a rule tree is no greater than 3 and the surface string's length is no greater than 7. After running GIZA++ (Och and Ney, 2003) to obtain word alignment, our rule extraction algorithm extracted 23.0M initial rules without adjoining sites, 6.6M initial rules with adjoining sites, and 5.3M auxiliary rules. We used the SRILM toolkit (Stolcke, 2002) to train a 4-gram language model on the Xinhua portion of the GIGAWORD corpus, which contains 238M English words. We used the 2002 NIST MT Chinese-English test set as the development set and the 2003-2005 NIST test sets as the test sets. We evaluated translation quality using the BLEU metric, as calculated by mteval-v11b.pl with *case-insensitive* matching of n-grams.

Table 2 shows top-10 phrase categories of foot nodes and their average occurrences in training corpus. We find that VP (verb phrase) is most likely to be the label of a foot node in an auxiliary rule. On average, there are 12.4 nodes labeled with VP are identical to one of its ancestors per tree. NP and IP are also found to be foot node labels frequently. Figure 4 shows the average occurrences of foot node labels VP, NP, and IP over various distances. A distance is the difference of levels between a foot node and the root node. For example, in Figure 2, the distance between $\text{NP}_{0,1}$ and $\text{NP}_{0,3}$ is 2 and the distance between $\text{VP}_{6,8}$ and $\text{VP}_{3,8}$ is 1. As most foot nodes are usually very close to the root nodes, we restrict that a foot node must be the direct descendant of the root node in our experiments.

system	grammar	MT03	MT04	MT05
Moses	-	33.10	33.96	32.17
hierarchical	SCFG	33.40	34.65	32.88
tree-to-string	STSG	33.13	34.55	31.94
	STAG	33.64	**35.28**	**32.71**

Table 3: BLEU scores on NIST Chinese-English test sets. Scores marked in bold are significantly better that those of STSG at $pl.01$ level.

Table 3 shows the BLEU scores on the NIST Chinese-English test sets. Our baseline system is the tree-to-string system using STSG (Liu et al., 2006; Huang et al., 2006). The STAG system outperforms the STSG system significantly on the MT04 and MT05 test sets at $pl.01$ level. Table 3 also gives the results of Moses (Koehn et al., 2007) and an in-house hierarchical phrase-based system (Chiang, 2007). Our STAG system achieves comparable performance with the hierarchical system. The absolute improvement of +0.7 BLEU over STSG is close to the finding of DeNeefe and Knight (2009) on string-to-tree translation. We feel that one major obstacle for achieving further improvement is that composed rules generated on the fly during decoding (e.g., $r_1 + r_3 + r_5$ in Figure 4) usually have too many non-terminals, making cube pruning in the in-

	STSG	STAG
matching	0.086	0.109
conversion	0.000	0.562
intersection	0.946	1.064
other	0.012	0.028
total	1.044	1.763

Table 4: Comparison of average decoding time.

tersection phase suffering from severe search errors (only a tiny fraction of the search space can be explored). To produce the 1-best translations on the MT05 test set that contains 1,082 sentences, while the STSG system used 40,169 initial rules without adjoining sites, the STAG system used 28,046 initial rules without adjoining sites, 1,057 initial rules with adjoining sites, and 1,527 auxiliary rules.

Table 4 shows the average decoding time on the MT05 test set. While rule matching for STSG needs 0.086 second per sentence, the matching time for STAG only increases to 0.109 second. For STAG, the conversion of derivation forests to translation forests takes 0.562 second when we restrict that at most 200 rules can be generated on the fly for each node. As we use cube pruning, although the translation forest of STAG is bigger than that of STSG, the intersection time barely increases. In total, the STAG system runs in 1.763 seconds per sentence, only 1.6 times slower than the baseline system.

6 Conclusion

We have presented a new tree-to-string translation system based on synchronous TAG. With translation rules learned from Treebank-style trees, the adjoining tree-to-string system outperforms the baseline system using STSG without significant loss in efficiency. We plan to introduce left-to-right target generation (Huang and Mi, 2010) into the STAG tree-to-string system. Our work can also be extended to forest-based rule extraction and decoding (Mi et al., 2008; Mi and Huang, 2008). It is also interesting to introduce STAG into tree-to-tree translation (Zhang et al., 2008; Liu et al., 2009; Chiang, 2010).

Acknowledgements

The authors were supported by National Natural Science Foundation of China Contracts 60736014, 60873167, and 60903138. We thank the anonymous reviewers for their insightful comments.

References

Anne Abeille, Yves Schabes, and Aravind Joshi. 1990. Using lexicalized tags for machine translation. In *Proc. of COLING 1990*.

John Chen and K. Vijay-Shanker. 2000. Automated extraction of tags from the penn treebank. In *Proc. of IWPT 2000*.

David Chiang. 2003. Statistical parsing with an automatically extracted tree adjoining grammar. *Data-Oriented Parsing*.

David Chiang. 2006. An introduction to synchronous grammars. ACL Tutorial.

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

David Chiang. 2010. Learning to translate with source and target syntax. In *Proc. of ACL 2010*.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4).

Steve DeNeefe and Kevin Knight. 2009. Synchronous tree adjoining machine translation. In *Proc. of EMNLP 2009*.

Mark Dras. 1999. A meta-level grammar: Redefining synchronous tag for translation and paraphrase. In *Proc. of ACL 1999*.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proc. of ACL 2003*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proc. of NAACL 2004*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. of ACL 2006*.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proc. of ACL 2007*.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proc. of EMNLP 2010*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proc. of AMTA 2006*.

Aravind Joshi, L. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1).

Aravind Joshi. 1985. How much contextsensitivity is necessary for characterizing structural descriptions—tree adjoining grammars. *Natural Language*

Processing—Theoretical, Computational, and Psychological Perspectives.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL 2007 (poster)*, pages 77–80, Prague, Czech Republic, June.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. of ACL 2006*.

Yang Liu, Yajuan Lü, and Qun Liu. 2009. Improving tree-to-tree translation with packed forests. In *Proc. of ACL 2009*.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of EMNLP 2008*.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL/HLT 2008*, pages 192–199, Columbus, Ohio, USA, June.

Rebecca Nesson, Stuart Shieber, and Alexander Rush. 2006. Induction of probabilistic synchronous tree-insertion grammars for machine translation. In *Proc. of AMTA 2006*.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

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

Gilles Prigent. 1994. Synchronous tags and machine translation. In *Proc. of TAG+3*.

Yves Schabes and Richard Waters. 1995. A cubic-time, parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, 21(4).

Stuart M. Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proc. of COLING 1990*.

Stuart M. Shieber. 2007. Probabilistic synchronous tree-adjoining grammars for machine translation: The argument from bilingual dictionaries. In *Proc. of SSST 2007*.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of ICSLP 2002*, pages 901–904.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

Fei Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proc. of the Fifth Natural Language Processing Pacific Rim Symposium*.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proc. of ACL 2006*.

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008. A tree sequence alignment-based tree-to-tree translation model. In *Proc. of ACL 2008*.

Enhancing Language Models in Statistical Machine Translation with Backward N-grams and Mutual Information Triggers

Deyi Xiong, Min Zhang, Haizhou Li
Human Language Technology
Institute for Infocomm Research
1 Fusionopolis Way, #21-01 Connexis, Singapore 138632
{dyxiong, mzhang, hli}@i2r.a-star.edu.sg

Abstract

In this paper, with a belief that a language model that embraces a larger context provides better prediction ability, we present two extensions to standard n-gram language models in statistical machine translation: a backward language model that augments the conventional forward language model, and a mutual information trigger model which captures long-distance dependencies that go beyond the scope of standard n-gram language models. We integrate the two proposed models into phrase-based statistical machine translation and conduct experiments on large-scale training data to investigate their effectiveness. Our experimental results show that both models are able to significantly improve translation quality and collectively achieve up to 1 BLEU point over a competitive baseline.

1 Introduction

Language model is one of the most important knowledge sources for statistical machine translation (SMT) (Brown et al., 1993). The standard n-gram language model (Goodman, 2001) assigns probabilities to hypotheses in the target language conditioning on a context history of the preceding $n-1$ words. Along with the efforts that advance translation models from word-based paradigm to syntax-based philosophy, in recent years we have also witnessed increasing efforts dedicated to extend standard n-gram language models for SMT. We roughly categorize these efforts into two directions: data-volume-oriented and data-depth-oriented.

In the first direction, more data is better. In order to benefit from monolingual corpora (LDC news data or news data collected from web pages) that consist of billions or even trillions of English words, huge language models are built in a distributed manner (Zhang et al., 2006; Brants et al., 2007). Such language models yield better translation results but at the cost of huge storage and high computation.

The second direction digs deeply into monolingual data to build linguistically-informed language models. For example, Charniak et al. (2003) present a syntax-based language model for machine translation which is trained on syntactic parse trees. Again, Shen et al. (2008) explore a dependency language model to improve translation quality. To some extent, these syntactically-informed language models are consistent with syntax-based translation models in capturing long-distance dependencies.

In this paper, we pursue the second direction without resorting to any linguistic resources such as a syntactic parser. With a belief that a language model that embraces a larger context provides better prediction ability, we learn additional information from training data to enhance conventional n-gram language models and extend their ability to capture richer contexts and long-distance dependencies. In particular, we integrate backward n-grams and mutual information (MI) triggers into language models in SMT.

In conventional n-gram language models, we look at the preceding $n-1$ words when calculating the probability of the current word. We henceforth call the previous $n-1$ words plus the current word as **forward n-grams** and a language model built

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1288–1297,
Portland, Oregon, June 19-24, 2011.

on forward n-grams as forward n-gram language model. Similarly, **backward n-grams** refer to the succeeding $n-1$ words plus the current word. We train a backward n-gram language model on backward n-grams and integrate the forward and backward language models together into the decoder. In doing so, we attempt to capture both the preceding and succeeding contexts of the current word.

Different from the backward n-gram language model, the MI trigger model still looks at previous contexts, which however go beyond the scope of forward n-grams. If the current word is indexed as w_i, the farthest word that the forward n-gram includes is w_{i-n+1}. However, the MI triggers are capable of detecting dependencies between w_i and words from w_1 to w_{i-n}. By these triggers ($\{w_k \rightarrow w_i\}, 1 \leq k \leq i-n$), we can capture long-distance dependencies that are outside the scope of forward n-grams.

We integrate the proposed backward language model and the MI trigger model into a state-of-the-art phrase-based SMT system. We evaluate the effectiveness of both models on Chinese-to-English translation tasks with large-scale training data. Compared with the baseline which only uses the forward language model, our experimental results show that the additional backward language model is able to gain about 0.5 BLEU points, while the MI trigger model gains about 0.4 BLEU points. When both models are integrated into the decoder, they collectively improve the performance by up to 1 BLEU point.

The paper is structured as follows. In Section 2, we will briefly introduce related work and show how our models differ from previous work. Section 3 and 4 will elaborate the backward language model and the MI trigger model respectively in more detail, describe the training procedures and explain how the models are integrated into the phrase-based decoder. Section 5 will empirically evaluate the effectiveness of these two models. Section 6 will conduct an in-depth analysis. In the end, we conclude in Section 7.

2 Related Work

Previous work devoted to improving language models in SMT mostly focus on two categories as we mentioned before[1]: large language models (Zhang et al., 2006; Emami et al., 2007; Brants et al., 2007; Talbot and Osborne, 2007) and syntax-based language models (Charniak et al., 2003; Shen et al., 2008; Post and Gildea, 2008). Since our philosophy is fundamentally different from them in that we build contextually-informed language models by using backward n-grams and MI triggers, we discuss previous work that explore these two techniques (backward n-grams and MI triggers) in this section.

Since the context "history" in the backward language model (BLM) is actually the future words to be generated, BLM is normally used in a post-processing where all words have already been generated or in a scenario where sentences are proceeded from the ending to the beginning. Duchateau et al. (2002) use the BLM score as a confidence measure to detect wrongly recognized words in speech recognition. Finch and Sumita (2009) use the BLM in their reverse translation decoder where source sentences are proceeded from the ending to the beginning. Our BLM is different from theirs in that we access the BLM during decoding (rather than after decoding) where source sentences are still proceeded from the beginning to the ending.

Rosenfeld et al. (1994) introduce trigger pairs into a maximum entropy based language model as features. The trigger pairs are selected according to their mutual information. Zhou (2004) also propose an enhanced language model (MI-Ngram) which consists of a standard forward n-gram language model and an MI trigger model. The latter model measures the mutual information of distance-dependent trigger pairs. Our MI trigger model is mostly inspired by the work of these two papers, especially by Zhou's MI-Ngram model (2004). The difference is that our model is distance-independent and, of course, we are interested in an SMT problem rather than a speech recognition one.

Raybaud et al. (2009) use MI triggers in their confidence measures to assess the quality of translation results after decoding. Our method is different from theirs in the MI calculation and trigger pair selection. Mauser et al. (2009) propose bilingual triggers where two source words trigger one target word to

[1]Language model adaptation is not very related to our work so we ignore it.

improve lexical choice of target words. Our analysis (Section 6) show that our monolingual triggers can also help in the selection of target words.

3 Backward Language Model

Given a sequence of words $w_1^m = (w_1...w_m)$, a standard forward n-gram language model assigns a probability $P_f(w_1^m)$ to w_1^m as follows.

$$P_f(w_1^m) = \prod_{i=1}^{m} P(w_i|w_1^{i-1}) \approx \prod_{i=1}^{m} P(w_i|w_{i-n+1}^{i-1}) \quad (1)$$

where the approximation is based on the nth order Markov assumption. In other words, when we predict the current word w_i, we only consider the preceding $n-1$ words $w_{i-n+1}...w_{i-1}$ instead of the whole context history $w_1...w_{i-1}$.

Different from the forward n-gram language model, the backward n-gram language model assigns a probability $P_b(w_1^m)$ to w_1^m by looking at the succeeding context according to

$$P_b(w_1^m) = \prod_{i=1}^{m} P(w_i|w_{i+1}^{m}) \approx \prod_{i=1}^{m} P(w_i|w_{i+1}^{i+n-1}) \quad (2)$$

3.1 Training

For the convenience of training, we invert the order in each sentence in the training data, i.e., from the original order $(w_1...w_m)$ to the reverse order $(w_m...w_1)$. In this way, we can use the same toolkit that we use to train a forward n-gram language model to train a backward n-gram language model without any other changes. To be consistent with training, we also need to reverse the order of translation hypotheses when we access the trained backward language model[2]. Note that the Markov context history of Eq. (2) is $w_{i+n-1}...w_{i+1}$ instead of $w_{i+1}...w_{i+n-1}$ after we invert the order. The words are the same but the order is completely reversed.

3.2 Decoding

In this section, we will present two algorithms to integrate the backward n-gram language model into two kinds of phrase-based decoders respectively: 1) a CKY-style decoder that adopts bracketing transduction grammar (BTG) (Wu, 1997; Xiong et al., 2006) and 2) a standard phrase-based decoder (Koehn et al., 2003). Both decoders translate source sentences from the beginning of a sentence to the ending. Wu (1996) introduce a dynamic programming algorithm to integrate a forward bigram language model with inversion transduction grammar. His algorithm is then adapted and extended for integrating forward n-gram language models into synchronous CFGs by Chiang (2007). Our algorithms are different from theirs in two major aspects

1. The string input to the algorithms is in a reverse order.
2. We adopt a different way to calculate language model probabilities for partial hypotheses so that we can utilize incomplete n-grams.

Before we introduce the integration algorithms, we define three functions $\mathcal{P}$, $\mathcal{L}$, and $\mathcal{R}$ on strings (in a reverse order) over the English terminal alphabet T. The function $\mathcal{P}$ is defined as follows.

$$\mathcal{P}(w_k...w_1) = \underbrace{P(w_k)...P(w_{k-n+2}|w_k...w_{k-n+3})}_{a} \times \underbrace{\prod_{1\le i\le k-n+1} P(w_i|w_{i+n-1}...w_{i+1})}_{b} \quad (3)$$

This function consists of two parts:

- The first part (a) calculates incomplete n-gram language model probabilities for word w_k to w_{k-n+2}. That means, we calculate the unigram probability for w_k ($P(w_k)$), bigram probability for w_{k-1} ($P(w_{k-1}|w_k)$) and so on until we take $n-1$-gram probability for w_{k-n+2} ($P(w_{k-n+2}|w_k...w_{k-n+3})$). This resembles the way in which the forward language model probability in the future cost is computed in the standard phrase-based SMT (Koehn et al., 2003).
- The second part (b) calculates complete n-gram backward language model probabilities for word w_{k-n+1} to w_1.

The function is different from Chiang's p function in that his function p only calculates language model probabilities for the complete n-grams. Since

[2]This is different from the reverse decoding in (Finch and Sumita, 2009) where source sentences are reversed in the order.

we calculate backward language model probabilities during a beginning-to-ending (left-to-right) decoding process, the succeeding context for the current word is either yet to be generated or incomplete in terms of n-grams. The $\mathcal{P}$ function enables us to utilize incomplete succeeding contexts to approximately predict words. Once the succeeding contexts are complete, we can quickly update language model probabilities in an efficient way in our algorithms.

The other two functions $\mathcal{L}$ and $\mathcal{R}$ are defined as follows

$$\mathcal{L}(w_k...w_1) = \begin{cases} w_k...w_{k-n+2}, & \text{if } k \geq n \\ w_k...w_1, & \text{otherwise} \end{cases} \quad (4)$$

$$\mathcal{R}(w_k...w_1) = \begin{cases} w_{n-1}...w_1, & \text{if } k \geq n \\ w_k...w_1, & \text{otherwise} \end{cases} \quad (5)$$

The $\mathcal{L}$ and $\mathcal{R}$ function return the leftmost and rightmost $n-1$ words from a string in a reverse order respectively.

Following Chiang (2007), we describe our algorithms in a deductive system. We firstly show the algorithm[3] that integrates the backward language model into a BTG-style decoder (Xiong et al., 2006) in Figure 1. The item $[A, i, j; l|r]$ indicates that a BTG node A has been constructed spanning from i to j on the source side with the leftmost|rightmost $n-1$ words $l|r$ on the target side. As mentioned before, all target strings assessed by the defined functions ($\mathcal{P}$, $\mathcal{L}$, and $\mathcal{R}$) are in an inverted order (denoted by $\overline{e}$). We only display the backward language model probability for each item, ignoring all other scores such as phrase translation probabilities. The Eq. (8) in Figure 1 shows how we calculate the backward language model probability for the axiom which applies a BTG lexicon rule to translate a source phrase c into a target phrase e. The Eq. (9) and (10) show how we update the backward language model probabilities for two inference rules which combine two neighboring blocks in a straight and inverted order respectively. The fundamental theories behind this update are

$$\mathcal{P}(\overline{e_1 e_2}) = \mathcal{P}(\overline{e_1})\mathcal{P}(\overline{e_2})\frac{\mathcal{P}(\mathcal{R}(\overline{e_2})\mathcal{L}(\overline{e_1}))}{\mathcal{P}(\mathcal{R}(\overline{e_2}))\mathcal{P}(\mathcal{L}(\overline{e_1}))} \quad (6)$$

[3]It can also be easily adapted to integrate the forward n-gram language model.

Function	Value
e_1	$a_1a_2a_3$
e_2	$b_1b_2b_3$
$\mathcal{R}(\overline{e_2})$	b_2b_1
$\mathcal{L}(\overline{e_1})$	a_3a_2
$\mathcal{P}(\mathcal{R}(\overline{e_2}))$	$P(b_2)P(b_1\|b_2)$
$\mathcal{P}(\mathcal{L}(\overline{e_1}))$	$P(a_3)P(a_2\|a_3)$
$\mathcal{P}(\overline{e_1})$	$P(a_3)P(a_2\|a_3)P(a_1\|a_3a_2)$
$\mathcal{P}(\overline{e_2})$	$P(b_3)P(b_2\|b_3)P(b_1\|b_3b_2)$
$\mathcal{P}(\mathcal{R}(\overline{e_2})\mathcal{L}(\overline{e_1}))$	$P(b_2)P(b_1\|b_2)$ $P(a_3\|b_2b_1)P(a_2\|b_1a_3)$
$\mathcal{P}(\overline{e_1e_2})$	$P(b_3)P(b_2\|b_3)P(b_1\|b_3b_2)$ $P(a_3\|b_2b_1)P(a_2\|b_1a_3)P(a_1\|a_3a_2)$

Table 1: Values of $\mathcal{P}$, $\mathcal{L}$, and $\mathcal{R}$ in a 3-gram example .

$$\mathcal{P}(\overline{e_2 e_1}) = \mathcal{P}(\overline{e_1})\mathcal{P}(\overline{e_2})\frac{\mathcal{P}(\mathcal{R}(\overline{e_1})\mathcal{L}(\overline{e_2}))}{\mathcal{P}(\mathcal{R}(\overline{e_1}))\mathcal{P}(\mathcal{L}(\overline{e_2}))} \quad (7)$$

Whenever two strings e_1 and e_2 are concatenated in a straight or inverted order, we can reuse their $\mathcal{P}$ values ($\mathcal{P}(\overline{e_1})$ and $\mathcal{P}(\overline{e_2})$) in terms of dynamic programming. Only the probabilities of boundary words (e.g., $\mathcal{R}(\overline{e_2})\mathcal{L}(\overline{e_1})$ in Eq. (6)) need to be recalculated since they have complete n-grams after the concatenation. Table 1 shows values of $\mathcal{P}$, $\mathcal{L}$, and $\mathcal{R}$ in a 3-gram example which helps to verify Eq. (6). These two equations guarantee that our algorithm can correctly compute the backward language model probability of a sentence stepwise in a dynamic programming framework.[4]

The theoretical time complexity of this algorithm is $\mathcal{O}(m^3|T|^{4(n-1)})$ because in the update parts in Eq. (6) and (7) both the numerator and denominator have up to $2(n-1)$ terminal symbols. This is the same as the time complexity of Chiang's language model integration (Chiang, 2007).

Figure 2 shows the algorithm that integrates the backward language model into a standard phrase-based SMT (Koehn et al., 2003). $\mathcal{V}$ denotes a coverage vector which records source words translated so far. The Eq. (11) shows how we update the backward language model probability for a partial hypothesis when it is extended into a longer hypothesis by a target phrase translating an uncovered source

[4]The start-of-sentence symbol $\langle s \rangle$ and end-of-sentence symbol $\langle /s \rangle$ can be easily added to update the final language model probability when a translation hypothesis covering the whole source sentence is completed.

$$\frac{A \rightarrow c/e}{[A, i, j; \mathcal{L}(\overline{e})|\mathcal{R}(\overline{e})] : \mathcal{P}(\overline{e})} \tag{8}$$

$$\frac{A \rightarrow [A_1, A_2] \quad [A_1, i, k; \mathcal{L}(\overline{e_1})|\mathcal{R}(\overline{e_1})] : \mathcal{P}(\overline{e_1}) \quad [A_2, k+1, j; \mathcal{L}(\overline{e_2})|\mathcal{R}(\overline{e_2})] : \mathcal{P}(\overline{e_2})}{[A, i, j; \mathcal{L}(\overline{e_1 e_2})|\mathcal{R}(\overline{e_1 e_2})] : \mathcal{P}(\overline{e_1})\mathcal{P}(\overline{e_2})\frac{\mathcal{P}(\mathcal{R}(\overline{e_2})\mathcal{L}(\overline{e_1}))}{\mathcal{P}(\mathcal{R}(\overline{e_2}))\mathcal{P}(\mathcal{L}(\overline{e_1}))}} \tag{9}$$

$$\frac{A \rightarrow \langle A_1, A_2 \rangle \quad [A_1, i, k; \mathcal{L}(\overline{e_1})|\mathcal{R}(\overline{e_1})] : \mathcal{P}(\overline{e_1}) \quad [A_2, k+1, j; \mathcal{L}(\overline{e_2})|\mathcal{R}(\overline{e_2})] : \mathcal{P}(\overline{e_2})}{[A, i, j; \mathcal{L}(\overline{e_2 e_1})|\mathcal{R}(\overline{e_2 e_1})] : \mathcal{P}(\overline{e_1})\mathcal{P}(\overline{e_2})\frac{\mathcal{P}(\mathcal{R}(\overline{e_1})\mathcal{L}(\overline{e_2}))}{\mathcal{P}(\mathcal{R}(\overline{e_1}))\mathcal{P}(\mathcal{L}(\overline{e_2}))}} \tag{10}$$

Figure 1: Integrating the backward language model into a BTG-style decoder.

$$\frac{[\mathcal{V}; \mathcal{L}(\overline{e_1})] : \mathcal{P}(\overline{e_1}) \quad c/e_2 : \mathcal{P}(\overline{e_2})}{[\mathcal{V}'; \mathcal{L}(\overline{e_1 e_2})] : \mathcal{P}(\overline{e_1})\mathcal{P}(\overline{e_2})\frac{\mathcal{P}(\mathcal{R}(\overline{e_2})\mathcal{L}(\overline{e_1}))}{\mathcal{P}(\mathcal{R}(\overline{e_2}))\mathcal{P}(\mathcal{L}(\overline{e_1}))}} \tag{11}$$

Figure 2: Integrating the backward language model into a standard phrase-based decoder.

segment. This extension on the target side is similar to the monotone combination of Eq. (9) in that a newly translated phrase is concatenated to an early translated sequence.

4 MI Trigger Model

It is well-known that long-distance dependencies between words are very important for statistical language modeling. However, n-gram language models can only capture short-distance dependencies within an n-word window. In order to model long-distance dependencies, previous work such as (Rosenfeld et al., 1994) and (Zhou, 2004) exploit trigger pairs. A trigger pair is defined as an ordered 2-tuple (x, y) where word x occurs in the preceding context of word y. It can also be denoted in a more visual manner as $x \rightarrow y$ with x being the trigger and y the triggered word[5].

We use pointwise mutual information (PMI) (Church and Hanks, 1990) to measure the strength of the association between x and y, which is defined as follows

$$PMI(x, y) = \log(\frac{P(x, y)}{P(x)P(y)}) \tag{12}$$

[5]In this paper, we require that word x and y occur in the same sentence.

Zhou (2004) proposes a new language model enhanced with MI trigger pairs. In his model, the probability of a given sentence w_1^m is approximated as

$$P(w_1^m) \approx (\prod_{i=1}^{m} P(w_i|w_{i-n+1}^{i-1})) \times \prod_{i=n+1}^{m} \prod_{k=1}^{i-n} \exp(PMI(w_k, w_i, i-k-1)) \tag{13}$$

There are two components in his model. The first component is still the standard n-gram language model. The second one is the MI trigger model which multiples all exponential PMI values for trigger pairs where the current word is the triggered word and all preceding words outside the n-gram window of the current word are triggers. Note that his MI trigger model is distance-dependent since trigger pairs (w_k, w_i) are sensitive to their distance $i-k-1$ (zero distance for adjacent words). Therefore the distance between word x and word y should be taken into account when calculating their PMI.

In this paper, for simplicity, we adopt a distance-independent MI trigger model as follows

$$MI(w_1^m) = \prod_{i=n+1}^{m} \prod_{k=1}^{i-n} \exp(PMI(w_k, w_i)) \tag{14}$$

We integrate the MI trigger model into the log-linear model of machine translation as an additional knowledge source which complements the standard n-gram language model in capturing long-distance dependencies. By MERT (Och, 2003), we are even able to tune the weight of the MI trigger model against the weight of the standard n-gram language model while Zhou (2004) sets equal weights for both models.

4.1 Training

We can use the maximum likelihood estimation method to calculate PMI for each trigger pair by taking counts from training data. Let $C(x, y)$ be the co-occurrence count of the trigger pair (x, y) in the training data. The joint probability of (x, y) is calculated as

$$P(x, y) = \frac{C(x, y)}{\sum_{x,y} C(x, y)} \tag{15}$$

The marginal probabilities of x and y can be deduced from the joint probability as follows

$$P(x) = \sum\nolimits_{y} P(x, y) \tag{16}$$

$$P(y) = \sum\nolimits_{x} P(x, y) \tag{17}$$

Since the number of distinct trigger pairs is $\mathcal{O}(|T|^2)$, the question is how to select valuable trigger pairs. We select trigger pairs according to the following three steps

1. The distance between x and y must not be less than $n - 1$. Suppose we use a 5-gram language model and $y = w_i$, then $x \in \{w_1 ... w_{i-5}\}$.

2. $C(x, y) > c$. In all our experiments we set $c = 10$.

3. Finally, we only keep trigger pairs whose PMI value is larger than 0. Trigger pairs whose PMI value is less than 0 often contain stop words, such as "the", "a". These stop words have very large marginal probabilities due to their high frequencies.

4.2 Decoding

The MI trigger model of Eq. (14) can be directly integrated into the decoder. For the standard phrase-based decoder (Koehn et al., 2003), whenever a partial hypothesis is extended by a new target phrase, we can quickly retrieve the pre-computed PMI value for each trigger pair where the triggered word locates in the newly translated target phrase and the trigger is outside the n-word window of the triggered word. It's a little more complicated to integrate the MI trigger model into the CKY-style phrase-based decoder. But we still can handle it by dynamic programming as follows

$$MI(e_1 e_2) = MI(e_1)MI(e_2)MI(e_1 \rightarrow e_2) \tag{18}$$

where $MI(e_1 \rightarrow e_2)$ represents the PMI values in which a word in e_1 triggers a word in e_2. It is defined as follows

$$MI(e_1 \rightarrow e_2) = \prod_{w_i \in e_2} \prod_{\substack{w_k \in e_1 \\ i-k \geq n}} \exp(PMI(w_k, w_i)) \tag{19}$$

5 Experiments

In this section, we conduct large-scale experiments on NIST Chinese-to-English translation tasks to evaluate the effectiveness of the proposed backward language model and MI trigger model in SMT. Our experiments focus on the following two issues:

1. How much improvements can we achieve by separately integrating the backward language model and the MI trigger model into our phrase-based SMT system?

2. Can we obtain a further improvement if we jointly apply both models?

5.1 System Overview

Without loss of generality[6], we evaluate our models in a phrase-based SMT system which adapts bracketing transduction grammars to phrasal translation (Xiong et al., 2006). The log-linear model of this system can be formulated as

$$\begin{aligned} w(\mathcal{D}) =& M_T(r^l_{1..n_l}) \cdot M_R(r^m_{1..n_m})^{\lambda_R} \\ & \cdot P_{fL}(e)^{\lambda_{fL}} \cdot exp(|e|)^{\lambda_w} \end{aligned} \tag{20}$$

where $\mathcal{D}$ denotes a derivation, $r^l_{1..n_l}$ are the BTG lexicon rules which translate source phrases to target phrases, and $r^m_{1..n_m}$ are the merging rules which combine two neighboring blocks into a larger block in a straight or inverted order. The translation model M_T consists of widely used phrase and lexical translation probabilities (Koehn et al., 2003).

[6] We have discussed how to integrate the backward language model and the MI trigger model into the standard phrase-based SMT system (Koehn et al., 2003) in Section 3.2 and 4.2 respectively.

The reordering model M_R predicts the merging order (straight or inverted) by using discriminative contextual features (Xiong et al., 2006). P_{fL} is the standard forward n-gram language model.

If we simultaneously integrate both the backward language model P_{bL} and the MI trigger model MI into the system, the new log-linear model will be formulated as

$$\begin{aligned} w(\mathcal{D}) = & M_T(r_{1..n_l}^l) \cdot M_R(r_{1..n_m}^m)^{\lambda_R} \cdot P_{fL}(e)^{\lambda_{fL}} \\ & \cdot P_{bL}(e)^{\lambda_{bL}} \cdot MI(e)^{\lambda_{MI}} \cdot exp(|e|)^{\lambda_w} \end{aligned} \quad (21)$$

5.2 Experimental Setup

Our training corpora[7] consist of 96.9M Chinese words and 109.5M English words in 3.8M sentence pairs. We used all corpora to train our translation model and smaller corpora without the United Nations corpus to build a maximum entropy based reordering model (Xiong et al., 2006).

To train our language models and MI trigger model, we used the Xinhua section of the English Gigaword corpus (306 million words). Firstly, we built a forward 5-gram language model using the SRILM toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing. Then we trained a backward 5-gram language model on the same monolingual corpus in the way described in Section 3.1. Finally, we trained our MI trigger model still on this corpus according to the method in Section 4.1. The trained MI trigger model consists of 2.88M trigger pairs.

We used the NIST MT03 evaluation test data as the development set, and the NIST MT04, MT05 as the test sets. We adopted the case-insensitive BLEU-4 (Papineni et al., 2002) as the evaluation metric, which uses the shortest reference sentence length for the brevity penalty. Statistical significance in BLEU differences is tested by paired bootstrap re-sampling (Koehn, 2004).

5.3 Experimental Results

The experimental results on the two NIST test sets are shown in Table 2. When we combine the backward language model with the forward language model, we obtain 0.49 and 0.56 BLEU points over the baseline on the MT-04 and MT-05 test set respectively. Both improvements are statistically significant ($p < 0.01$). The MI trigger model also achieves statistically significant improvements of 0.33 and 0.44 BLEU points over the baseline on the MT-04 and MT-05 respectively.

Model	MT-04	MT-05
Forward (Baseline)	35.67	34.41
Forward+Backward	36.16+	34.97+
Forward+MI	36.00+	34.85+
Forward+Backward+MI	36.76+	35.12+

Table 2: BLEU-4 scores (%) on the two test sets for different language models and their combinations. +: better than the baseline ($p < 0.01$).

When we integrate both the backward language model and the MI trigger model into our system, we obtain improvements of 1.09 and 0.71 BLEU points over the single forward language model on the MT-04 and MT-05 respectively. These improvements are larger than those achieved by using only one model (the backward language model or the MI trigger model).

6 Analysis

In this section, we will study more details of the two models by looking at the differences that they make on translation hypotheses. These differences will help us gain some insights into how the presented models improve translation quality.

Table 3 shows an example from our test set. The italic words in the hypothesis generated by using the backward language model (F+B) exactly match the reference. However, the italic words in the baseline hypothesis fail to match the reference due to the incorrect position of the word "decree" (法令). We calculate the forward/backward language model score (the logarithm of language model probability) for the italic words in both the baseline and F+B hypothesis according to the trained language models. The difference in the forward language model score is only 1.58, which may be offset by differences in other features in the log-linear translation model. On the other hand, the difference in the backward language model score is 3.52. This larger difference may guarantee that the hypothesis generated by F+B

[7] LDC2004E12, LDC2004T08, LDC2005T10, LDC2003E14, LDC2002E18, LDC2005T06, LDC2003E07 and LDC2004T07.

Source	北京 青年报 报导 , 北京 农业局 最近 发出 一连串 的 防治 及 监督 法令
Baseline	*Beijing Youth Daily reported that Beijing Agricultural decree recently issued a series of* control and supervision
F+B	*Beijing Youth Daily reported that Beijing Bureau of Agriculture recently issued a series of* prevention and control laws
Reference	Beijing Youth Daily reported that Beijing Bureau of Agriculture recently issued a series of preventative and monitoring ordinances

Table 3: Translation example from the MT-04 test set, comparing the baseline with the backward language model. F+B: forward+backward language model .

Source	自卫队 此行 之所以 引人瞩目 , 是因为 它 并非 是 一个 孤立 的 事件 。
Baseline	Self-Defense Force 's trip is remarkable , because it <u>was</u> not an isolated incident .
F+M	Self-Defense Force 's trip is remarkable , because it <u>is</u> not an isolated incident .
Reference	The Self-Defense Forces' trip arouses attention because it <u>is</u> not an isolated incident.

Table 4: Translation example from the MT-04 test set, comparing the baseline with the MI trigger model. Both system outputs are not detokenized so that we can see how language model scores are calculated. The underlined words highlight the difference between the enhanced models and the baseline. F+M: forward language model + MI trigger model.

is better enough to be selected as the best hypothesis by the decoder. This suggests that the backward language model is able to provide useful and discriminative information which is complementary to that given by the forward language model.

In Table 4, we present another example to show how the MI trigger model improves translation quality. The major difference in hypotheses of this example is the word choice between "is" and "was". The new system enhanced with the MI trigger model (F+M) selects the former while the baseline selects the latter. The forward language model score for the baseline hypothesis is -26.41, which is higher than the score of the F+M hypothesis -26.67. This could be the reason why the baseline selects the word "was" instead of "is". As can be seen, there is another "is" in the preceding context of the word "was" in the baseline hypothesis. Unfortunately, this word "is" is located just outside the scope of the preceding 5-gram context of "was". The forward 5-gram language model is hence not able to take it into account when calculating the probability of "was". However, this is not a problem for the MI trigger model. Since "is" and "was" rarely co-occur in the same sentence, the PMI value of the trigger pair (is, was)[8] is -1.03 while the PMI value of the trigger pair (is, is) is as high as 0.32. Therefore our MI trigger model selects "is" rather than "was".[9] This example illustrates that the MI trigger model is capable of selecting correct words by using long-distance trigger pairs.

7 Conclusion

We have presented two models to enhance the ability of standard n-gram language models in capturing richer contexts and long-distance dependencies that go beyond the scope of forward n-gram windows. The two models have been integrated into the decoder and have shown to improve a state-of-the-art phrase-based SMT system. The first model is the backward language model which uses backward n-grams to predict the current word. We introduced algorithms that directly integrate the backward language model into a CKY-style and a standard phrase-based decoder respectively. The second model is the MI trigger model that incorporates long-distance trigger pairs into language modeling.

Overall improvements are up to 1 BLEU point on the NIST Chinese-to-English translation tasks with large-scale training data. Further study of the two

[8]Since we remove all trigger pairs whose PMI value is negative, the PMI value of this pair (is, was) is set 0 in practice in the decoder.

[9]The overall MI trigger model scores (the logarithm of Eq. (14)) of the baseline hypothesis and the F+M hypothesis are 2.09 and 2.25 respectively.

models indicates that backward n-grams and long-distance triggers provide useful information to improve translation quality.

In future work, we would like to integrate the backward language model into a syntax-based system in a way that is similar to the proposed algorithm shown in Figure 1. We are also interested in exploring more morphologically- or syntactically-informed triggers. For example, a verb in the past tense triggers another verb also in the past tense rather than the present tense.

References

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867, Prague, Czech Republic, June. Association for Computational Linguistics.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Eugene Charniak, Kevin Knight, and Kenji Yamada. 2003. Syntax-based language models for statistical machine translation. In *Proceedings of MT Summit IX. Intl. Assoc. for Machine Translation*.

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

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Jacques Duchateau, Kris Demuynck, and Patrick Wambacq. 2002. Confidence scoring based on backward language models. In *Proceedings of ICASSP*, pages 221–224, Orlando, FL, April.

Ahmad Emami, Kishore Papineni, and Jeffrey Sorensen. 2007. Large-scale distributed language modeling. In *Proceedings of ICASSP*, pages 37–40, Honolulu, HI, April.

Andrew Finch and Eiichiro Sumita. 2009. Bidirectional phrase-based statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1124–1132, Singapore, August. Association for Computational Linguistics.

Joshua T. Goodman. 2001. A bit of progress in language modeling extended version. Technical report, Microsoft Research.

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 58–54, Edmonton, Canada, May-June.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July.

Arne Mauser, Saša Hasan, and Hermann Ney. 2009. Extending statistical machine translation with discriminative and trigger-based lexicon models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 210–218, Singapore, August. Association for Computational Linguistics.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July.

Matt Post and Daniel Gildea. 2008. Parsers as language models for statistical machine translation. In *Proceedings of AMTA*.

Sylvain Raybaud, Caroline Lavecchia, David Langlois, and Kamel Smaïli. 2009. New confidence measures for statistical machine translation. In *Proceedings of the International Conference on Agents and Artificial Intelligence*, pages 61–68, Porto, Portugal, January.

Roni Rosenfeld, Jaime Carbonell, and Alexander Rudnicky. 1994. Adaptive statistical language modeling: A maximum entropy approach. Technical report, Carnegie Mellon University.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585, Columbus, Ohio, June. Association for Computational Linguistics.

Andreas Stolcke. 2002. Srilm–an extensible language modeling toolkit. In *Proceedings of the 7th International Conference on Spoken Language Processing*, pages 901–904, Denver, Colorado, USA, September.

David Talbot and Miles Osborne. 2007. Randomised language modelling for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 512–519,

Prague, Czech Republic, June. Association for Computational Linguistics.

Dekai Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 152–158, Santa Cruz, California, USA, June.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 521–528, Sydney, Australia, July. Association for Computational Linguistics.

Ying Zhang, Almut Silja Hildebrand, and Stephan Vogel. 2006. Distributed language modeling for n-best list re-ranking. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 216–223, Sydney, Australia, July. Association for Computational Linguistics.

GuoDong Zhou. 2004. Modeling of long distance context dependency. In *Proceedings of Coling*, pages 92–98, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Translating from Morphologically Complex Languages: A Paraphrase-Based Approach

Preslav Nakov
Department of Computer Science
National University of Singapore
13 Computing Drive
Singapore 117417
nakov@comp.nus.edu.sg

Hwee Tou Ng
Department of Computer Science
National University of Singapore
13 Computing Drive
Singapore 117417
nght@comp.nus.edu.sg

Abstract

We propose a novel approach to translating from a morphologically complex language. Unlike previous research, which has targeted word inflections and concatenations, we focus on the pairwise relationship between morphologically related words, which we treat as *potential paraphrases* and handle using paraphrasing techniques at the word, phrase, and sentence level. An important advantage of this framework is that it can cope with derivational morphology, which has so far remained largely beyond the capabilities of statistical machine translation systems. Our experiments translating from Malay, whose morphology is mostly derivational, into English show significant improvements over rivaling approaches based on five automatic evaluation measures (for 320,000 sentence pairs; 9.5 million English word tokens).

1 Introduction

Traditionally, statistical machine translation (SMT) models have assumed that the *word* should be the basic token-unit of translation, thus ignoring any word-internal morphological structure. This assumption can be traced back to the first word-based models of IBM (Brown et al., 1993), which were initially proposed for two languages with limited morphology: French and English. While several significantly improved models have been developed since then, including phrase-based (Koehn et al., 2003), hierarchical (Chiang, 2005), treelet (Quirk et al., 2005), and syntactic (Galley et al., 2004) models, they all preserved the assumption that words should be atomic.

Ignoring morphology was fine as long as the main research interest remained focused on languages with limited (e.g., English, French, Spanish) or minimal (e.g., Chinese) morphology. Since the attention shifted to languages like Arabic, however, the importance of morphology became obvious and several approaches to handle it have been proposed. Depending on the particular language of interest, researchers have paid attention to *word inflections* and *clitics*, e.g., for Arabic, Finnish, and Turkish, or to *noun compounds*, e.g., for German. However, *derivational morphology* has not been specifically targeted so far.

In this paper, we propose a paraphrase-based approach to translating from a morphologically complex language. Unlike previous research, we focus on the pairwise relationship between morphologically related wordforms, which we treat as *potential paraphrases*, and which we handle using paraphrasing techniques at various levels: word, phrase, and sentence level. An important advantage of this framework is that it can cope with various kinds of morphological wordforms, including derivational ones. We demonstrate its potential on Malay, whose morphology is mostly derivational.

The remainder of the paper is organized as follows: Section 2 gives an overview of Malay morphology, Section 3 introduces our paraphrase-based approach to translating from morphologically complex languages, Section 4 describes our dataset and our experimental setup, Section 5 presents and analyses the results, and Section 6 compares our work to previous research. Finally, Section 7 concludes the paper and suggests directions for future work.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1298–1307,
Portland, Oregon, June 19-24, 2011.

2 Malay Morphology and SMT

Malay is an Astronesian language, spoken by about 180 million people. It is official in Malaysia, Indonesia, Singapore, and Brunei, and has two major dialects, sometimes regarded as separate languages, which are mutually intelligible, but occasionally differ in orthography/pronunciation and vocabulary: Bahasa Malaysia (*lit.* 'language of Malaysia') and Bahasa Indonesia (*lit.* 'language of Indonesia').

Malay is an agglutinative language with very rich morphology. Unlike other agglutinative languages such as Finnish, Hungarian, and Turkish, which are rich in both inflectional and derivational forms, Malay morphology is mostly derivational. Inflectionally,[1] Malay is very similar to Chinese: there is no grammatical gender, number, or tense, verbs are not marked for person, etc.

In Malay, new words can be formed by the following three morphological processes:

- **Affixation**, i.e., attaching affixes, which are not words themselves, to a word. These can be prefixes (e.g., *ajar*/'teach' → ***pel****ajar*/'student'), suffixes (e.g., *ajar* → *ajar****an***/'teachings'), circumfixes (e.g., *ajar* → ***peng****ajar****an***/'lesson'), and infixes (e.g., *gigi*/'teeth' → *ge****ri****gi*/'toothed blade'). Infixes only apply to a small number of words and are not productive.

- **Compounding**, i.e., forming a new word by putting two or more existing words together. For example, *kereta*/'car' + *api*/'fire' make *kereta api* and *keretapi* in Bahasa Indonesia and Bahasa Malaysia, respectively, both meaning 'train'. As in English, Malay compounds are written separately, but some stable ones like *kerjasama*/'collaboration' (from *kerja*/'work' and *sama*/'same') are concatenated. Concatenation is also required when a circumfix is applied to a compound, e.g., *ambil alih*/'take over' (*ambil*/'take' + *alih*/'move') is concatenated to form ***peng****ambilalih****an***/'takeover' when targeted by the circumfix *peng-. . .-an*.

- **Reduplication**, i.e., word repetition. In Malay, reduplication requires using a dash. It can be full (e.g., *pelajar-pelajar*/'students'), partial (e.g., *adik-beradik*/'siblings', from *adik*/'younger brother/sister'), and rhythmic (e.g., *gunung-ganang*/'mountains', from the word *gunung*/'mountain').

Malay has very little inflectional morphology, It also has some **clitics**[2], which are not very frequent and are typically spelled concatenated to the preceding word. For example, the politeness marker *lah* can be added to the command *duduk*/'sit down' to yield *duduk****lah***/'please, sit down', and the pronoun *nya* can attach to *kereta* to form *kereta****nya***/'his car'. Note that clitics are not affixes, and clitic attachment is not a word derivation or a word inflection process.

Taken together, affixation, compounding, reduplication, and clitic attachment yield a rich variety of wordforms, which cause data sparseness issues. Moreover, the predominantly derivational nature of Malay morphology limits the applicability of standard techniques such as (1) removing some/all of the source-language inflections, (2) segmenting affixes from the root, and (3) clustering words with the same target translation. For example, if ***pel****ajar*/'student' is an unknown word and lemmatization/stemming reduces it to *ajar*/'teach', would this enable a good translation? Similarly, would segmenting[3] ***pel****ajar* as *peN+ ajar*, i.e., as 'person doing the action' + 'teach', make it possible to generate 'student' (e.g., as opposed to 'teacher')? Finally, if affixes tend to change semantics so much, how likely are we to find morphologically related wordforms that share the same translation? Still, there are many good reasons to believe that morphological processing should help SMT for Malay.

Consider *affixation*, which can yield words with similar semantics that can use each other's translation options, e.g., ***di****ajar*/'be taught (intransitive)' and ***di****ajarkan*/'be taught (transitive)'. However, this cannot be predicted from the affix, e.g., compare *minum*/'drink (verb)' – *minum****an***/'drink (noun)' and *makan*/'eat' – *makan****an***/'food'.

[1] *Inflection* is variation in the form of a word that is obligatory in some given grammatical context. For example, *plays, playing, played* are all inflected forms of the verb *play*. It does not yield a new word and cannot change the part of speech.

[2] A *clitic* is a morpheme that has the syntactic characteristics of a word, but is phonologically bound to another word. For example, *'s* is a clitic in *The Queen of England's crown*.

[3] The prefix *peN* suffers a nasal replacement of the archiphoneme *N* to become *pel* in *pelajar*.

Looking at *compounding*, it is often the case that the semantics of a compound is a specialization of the semantics of its head, and thus the target language translations available for the head could be usable to translate the whole compound, e.g., compare *kerjasama*/'collaboration' and *kerja*/'work'. Alternatively, it might be useful to consider a segmented version of the compound, e.g., *kerja sama*.

Reduplication, among other functions, expresses plural, e.g., *pelajar-pelajar*/'students'. Note, however, that it is not used when a quantity or a number word is present, e.g., *dua pelajar*/'two students' and *banyak pelajar*/'many students'. Thus, if we do not know how to translate *pelajar-pelajar*, it would be reasonable to consider the translation options for *pelajar* since it could potentially contain among its translation options the plural 'students'.

Finally, consider *clitics*. In some cases, a clitic could express a fine-grained distinction such as politeness, which might not be expressible in the target language; thus, it might be feasible to simply remove it. In other cases, e.g., when it is a pronoun, it might be better to segment it out as a separate word.

3 Method

We propose a *paraphrase-based approach* to Malay morphology, where we use paraphrases at three different levels: word, phrase, and sentence level.

First, we transform each development/testing Malay sentence into a *word lattice*, where we add simplified *word-level paraphrasing* alternatives for each morphologically complex word. In the lattice, each alternative w' of an original word w is assigned the weight of $\Pr(w'|w)$, which is estimated using pivoting over the English side of the training bi-text. Then, we generate *sentence-level paraphrases* of the training Malay sentences, in which exactly one morphologically complex word is substituted by a simpler alternative. Finally, we extract additional Malay phrases from these sentences, which we use to augment the phrase table with additional translation options to match the alternative wordforms in the lattice. We assign each such additional phrase p' a probability $\max_p \Pr(p'|p)$, where p is a Malay phrase that is found in the original training Malay text. The probability is calculated using *phrase-level pivoting* over the English side of the training bi-text.

3.1 Morphological Analysis

Given a Malay word, we build a list of morphologically simpler words that could be derived from it; we also generate alternative word segmentations:

(a) words obtainable by affix stripping
e.g., *pelajaran* → *pelajar*, *ajaran*, *ajar*

(b) words that are part of a compound word
e.g., *kerjasama* → *kerja*

(c) words appearing on either side of a dash
e.g., *adik-beradik* → *adik*, *beradik*

(d) words without clitics
e.g., *keretanya* → *kereta*

(e) clitic-segmented word sequences
e.g., *keretanya* → *kereta nya*

(f) dash-segmented wordforms
e.g., *aceh-nias* → *aceh - nias*

(g) combinations of the above.

The list is built by reversing the basic morphological processes in Malay: (a) addresses affixation, (b) handles compounding, (c) takes care of reduplication, and (d) and (e) deal with clitics. Strictly speaking, (f) does not necessarily model a morphological process: it proposes an alternative tokenization, but this could make morphological sense too.

Note that (g) could cause potential problems when interacting with (f), e.g., *adik-beradik* would become *adik - beradik* and then by (a) it would turn into *adik - adik*, which could cause the SMT system to generate two separate translations for the two instances of *adik*. To prevent this, we forbid the application of (f) to reduplications. Taking into account that reduplications can be partial, we only allow (f) if $\frac{|LCS(l,r)|}{\min(|l|,|r|)} < 0.5$, where l and r are the strings to the left and to the right of the dash, respectively, $LCS(x, y)$ is the longest common character subsequence, not necessarily consecutive, of the strings x and y, and $|x|$ is the length of the string x. For example, LCS(*adik*,*beradik*)=*adik*, and thus, the ratio is 1 (≥ 0.5) for *adik-beradik*. Similarly, LCS(*gunung*,*ganang*)=*gnng*, and thus, the ratio is 4/6=0.67 (≥ 0.5) for *gunung-ganang*. However, for *aceh-nias*, it is 1/4=0.25, and thus (f) is applicable.

As an illustration, here are the wordforms we generate for *adik-beradiknya*/'his siblings': *adik*, *adik-beradiknya*, *adik-beradik nya*, *adik-beradik*, *beradiknya*, *beradik nya*, *adik nya*, and *beradik*. And for *berpelajaran*/'is educated', we build the list: *berpelajaran*, *pelajaran*, *pelajar*, *ajaran*, and *ajar*. Note that the lists do include the original word.

To generate the above wordforms, we used two morphological analyzers: a freely available Malay lemmatizer (Baldwin and Awab, 2006), and an in-house re-implementation of the Indonesian stemmer described in (Adriani et al., 2007). Note that these tools' objective is to return a single lemma/stem, e.g., they would return *adik* for *adik-beradiknya*, and *ajar* for *berpelajaran*. However, it was straightforward to modify them to also output the above intermediary wordforms, which the tools were generating internally anyway when looking for the final lemma/stem. Finally, since the two modified analyzers had different strengths and weaknesses, we combined their outputs to increase recall.

3.2 Word-Level Paraphrasing

We perform word-level paraphrasing of the Malay sides of the development and the testing bi-texts.

First, for each Malay word, we generate the above-described list of morphologically simpler words and alternative word segmentations; we think of the words in this list as *word-level paraphrases*. Then, for each development/testing Malay sentence, we generate a lattice encoding all possible paraphrasing options for each individual word.

We further specify a weight for each arc. We assign 1 to the original Malay word w, and $\Pr(w'|w)$ to each paraphrase w' of w, where $\Pr(w'|w)$ is the probability that w' is a *good paraphrase* of w. Note that multi-word paraphrases, e.g., resulting from clitic segmentation, are encoded using a sequence of arcs; in such cases, we assign $\Pr(w'|w)$ to the first arc, and 1 to each subsequent arc.

We calculate the probability $\Pr(w'|w)$ using the training Malay-English bi-text, which we align at the word level using IBM model 4 (Brown et al., 1993), and we observe which English words w and w' are aligned to. More precisely, we use *pivoting* to estimate the probability $\Pr(w'|w)$ as follows:

$$\Pr(w'|w) = \sum_i \Pr(w'|w, e_i)\Pr(e_i|w)$$

Then, following (Callison-Burch et al., 2006; Wu and Wang, 2007), we make the simplifying assumption that w' is conditionally independent of w given e_i, thus obtaining the following expression:

$$\Pr(w'|w) = \sum_i \Pr(w'|e_i)\Pr(e_i|w)$$

We estimate the probability $\Pr(e_i|w)$ directly from the word-aligned training bi-text as follows:

$$\Pr(e_i|w) = \frac{\#(w,e_i)}{\sum_j \#(w,e_j)}$$

where $\#(x, e)$ is the number of times the Malay word x is aligned to the English word e.

Estimating $\Pr(w'|e_i)$ cannot be done directly since w' might not be present on the Malay side of the training bi-text, e.g., because it is a multi-token sequence generated by clitic segmentation. Thus, we think of w' as a pseudoword that stands for the union of all Malay words in the training bi-text that are reducible to w' by our morphological analysis procedure. So, we estimate $\Pr(w'|e_i)$ as follows:

$$\Pr(w'|e_i) = \Pr(\{v : w' \in forms(v)\}|e_i)$$

where $forms(x)$ is the set of the word-level paraphrases[4] for the Malay word x.

Since the training bi-text occurrences of the words that are reducible to w' are distinct, we can rewrite the above as follows:

$$\Pr(w'|e_i) = \sum_{v:w' \in forms(v)} \Pr(v|e_i)$$

Finally, the probability $\Pr(v|e_i)$ can be estimated using maximum likelihood:

$$\Pr(v|e_i) = \frac{\#(v,e_i)}{\sum_u \#(u,e_i)}$$

3.3 Sentence-Level Paraphrasing

In order for the word-level paraphrases to work, there should be phrases in the phrase table that could potentially match them. For some of the words, e.g., the lemmata, there could already be such phrases, but for other transformations, e.g., clitic segmentation, this is unlikely. Thus, we need to augment the phrase table with additional translation options.

One approach would be to modify the phrase table directly, e.g., by adding additional entries, where one or more Malay words are replaced by their paraphrases. This would be problematic since the phrase translation probabilities associated with these new

[4]Note that our paraphrasing process is directed: the paraphrases are morphologically simpler than the original word.

entries would be hard to estimate. For example, the clitics, and even many of the intermediate morphological forms, would not exist as individual words in the training bi-text, which means that there would be no word alignments or lexical probabilities available for them.

Another option would be to generate separate word alignments for the original training bi-text and for a version of it where the source (Malay) side has been paraphrased. Then, the two bi-texts and their word alignments would be concatenated and used to build a phrase table (Dyer, 2007; Dyer et al., 2008; Dyer, 2009). This would solve the problems with the word alignments and the phrase pair probabilities estimations in a principled manner, but it would require choosing for each word only one of the paraphrases available to it, while we would prefer to have a way to allow all options. Moreover, the paraphrased and the original versions of the corpus would be given equal weights, which might not be desirable. Finally, since the two versions of the bi-text would be word-aligned separately, there would be no interaction between them, which might lead to missed opportunities for improved alignments in both parts of the bi-text (Nakov and Ng, 2009).

We avoid the above issues by adopting a sentence-level paraphrasing approach. Following the general framework proposed in (Nakov, 2008), we first create multiple paraphrased versions of the source-side sentences of the training bi-text. Then, each paraphrased source sentence is paired with its original translation. This augmented bi-text is word-aligned and a phrase table T' is built from it, which is merged with a phrase table T for the original bi-text. The merged table contains all phrase entries from T, and the entries for the phrase pairs from T' that are not in T. Following Nakov and Ng (2009), we add up to three additional indicator features (taking the values 0.5 and 1) to each entry in the merged phrase table, showing whether the entry came from (1) T only, (2) T' only, or (3) both T and T'. We also try using the first one or two features only. We set all feature weights using minimum error rate training (Och, 2003), and we optimize their number (one, two, or three) on the development dataset.[5]

[5]In theory, we should re-normalize the probabilities; in practice, this is not strictly required by the log-linear SMT model.

Each of our paraphrased sentences differs from its original sentence by a single word, which prevents combinatorial explosions: on average, we generate 14 paraphrased versions per input sentence. It further ensures that the paraphrased parts of the sentences will not dominate the word alignments or the phrase pairs, and that there would be sufficient interaction at word alignment time between the original sentences and their paraphrased versions.

3.4 Phrase-Level Paraphrasing

While our sentence-level paraphrasing informs the decoder about the origin of each phrase pair (original or paraphrased bi-text), it provides no indication about how good the phrase pairs from the paraphrased bi-text are likely to be.

Following Callison-Burch et al. (2006), we further augment the phrase table with one additional feature whose value is 1 for the phrase pairs coming from the original bi-text, and $\max_p \Pr(p'|p)$ for the phrase pairs extracted from the paraphrased bi-text. Here p is a Malay phrase from T, and p' is a Malay phrase from T' that does not exist in T but is obtainable from p by substituting one or more words in p with their derivationally related forms generated by morphological analysis. The probability $\Pr(p'|p)$ is calculated using phrase-level pivoting through English in the original phrase table T as follows (unlike word-level pivoting, here e_i is an English *phrase*):

$$\Pr(p'|p) = \sum_i \Pr(p'|e_i)\Pr(e_i|p)$$

We estimate the probabilities $\Pr(e_i|p)$ and $\Pr(p'|e_i)$ as we did for word-level pivoting, except that this time we use the list of the phrase pairs extracted from the original training bi-text, while before we used IBM model 4 word alignments. When calculating $\Pr(p'|e_i)$, we think of p' as the set of all possible Malay phrases q in T that are reducible to p' by morphological analysis of the words they contain. This can be rewritten as follows:

$$\Pr(p'|e_i) = \sum_{q:p' \in par(q)} \Pr(q|e_i)$$

where $par(q)$ is the set of all possible phrase-level paraphrases for the Malay phrase q.

The probability $\Pr(q|e_i)$ is estimated using maximum likelihood from the list of phrase pairs. There is no combinatorial explosion here, since the phrases are short and contain very few paraphrasable words.

Number of sentence pairs	**1K**	**2K**	**5K**	**10K**	**20K**	**40K**	**80K**	**160K**	**320K**
Number of English words	30K	60K	151K	301K	602K	1.2M	2.4M	4.7M	9.5M
baseline	23.81	27.43	31.53	33.69	36.68	38.49	40.53	41.80	43.02
lemmatize all	22.67	26.20	29.68	31.53	33.91	35.64	37.17	38.58	39.68
	-1.14	-1.23	-1.85	-2.16	-2.77	-2.85	-3.36	-3.22	-3.34
'noisier' channel model (Dyer, 2007)	23.27	**28.42**	**32.66**	33.69	37.16	38.14	39.79	41.76	42.77
	-0.54	**+0.99**	**+1.13**	+0.00	**+0.48**	-0.35	-0.74	-0.04	-0.25
lattice + sent-par (orig+lemma)	**24.71**	**28.65**	**32.42**	**34.95**	**37.32**	38.40	39.82	41.97	43.36
	+0.90	**+1.22**	**+0.89**	**+1.26**	**+0.64**	-0.09	-0.71	+0.17	+0.34
lattice + sent-par	**24.97**	**29.11**	**33.03**	**35.12**	**37.39**	38.73	**41.04**	42.24	**43.52**
	+1.16	**+1.68**	**+1.50**	**+1.43**	**+0.71**	+0.24	**+0.51**	**+0.44**	**+0.50**
lattice + sent-par + word-par	**25.14**	**29.17**	**33.00**	**35.09**	**37.39**	38.76	*40.75*	**42.23**	**43.58**
	+1.33	**+1.74**	**+1.47**	**+1.40**	**+0.71**	+0.27	*+0.22*	**+0.43**	**+0.56**
lattice + sent-par + word-par + phrase-par	**25.27**	**29.19**	**33.35**	**35.23**	**37.46**	**39.00**	**40.95**	**42.30**	**43.73**
	+1.46	**+1.76**	**+1.82**	**+1.54**	**+0.78**	**+0.51**	**+0.42**	**+0.50**	**+0.71**

Table 1: **Evaluation results.** Shown are BLEU scores and improvements over the baseline (in %) for different numbers of training sentences. Statistically significant improvements are in **bold** for $p < 0.01$ and in *italic* for $p < 0.05$.

4 Experiments

4.1 Data

We created our Malay-English training and development datasets from data that we downloaded from the Web and then sentence-aligned using various heuristics. Thus, we ended up with 350,003 *training* sentence pairs, including 10.4M English and 9.7M Malay word tokens. We further downloaded 49.8M word tokens of monolingual English text, which we used for *language modeling*.

For *testing*, we used 1,420 sentences with 28.8K Malay word tokens, which were translated by three human translators, yielding translations of 32.8K, 32.4K, and 32.9K English word tokens, respectively. For *development*, we used 2,000 sentence pairs of 63.4K English and 58.5K Malay word tokens.

4.2 General Experimental Setup

First, we tokenized and lowercased all datasets: training, development, and testing. We then built directed word-level alignments for the training bitext for English→Malay and for Malay→English using IBM model 4 (Brown et al., 1993), which we symmetrized using the intersect+grow heuristic (Och and Ney, 2003). Next, we extracted phrase-level translation pairs of maximum length seven, which we scored and used to build a phrase table where each phrase pair is associated with the following five standard feature functions: forward and reverse phrase translation probabilities, forward and reverse lexicalized phrase translation probabilities, and phrase penalty.

We trained a log-linear model using the following standard SMT feature functions: trigram language model probability, word penalty, distance-based distortion cost, and the five feature functions from the phrase table. We set all weights on the development dataset by optimizing BLEU (Papineni et al., 2002) using minimum error rate training (Och, 2003), and we plugged them in a beam search decoder (Koehn et al., 2007) to translate the Malay test sentences to English. Finally, we detokenized the output, and we evaluated it against the three reference translations.

4.3 Systems

Using the above general experimental setup, we implemented the following baseline systems:

- **baseline**. This is the default system, which uses no morphological processing.
- **lemmatize all**. This is the second baseline that uses lemmatized versions of the Malay side of the training, development and testing datasets.
- **'noisier' channel model**.[6] This is the model of Dyer (2007). It uses 0-1 weights in the lattice and only allows lemmata as alternative word-forms; it uses no sentence-level or phrase-level paraphrases.

[6]We also tried the word segmentation model of Dyer (2009) as implemented in the *cdec* decoder (Dyer et al., 2010), which learns word segmentation lattices from raw text in an unsupervised manner. Unfortunately, it could not learn meaningful word segmentations for Malay, and thus we do not compare against it. We believe this may be due to its focus on word segmentation, which is of limited use for Malay.

sent.	system	1-gram	2-gram	3-gram	4-gram
1k	baseline	59.78	29.60	17.36	10.46
	paraphrases	62.23	31.19	18.53	11.35
2k	baseline	64.20	33.46	20.41	12.92
	paraphrases	66.38	35.42	21.97	14.06
5k	baseline	68.12	38.12	24.20	15.72
	paraphrases	70.41	40.13	25.71	17.02
10k	baseline	70.13	40.67	26.15	17.27
	paraphrases	72.04	42.28	27.55	18.36
20k	baseline	73.19	44.12	29.14	19.50
	paraphrases	73.28	44.43	29.77	20.31
40k	baseline	74.66	45.97	30.70	20.83
	paraphrases	75.47	46.54	31.09	21.17
80k	baseline	75.72	48.08	32.80	22.59
	paraphrases	76.03	48.47	33.20	23.00
160k	baseline	76.55	49.21	34.09	23.78
	paraphrases	77.14	49.89	34.57	24.06
320k	baseline	77.72	50.54	35.19	24.78
	paraphrases	78.03	51.24	35.99	25.42

Table 2: **Detailed BLEU n-gram precision scores:** in %, for different numbers of training sentence pairs, for *baseline* and *lattice + sent-par + word-par + phrase-par*.

Sent.	System	BLEU	NIST	TER	METEOR	TESLA
1k	baseline	23.81	6.7013	64.50	49.26	1.6794
	paraphrases	25.27	6.9974	63.03	52.32	1.7579
2k	baseline	27.43	7.3790	61.03	54.29	1.8718
	paraphrases	29.19	7.7306	59.37	57.32	2.0031
5k	baseline	31.53	8.0992	57.12	59.09	2.1172
	paraphrases	33.35	8.4127	55.41	61.67	2.2240
10k	baseline	33.69	8.5314	55.24	62.26	2.2656
	paraphrases	35.23	8.7564	53.60	63.97	2.3634
20k	baseline	36.68	8.9604	52.56	64.67	2.3961
	paraphrases	37.46	9.0941	52.16	66.42	2.4621
40k	baseline	38.49	9.3016	51.20	66.68	2.5166
	paraphrases	39.00	9.4184	50.68	67.60	2.5604
80k	baseline	40.53	9.6047	49.88	68.77	2.6331
	paraphrases	40.95	9.6289	49.09	69.10	2.6628
160k	baseline	41.80	9.7479	48.97	69.59	2.6887
	paraphrases	42.30	9.8062	48.29	69.62	2.7049
320k	baseline	43.02	9.8974	47.44	70.23	2.7398
	paraphrases	43.73	9.9945	47.07	70.87	2.7856

Table 3: **Results for different evaluation measures:** for *baseline* and *lattice + sent-par + word-par + phrase-par* (in % for all measures except for NIST).

Our full morphological paraphrasing system is **lattice + sent-par + word-par + phrase-par**. We also experimented with some of its components turned off. **lattice + sent-par + word-par** excludes the additional feature from phrase-level paraphrasing. **lattice + sent-par** has all the morphologically simpler derived forms in the lattice during decoding, but their weights are uniformly set to 0 rather than obtained using pivoting from word alignments. Finally, in order to compare closely to the 'noisier' channel model, we further limited the morphological variants of **lattice + sent-par** in the lattice to lemmata only in **lattice + sent-par (orig+lemma)**.

5 Results and Discussion

The experimental results are shown in Table 1.

First, we can see that *lemmatize all* has a consistently disastrous effect on BLEU, which shows that Malay morphology does indeed contain information that is important when translating to English.

Second, Dyer (2007)'s *'noisier' channel model* helps for small datasets only. It performs worse than *lattice + sent-par (orig+lemma)*, from which it differs in the phrase table only; this confirms the importance of our sentence-level paraphrasing.

Moving down to *lattice + sent-par*, we can see that using multiple morphological wordforms instead of just lemmata has a consistently positive impact on BLEU for datasets of all sizes.

Adding weights obtained using word-level pivoting in *lattice + sent-par + word-par* helps a bit more, and also using phrase-level paraphrasing weights yields even bigger further improvements for *lattice + sent-par + word-par + phrase-par*.

Overall, our morphological paraphrases yield statistically significant improvements ($p < 0.01$) in BLEU, according to Collins et al. (2005)'s sign test, for bi-texts as large as 320,000 sentence pairs.

A closer look at BLEU. Table 2 shows detailed n-gram BLEU precision scores for n=1,2,3,4. Our system outperforms the baseline on all precision scores and for all numbers of training sentences.

Other evaluation measures. Table 3 reports the results for five evaluation measures: BLEU and NIST 11b, TER 0.7.25 (Snover et al., 2006), METEOR 1.0 (Lavie and Denkowski, 2009), and TESLA (Liu et al., 2010). Our system consistently outperforms the baseline for all measures.

Example translations. Table 4 shows two translation examples. In the first example, the reduplication *bekalan-bekalan* ('supplies') is an unknown word, and was left untranslated by the baseline system. It was not a problem for our system though, which first paraphrased it as *bekalan* and then translated it as *supply*. Even though this is still wrong (we need the plural *supplies*), it is arguably preferable to passing the word untranslated; it also allowed for a better translation of the surrounding context.

src :	Mercy Relief telah menghantar 17 khemah khas bernilai $5,000 setiap satu yang boleh menampung kelas seramai 30 pelajar, selain **bekalan-bekalan lain seperti 500 khemah biasa**, barang makanan dan ubat-ubatan untuk mangsa gempa Sichuan.
ref1:	Mercy Relief has sent 17 special tents valued at $5,000 each, that can accommodate a class of 30 students, including **other aid supplies such as 500 normal tents**, food and medicine for the victims of Sichuan quake.
base:	mercy relief has sent 17 special tents worth $5,000 each class could accommodate a total of 30 students, besides ***other bekalan-bekalan 500 tents as usual***, foodstuff and medicines for sichuan quake relief.
para:	mercy relief has sent 17 special tents worth $5,000 each class could accommodate a total of 30 students, besides **other supply such as 500 tents**, food and medicines for sichuan quake relief.
src :	Walaupun hidup susah, kami tetap berusaha untuk **menjalani kehidupan** seperti biasa.
ref1:	Even though life is difficult, we are still trying to **go through life** as usual.
base:	despite the hard life, we will always strive to ***undergo training*** as usual.
para:	despite the hard life, we will always strive to **live** normal.

Table 4: **Example translations**. For each example, we show a source sentence (src), one of the three reference translations (ref1), and the outputs of *baseline* (base) and of *lattice + sent-par + word-par + phrase-par* (para).

In the second example, the baseline system translated *menjalani kehidupan* (lit. 'go through life') as *undergo training*, because of a bad phrase pair, which was extracted from wrong word alignments. Note that the words *menjalani* ('go through') and *kehidupan* ('life/existence') are derivational forms of *jalan* ('go') and *hidup* ('life/living'), respectively. Thus, in the paraphrasing system, they were involved in sentence-level paraphrasing, where the alignments were improved. While the wrong phrase pair was still available, the system chose a better one from the paraphrased training bi-text.

6 Related Work

Most research in SMT for a morphologically rich **source** language has focused on inflected *forms of the same* word. The assumption is that they would have similar semantics and thus could have the same translation. Researchers have used *stemming* (Yang and Kirchhoff, 2006), *lemmatization* (Al-Onaizan et al., 1999; Goldwater and McClosky, 2005; Dyer, 2007), or *direct clustering* (Talbot and Osborne, 2006) to identify such groups of words and use them as *equivalence classes* or as possible *alternatives* in translation. Frameworks for the simultaneous use of different word-level representations have been proposed as well (Koehn and Hoang, 2007).

A second important line of research has focused on *word segmentation*, which is useful for languages like German, which are rich in *compound words* that are spelled concatenated (Koehn and Knight, 2003; Yang and Kirchhoff, 2006), or like Arabic, Turkish, Finnish, and, to a lesser extent, Spanish and Italian, where *clitics* often attach to the preceding word (Habash and Sadat, 2006). For languages with more or less regular inflectional morphology like Arabic or Turkish, another good idea is to segment words into *morpheme sequences*, e.g., prefix(es)-stem-suffix(es), which can be used instead of the original words (Lee, 2004) or in addition to them. This can be achieved using a lattice input to the translation system (Dyer et al., 2008; Dyer, 2009).

Unfortunately, none of these general lines of research suits Malay well, whose compounds are rarely concatenated, clitics are not so frequent, and morphology is mostly derivational, and thus likely to generate words whose semantics substantially differs from the semantics of the original word. Therefore, we cannot expect the existence of equivalence classes: it is only occasionally that two derivationally related wordforms would share the same target language translation. Thus, instead of looking for equivalence classes, we have focused on the pairwise relationship between derivationally related wordforms, which we treat as *potential paraphrases*.

Our approach is an extension of the *'noisier' channel model* of Dyer (2007). He starts by generating separate word alignments for the original training bi-text and for a version of it where the source side has been lemmatized. Then, the two bi-texts and their word alignments are concatenated and used to build a phrase table. Finally, the source sides of the development and the test datasets are converted into confusion networks where additional arcs are added for word lemmata. The arc weights are set to 1 for the original wordforms and to 0 for the lemmata. In contrast, we provide *multiple* paraphrasing alternatives for each morphologically complex word, including derivational forms that occupy intermediary positions between the original wordform

and its lemma. Note that some of those paraphrasing alternatives are *multi-word*, and thus we use a *lattice* instead of a confusion network. Moreover, we give *different weights* to the different alternatives rather then assigning them all 0.

Second, our work is related to that of Dyer et al. (2008), who use a lattice to add a single alternative clitic-segmented version of the original word for Arabic. However, we provide *multiple* alternatives. We also include *derivational forms* in addition to clitic-segmented ones, and we give *different weights* to the different alternatives (instead of 0).

Third, our work is also related to that of Dyer (2009), who uses a lattice to add multiple alternative segmented versions of the original word for German, Hungarian, and Turkish. However, we focus on *derivational morphology* rather than on clitics and inflections, add *derivational forms* in addition to clitic-segmented ones, and use *cross-lingual word pivoting* to estimate paraphrase probabilities.

Finally, our work is related to that of Callison-Burch et al. (2006), who use cross-lingual pivoting to generate phrase-level paraphrases with corresponding probabilities. However, our paraphrases are derived through *morphological analysis*; thus, we do not need corpora in additional languages.

7 Conclusion and Future Work

We have presented a novel approach to translating from a morphologically complex language, which uses paraphrases and paraphrasing techniques at three different levels of translation: word-level, phrase-level, and sentence-level. Our experiments translating from Malay, whose morphology is mostly derivational, into English have shown significant improvements over rivaling approaches based on several automatic evaluation measures.

In future work, we want to improve the probability estimations for our paraphrasing models. We also want to experiment with other morphologically complex languages and other SMT models.

Acknowledgments

This work was supported by research grant POD0713875. We would like to thank the anonymous reviewers for their detailed and constructive comments, which have helped us improve the paper.

References

Mirna Adriani, Jelita Asian, Bobby Nazief, S. M.M. Tahaghoghi, and Hugh E. Williams. 2007. Stemming Indonesian: A confix-stripping approach. *ACM Transactions on Asian Language Information Processing*, 6:1–33.

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz-Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. 1999. Statistical machine translation. Technical report, JHU Summer Workshop.

Timothy Baldwin and Su'ad Awab. 2006. Open source corpus analysis tools for Malay. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, LREC '06, pages 2212–2215.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, HLT-NAACL '06, pages 17–24.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, ACL '05, pages 263–270.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, ACL '05, pages 531–540.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, ACL '08, pages 1012–1020.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, ACL '10, pages 7–12.

Christopher Dyer. 2007. The 'noisier channel': translation from morphologically complex languages. In *Proceedings of the Second Workshop on Statistical Machine Translation*, WMT '07, pages 207–211.

Chris Dyer. 2009. Using a maximum entropy model to build segmentation lattices for MT. In *Proceedings of Human Language Technologies: The 2009 Annual*

Conference of the North American Chapter of the Association for Computational Linguistics, NAACL '09, pages 406–414.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, HLT-NAACL '04, pages 273–280.

Sharon Goldwater and David McClosky. 2005. Improving statistical MT through morphological analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT-EMNLP '05, pages 676–683.

Nizar Habash and Fatiha Sadat. 2006. Arabic preprocessing schemes for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, HLT-NAACL '06, pages 49–52.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '07, pages 868–876.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '03, pages 187–193.

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

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

Alon Lavie and Michael J. Denkowski. 2009. The meteor metric for automatic evaluation of machine translation. *Machine Translation*, 23:105–115.

Young-Suk Lee. 2004. Morphological analysis for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, HLT-NAACL '04, pages 57–60.

Chang Liu, Daniel Dahlmeier, and Hwee Tou Ng. 2010. TESLA: Translation evaluation of sentences with linear-programming-based analysis. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT '10, pages 354–359.

Preslav Nakov and Hwee Tou Ng. 2009. Improved statistical machine translation for resource-poor languages using related resource-rich languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, EMNLP '09, pages 1358–1367.

Preslav Nakov. 2008. Improved statistical machine translation using monolingual paraphrases. In *Proceedings of the 18th European Conference on Artificial Intelligence*, ECAI '08, pages 338–342.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, ACL '02, pages 311–318.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, ACL '05, pages 271–279.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the Association for Machine Translation in the Americas*, AMTA '06, pages 223–231.

David Talbot and Miles Osborne. 2006. Modelling lexical redundancy for machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, COLING-ACL '06, pages 969–976.

Hua Wu and Haifeng Wang. 2007. Pivot language approach for phrase-based statistical machine translation. *Machine Translation*, 21(3):165–181.

Mei Yang and Katrin Kirchhoff. 2006. Phrase-based backoff models for machine translation of highly inflected languages. In *Proceedings of the European Chapter of the Association for Computational Linguistics*, EACL '06, pages 41–48.

Gappy Phrasal Alignment by Agreement

Mohit Bansal*
UC Berkeley, CS Division
mbansal@cs.berkeley.edu

Chris Quirk
Microsoft Research
chrisq@microsoft.com

Robert C. Moore
Google Research
robert.carter.moore@gmail.com

Abstract

We propose a principled and efficient phrase-to-phrase alignment model, useful in machine translation as well as other related natural language processing problems. In a hidden semi-Markov model, word-to-phrase and phrase-to-word translations are modeled directly by the system. Agreement between two directional models encourages the selection of parsimonious phrasal alignments, avoiding the overfitting commonly encountered in unsupervised training with multi-word units. Expanding the state space to include "gappy phrases" (such as French *ne ⋆ pas*) makes the alignment space more symmetric; thus, it allows agreement between discontinuous alignments. The resulting system shows substantial improvements in both alignment quality and translation quality over word-based Hidden Markov Models, while maintaining asymptotically equivalent runtime.

1 Introduction

Word alignment is an important part of statistical machine translation (MT) pipelines. Phrase tables containing pairs of source and target language phrases are extracted from word alignments, forming the core of phrase-based statistical machine translation systems (Koehn et al., 2003). Most syntactic machine translation systems extract synchronous context-free grammars (SCFGs) from aligned syntactic fragments (Galley et al., 2004; Zollmann et al., 2006), which in turn are derived from bilingual word alignments and syntactic parses. Alignment is also used in various other NLP problems such as entailment, paraphrasing, question answering, summarization and spelling correction.

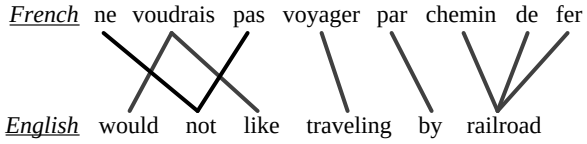


Figure 1: French-English pair with complex word alignment.

A limitation to word-based alignment is undesirable. As seen in the French-English example in Figure 1, many sentence pairs are naturally aligned with multi-word units in both languages (*chemin de fer*; *would ⋆ like*, where ⋆ indicates a gap). Much work has addressed this problem: generative models for direct phrasal alignment (Marcu and Wong, 2002), heuristic word-alignment combinations (Koehn et al., 2003; Och and Ney, 2003), models with pseudo-word collocations (Lambert and Banchs, 2006; Ma et al., 2007; Duan et al., 2010), synchronous grammar based approaches (Wu, 1997), etc. Most have a large state-space, using constraints and approximations for efficient inference.

We present a new phrasal alignment model based on the hidden Markov framework (Vogel et al., 1996). Our approach is semi-Markov: each state can generate multiple observations, representing word-to-phrase alignments. We also augment the state space to include contiguous sequences. This corresponds to phrase-to-word and phrase-to-phrase alignments. We generalize alignment by agreement (Liang et al., 2006) to this space, and find that agreement discourages EM from overfitting. Finally, we make the alignment space more symmetric by including *gappy* (or non-contiguous) phrases. This allows agreement to reinforce non-contiguous align-

*Author was a summer intern at Microsoft Research during this project.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1308–1317,
Portland, Oregon, June 19-24, 2011.

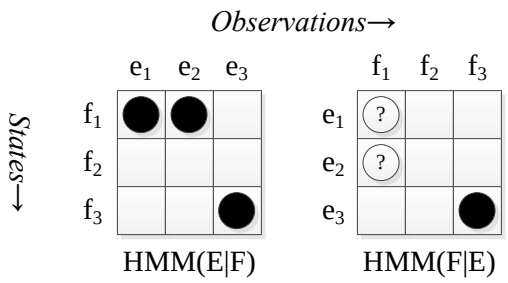


Figure 2: The model of E given F can represent the phrasal alignment $\{e_1, e_2\} \sim \{f_1\}$. However, the model of F given E cannot: the probability mass is distributed between $\{e_1\} \sim \{f_1\}$ and $\{e_2\} \sim \{f_1\}$. Agreement of the forward and backward HMM alignments tends to place less mass on phrasal links and greater mass on word-to-word links.

ments, such English *not* to French *ne ⋆ pas*. Pruning the set of allowed phrases preserves the time complexity of the word-to-word HMM alignment model.

1.1 Related Work

Our first major influence is that of *conditional phrase-based models*. An early approach by Deng and Byrne (2005) changed the parameterization of the traditional word-based HMM model, modeling subsequent words from the same state using a bigram model. However, this model changes only the parameterization and not the set of possible alignments. More closely related are the approaches of Daumé III and Marcu (2004) and DeNero et al. (2006), which allow phrase-to-phrase alignments between the source and target domain. As DeNero warns, though, an unconstrained model may overfit using unusual segmentations. Interestingly, the phrase-based hidden semi-Markov model of Andrés-Ferrer and Juan (2009) does not seem to encounter these problems. We suspect two main causes: first, the model interpolates with Model 1 (Brown et al., 1994), which may help prevent overfitting, and second, the model is monotonic, which screens out many possible alignments. Monotonicity is generally undesirable, though: almost all parallel sentences exhibit some reordering phenomena, even when languages are syntactically very similar.

The second major inspiration is *alignment by agreement* by Liang et al. (2006). Here, soft intersection between the forward (F→E) and backward (E→F) alignments during parameter estimation produces better word-to-word correspondences. This unsupervised approach produced alignments with incredibly low error rates on French-English, though only moderate gains in end-to-end machine translation results. Likely this is because the symmetric portion of the HMM space contains only single word to single word links. As shown in Figure 2, in order to retain the phrasal link $f_1 \sim e_1, e_2$ after agreement, we need the reverse phrasal link $e_1, e_2 \backsim f_1$ in the backward direction. However, this is not possible in a word-based HMM where each observation must be generated by a single state. Agreement tends to encourage 1-to-1 alignments with very high precision and but lower recall. As each word alignment acts as a constraint on phrase extraction, the phrase-pairs obtained from those alignments have high recall and low precision.

2 Gappy Phrasal Alignment

Our goal is to unify phrasal alignment and alignment by agreement. We use a phrasal hidden semi-Markov alignment model, but without the monotonicity requirement of Andrés-Ferrer and Juan (2009). Since phrases may be used in both the state and observation space of both sentences, agreement during EM training no longer penalizes phrasal links such as those in Figure 2. Moreover, the benefits of agreement are preserved: meaningful phrasal links that are likely in both directions of alignment will be reinforced, while phrasal links likely in only one direction will be discouraged. This avoids segmentation problems encountered by DeNero et al. (2006).

Non-contiguous sequences of words present an additional challenge. Even a semi-Markov model with phrases can represent the alignment between English *not* and French *ne ⋆ pas* in one direction only. To make the model more symmetric, we extend the state space to include *gappy phrases* as well.[1] The set of alignments in each model becomes symmetric, though the two directions model gappy phrases differently. Consider *not* and *ne ⋆ pas*: when predicting French given English, the alignment corresponds to generating multiple distinct ob-

[1] We only allow a single gap with one word on each end. This is sufficient for the vast majority of the gapped phenomena that we have seen in our training data.

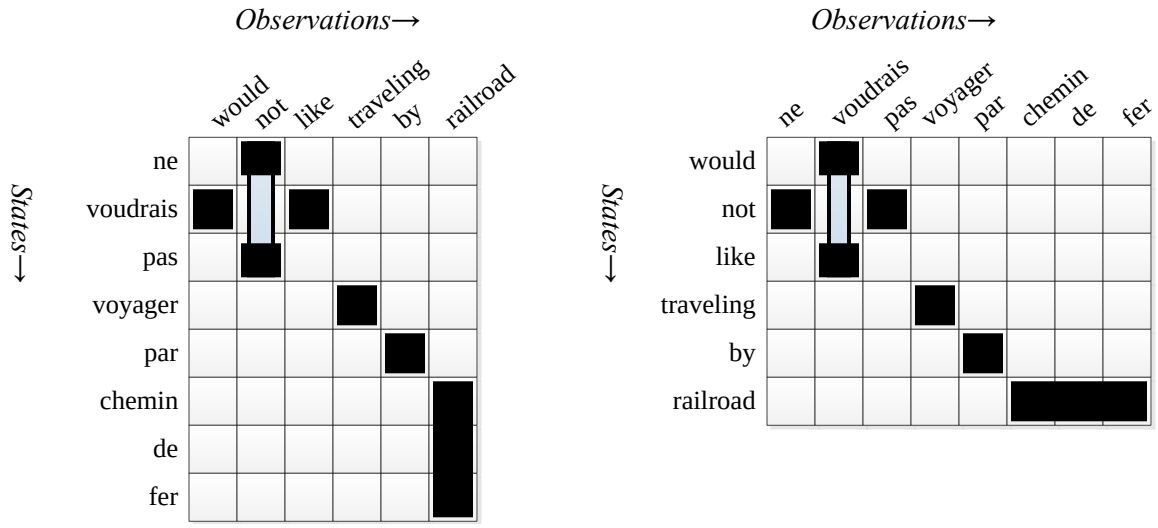


Figure 3: Example English-given-French and French-given-English alignments of the same sentence pair using the Hidden Semi-Markov Model (HSMM) for gapped-phrase-to-phrase alignment. It allows the state side phrases (denoted by vertical blocks), observation side phrases (denoted by horizontal blocks), and state-side gaps (denoted by discontinuous blocks in the same column connected by a hollow vertical "bridge"). Note both directions can capture the desired alignment for this sentence pair.

servations from the same state; in the other direction, the word *not* is generated by a single gappy phrase *ne* ⋆ *pas*. Computing posteriors for agreement is somewhat complicated, so we resort to an approximation described later. Exact inference retains a low-order polynomial runtime; we use pruning to increase speed.

2.1 Hidden Markov Alignment Models

Our model can be seen as an extension of the standard word-based Hidden Markov Model (HMM) used in alignment (Vogel et al., 1996). To ground the discussion, we first review the structure of that model. This generative model has the form $p(O|S) = \sum_A p(A, O|S)$, where $S = (s_1, \ldots, s_I) \in \Sigma^\star$ is a sequence of words from a vocabulary Σ; $O = (o_1, \ldots, o_J) \in \Pi^\star$ is a sequence from vocabulary Π; and $A = (a_1, \ldots, a_J)$ is the alignment between the two sequences. Since some words are systematically inserted during translation, the target (state) word sequence is augmented with a special NULL word. To retain the position of the last aligned word, the state space contains I copies of the NULL word, one for each position (Och and Ney, 2003). The alignment uses positive positions for words and negative positions for NULL states, so $a_j \in \{1..I\} \cup \{-1.. - I\}$, and $s_i = \text{NULL}$ if $i < 0$.

It uses the following generative procedure. First the length of the observation sequence is selected based on $p_l(J|I)$. Then for each observation position, the state is selected based on the prior state: a null state with probability p_0, or a non-null state at position a_j with probability $(1 - p_0) \cdot p_j(a_j|a_{j-1})$ where p_j is a jump distribution. Finally the observation word o_j at that position is generated with probability $p_t(o_j|s_{a_j})$, where p_t is an emission distribution:

$$p(A, O|S) = p_l(J|I) \prod_{j=1}^{J} p_j(a_j|a_{j-1}) p_t(o_j|s_{a_j})$$

$$p_j(a|a') = \begin{cases} (1 - p_0) \cdot p_d(a - |a'|) & a > 0 \\ p_0 \cdot \delta(|a|, |a'|) & a < 0 \end{cases}$$

We pick p_0 using grid search on the development set, p_l is uniform, and the p_j and p_t are optimized by EM.[2]

2.2 Gappy Semi-Markov Models

The HMM alignment model identifies a word-to-word correspondence between the observation

[2]Note that jump distances beyond -10 or 10 share a single parameter to prevent sparsity.

words and the state words. We make two changes to expand this model. First, we allow contiguous phrases on the observation side, which makes the model semi-Markov: at each time stamp, the model may emit more than one observation word. Next, we also allow contiguous and gappy phrases on the state side, leading to an alignment model that can retain phrasal links after agreement (see Section 4).

The S and O random variables are unchanged. Since a single state may generate multiple observation words, we add a new variable K representing the number of states. K should be less than J, the number of observations. The alignment variable is augmented to allow contiguous and non-contiguous ranges of words. We allow only a single gap, but of unlimited length. The null state is still present, and is again represented by negative numbers.

$$\begin{aligned} A =& (a_1, \ldots, a_K) \in \mathcal{A}(I) \\ \mathcal{A}(I) =& \{(i_1, i_2, g) | 0 < i_1 \leq i_2 \leq I, \\ & g \in \{\text{GAP}, \text{CONTIG}\}\} \cup \\ & \{(-i, -i, \text{CONTIG}) \mid 0 < i \leq I\} \end{aligned}$$

We add one more random variable to capture the total number of observations generated by each state.

$$L \in \{(l_0, l_1, \ldots, l_K) \mid 0 = l_0 < \cdots < l_K = J\}$$

The generative model takes the following form:

$$\begin{aligned} p(A, L, O|S) =& p_l(J|I) p_f(K|J) \prod_{k=1}^{K} p_j(a_k|a_{k-1}) \cdot \\ & p_t(l_k, o_{l_{k-1}+1}^{l_k} | S[a_k], l_{k-1}) \end{aligned}$$

First, the length of the observation sequence (J) is selected, based on the number of words in the state-side sentence (I). Since it does not affect the alignment, p_l is modeled as a uniform distribution. Next, we pick the total number of states to use (K), which must be less than the number of observations (J). Short state sequences receive an exponential penalty: $p_f(K|J) \propto \eta^{(J-K)}$ if $0 \leq K \leq J$, or 0 otherwise. A harsh penalty (small positive value of η) may prevent the systematic overuse of phrases.[3]

[3]We found that this penalty was crucial to prevent overfitting in independent training. Joint training with agreement made it basically unnecessary.

Next we decide the assignment of each state. We retain the first-order Markov assumption: the selection of each state is conditioned only on the prior state. The transition distribution is identical to the word-based HMM for single word states. For phrasal and gappy states, we jump into the first word of that state, and out of the last word of that state, and then pay a cost according to how many words are covered within that state. If $a = (i_1, i_2, g)$, then the beginning word of a is $F(a) = i_1$, the ending word is $L(a) = i_2$, and the length $N(a)$ is 2 for gapped states, 0 for null states, and $last(a) - first(a) + 1$ for all others. The transition probability is:

$$p_j(a|a') = \begin{cases} p_0 \cdot \delta(|F(a)|, |L(a')|) & \text{if } F(a) < 0 \\ (1 - p_0) p_d(F(a) - |L(a')|) \cdot & \\ \quad p_n(N(a)) & \text{otherwise} \end{cases}$$

where $p_n(c) \propto \kappa^c$ is an exponential distribution. As in the word HMM case, we use a mixture parameter p_0 to determine the likelihood of landing in a NULL state. The position of that NULL state remembers the last position of the prior state. For non-null words, we pick the first word of the state according to the distance from the last word of the prior state. Finally, we pick a length for that final state according to an exponential distribution: values of κ less than one will penalize the use of phrasal states.

For each set of state words, we maintain an emission distribution over observation word sequences. Let $S[a]$ be the set of state words referred to by the alignment variable a. For example, the English given French alignment of Figure 3 includes the following state word sets:

$$\begin{aligned} S[(2, 2, \text{CONTIG})] &= \textit{voudrais} \\ S[(1, 3, \text{GAP})] &= \textit{ne} \star \textit{pas} \\ S[(6, 8, \text{CONTIG})] &= \textit{chemin de fer} \end{aligned}$$

For the emission distribution we keep a multinomial over observation phrases for each set of state words:

$$p(l, o_{l'}^{l} | S[a], l') \propto c(o_{l'}^{l} | S[a])$$

In contrast to the approach of Deng and Byrne (2005), this encourages greater consistency across instances, and more closely resembles the commonly used phrasal translation models.

We note in passing that $p_f(K|J)$ may be moved inside the product: $p_f(K|J) \propto \eta^{(J-K)} = \prod_{k=1}^{K} \eta^{(l_k - l_{k-1} - 1)}$. The following form derived using the above rearrangement is helpful during EM.

$$p(A, L, O|S) \propto \prod_{k=1}^{K} p_j(a_k|a_{k-1}) \cdot p_t(l_k, o_{l_{k-1}+1}^{l_k}|S[a_k], l_{k-1}) \cdot \eta^{(l_k - l_{k-1} - 1)}$$

where $l_k - l_{k-1} - 1$ is the length of the observation phrase emitted by state $S[a_k]$.

2.3 Minimality

At alignment time we focus on finding the minimal phrase pairs, under the assumption that composed phrase pairs can be extracted in terms of these minimal pairs. We are rather strict about this, allowing only $1 \rightarrow k$ and $k \rightarrow 1$ phrasal alignment edges (or links). This should not cause undue stress, since edges of the form $2-3$ (say $e_1e_2 \sim f_1f_2f_3$) can generally be decomposed into $1-1 \cup 1-2$ (i.e., $e_1 \sim f_1 \cup e_2 \sim f_2f_3$), etc. However, the model does not require this to be true: we will describe re-estimation for unconstrained general models, but use the limited form for word alignment.

3 Parameter Estimation

We use Expectation-Maximization (EM) to estimate parameters. The forward-backward algorithm efficiently computes posteriors of transitions and emissions in the word-based HMM. In a standard HMM, emission always advances the observation position by one, and the next transition is unaffected by the emission. Neither of these assumptions hold in our model: multiple observations may be emitted at a time, and a state may cover multiple state-side words, which affects the outgoing transition. A modified dynamic program computes posteriors for this generalized model.

The following formulation of the forward-backward algorithm for word-to-word alignment is a good starting point. $\alpha[x, 0, y]$ indicates the total mass of paths that have just transitioned into state y at observation x but have not yet emitted; $\alpha[x, 1, y]$ represents the mass after emission but before subsequent transition. β is defined similarly. (We omit NULL states for brevity; the extension is straightforward.)

$$\begin{aligned}
\alpha[0, 0, y] &= p_j(y|\text{INIT}) \\
\alpha[x, 1, y] &= \alpha[x, 0, y] \cdot p_t(o_x|s_y) \\
\alpha[x, 0, y] &= \sum_{y'} \alpha[x-1, 1, y'] \cdot p_j(y|y') \\
\beta[n, 1, y] &= 1 \\
\beta[x, 0, y] &= p_t(o_x|s_y) \cdot \beta[x, 1, y] \\
\beta[x, 1, y] &= \sum_{y'} p_j(y'|y) \cdot \beta[x+1, 0, y']
\end{aligned}$$

Not only is it easy to compute posteriors of both emissions ($\alpha[x, 0, y]p_t(o_x|s_y)\beta[x, 1, y]$) and transitions ($\alpha[x, 1, y]p_j(y'|y)\beta[x+1, 0, y']$) with this formulation, it also simplifies the generalization to complex emissions. We update the emission forward probabilities to include a search over the possible starting points in the state and observation space:

$$\begin{aligned}
\alpha[0, 0, y] &= p_j(y|\text{INIT}) \\
\alpha[x, 1, y] &= \sum_{x' < x, y' \leq y} \alpha[x', 0, y'] \cdot \text{EMIT}(x' : x, y' : y) \\
\alpha[x, 0, y] &= \sum_{y'} \alpha[x-1, 1, y'] \cdot p_j(y|y')
\end{aligned}$$

$$\begin{aligned}
\beta[n, 1, y] &= 1 \\
\beta[x', 0, y'] &= \sum_{x' < x, y' \leq y} \text{EMIT}(x' : x, y' : y) \cdot \beta[x, 1, y] \\
\beta[x, 1, y] &= \sum_{y'} p_j(y'|y) \cdot \beta[x+1, 0, y']
\end{aligned}$$

Phrasal and gapped emissions are pooled into EMIT:

$$\begin{aligned}
\text{EMIT}(w : x, y : z) = &p_t(o_w^x|s_y^z) \cdot \eta^{z-y+1} \cdot \kappa^{x-w+1} + \\
&p_t(o_w^x|s_y \star s_z) \cdot \eta^2 \cdot \kappa^{x-w+1}
\end{aligned}$$

The transition posterior is the same as above. The emission is very similar: the posterior probability that o_w^x is aligned to s_y^z is proportional to $\alpha[w, 0, y] \cdot p_t(o_w^x|s_y^z) \cdot \eta^{z-y+1} \cdot \kappa^{x-w+1} \cdot \beta[x, 1, z]$. For a gapped phrase, the posterior is proportional to $\alpha[w, 0, y] \cdot p_t(o_w^x|s_y \star s_z) \cdot \eta^2 \cdot \kappa^{x-w+1} \cdot \beta[x, 1, z]$.

Given an inference procedure for computing posteriors, unsupervised training with EM follows immediately. We use a simple maximum-likelihood update of the parameters using expected counts based on the posterior distribution.

4 Alignment by Agreement

Following Liang et al. (2006), we quantify agreement between two models as the probability that the alignments produced by the two models agree on the alignment $\mathbf{z}$ of a sentence pair $\mathbf{x} = (S, O)$:

$$\sum_{\mathbf{z}} p_1(\mathbf{z}|\mathbf{x};\theta_1)p_2(\mathbf{z}|\mathbf{x};\theta_2)$$

To couple the two models, the (log) probability of agreement is added to the standard log-likelihood objective:

$$\max_{\theta_1,\theta_2} \sum_{\mathbf{x}} \Big[\log p_1(\mathbf{x};\theta_1) + \log p_2(\mathbf{x};\theta_2) + \log \sum_{\mathbf{z}} p_1(\mathbf{z}|\mathbf{x};\theta_1)p_2(\mathbf{z}|\mathbf{x};\theta_2) \Big]$$

We use the heuristic estimator from Liang et al. (2006), letting q be a product of marginals:

$$\text{E} : q(\mathbf{z};\mathbf{x}) := \prod_{z \in \mathbf{z}} p_1(z|\mathbf{x};\theta_1)p_2(z|\mathbf{x};\theta_2)$$

where each $p_k(z|\mathbf{x};\theta_k)$ is the posterior marginal of some edge z according to each model. Such a heuristic E step computes the marginals for each model separately, then multiplies the marginals corresponding to the same edge. This product of marginals acts as the approximation to the posterior used in the M step for each model. The intuition is that if the two models disagree on a certain edge z, then the marginal product is small, hence that edge is dis-preferred in each model.

Contiguous phrase agreement. It is simple to extend agreement to alignments in the absence of gaps. Multi-word (phrasal) links are assigned some posterior probability in both models, as shown in the example in Figure 3, and we multiply the posteriors of these phrasal links just as in the single word case.[4]

$$\begin{aligned}\gamma_{F\to E}(f_i, e_j) &:= \gamma_{E\to F}(e_j, f_i) \\ &:= [\gamma_{F\to E}(f_i, e_j) \times \gamma_{E\to F}(e_j, f_i)]\end{aligned}$$

[4]Phrasal correspondences can be represented in multiple ways: multiple adjacent words could be generated from the same state either using one semi-Markov emission, or using multiple single word emissions followed by self-jumps. Only the first case is reinforced through agreement, so the latter is implicitly discouraged. We explored an option to forbid same-state transitions, but found it made little difference in practice.

Gappy phrase agreement. When we introduce gappy phrasal states, agreement becomes more challenging. In the forward direction F→E, if we have a gappy state aligned to an observation, say $f_i \star f_j \sim e_k$, then its corresponding edge in the backward direction E→F would be $e_k \backsim f_i \star f_j$. However, this is represented by two distinct and unrelated emissions. Although it is possible the compute the posterior probability of two non-adjacent emissions, this requires running a separate dynamic program for each such combination to sum the mass between these emissions. For the sake of efficiency we resort to an approximate computation of posterior marginals using the two word-to-word edges $e_k \backsim f_i$ and $e_k \backsim f_j$.

The forward posterior $\gamma_{F\to E}$ for edge $f_i \star f_j \sim e_k$ is multiplied with the $\min$ of the backward posteriors of the edges $e_k \backsim f_i$ and $e_k \backsim f_j$.

$$\begin{aligned}\gamma_{F\to E}(f_i \star f_j, e_k) := \gamma_{F\to E}(f_i \star f_j, e_k) \times \\ \min\Big\{\gamma_{E\to F}(e_k, f_i), \gamma_{E\to F}(e_k, f_j)\Big\}\end{aligned}$$

Note that this $\min$ is an upper bound on the desired posterior of edge $e_k \backsim f_i \star f_j$, since every path that passes through $e_k \backsim f_i$ and $e_k \backsim f_j$ must pass through $e_k \backsim f_i$, therefore the posterior of $e_k \backsim f_i \star f_j$ is less than that of $e_k \backsim f_i$, and likewise less than that of $e_k \backsim f_j$.

The backward posteriors of the edges $e_k \backsim f_i$ and $e_k \backsim f_j$ are also mixed with the forward posteriors of the edges to which they correspond.

$$\gamma_{E\to F}(e_k, f_i) := \gamma_{E\to F}(e_k, f_i) \times \Bigg[\gamma_{F\to E}(f_i, e_k) + \sum_{h<i<j} \Big\{\gamma_{F\to E}(f_h \star f_i, e_k) + \gamma_{F\to E}(f_i \star f_j, e_k)\Big\}\Bigg]$$

5 Pruned Lists of 'Allowed' Phrases

To identify contiguous and gapped phrases that are more likely to lead to good alignments, we use word-to-word HMM alignments from the full training data in both directions (F→E and E→F). We collect observation phrases of length 2 to K aligned to a single state, i.e. $o_i^j \sim s$, to add to a list of allowed phrases. For gappy phrases, we find all non-consecutive observation pairs o_i and o_j such that: (a) both are

aligned to the same state s_k, (b) state s_k is aligned to only these two observations, and (c) at least one observation between o_i and o_j is aligned to a non-null state other than s_k. These observation phrases are collected from F→E and E→F models to build contiguous and gappy phrase lists for both languages.

Next, we order the phrases in each contiguous list using the discounted probability:

$$p_\delta(o_i^j \sim s | o_i^j) = \frac{\max(0, count(o_i^j \sim s) - \delta)}{count(o_i^j)}$$

where $count(o_i^j \sim s)$ is the count of occurrence of the observation-phrase o_i^j, all aligned to some single state s, and $count(o_i^j)$ is the count of occurrence of the observation phrase o_i^j, not all necessarily aligned to a single state. Similarly, we rank the gappy phrases using the discounted probability:

$$p_\delta(o_i \star o_j \sim s | o_i \star o_j) = \frac{\max(0, count(o_i \star o_j \sim s) - \delta)}{count(o_i \star o_j)}$$

where $count(o_i \star o_j \sim s)$ is the count of occurrence of the observations o_i and o_j aligned to a single state s with the conditions mentioned above, and $count(o_i \star o_j)$ is the count of general occurrence of the observations o_i and o_j in order. We find that 200 gappy phrases and 1000 contiguous phrases works well, based on tuning with a development set.

6 Complexity Analysis

Let m be the length of the state sentence S and n be the length of the observation sentence O. In IBM Model 1 (Brown et al., 1994), with only a translation model, we can infer posteriors or max alignments in $\mathcal{O}(mn)$. HMM-based word-to-word alignment model (Vogel et al., 1996) adds a distortion model, increasing the complexity to $\mathcal{O}(m^2 n)$.

Introducing phrases (contiguous) on the observation side, we get a HSMM (Hidden Semi-Markov Model). If we allow phrases of length no greater than K, then the number of observation types rises from n to Kn for an overall complexity of $\mathcal{O}(m^2 Kn)$. Introducing state phrases (contiguous) with length $\leq K$ grows the number of state types from m to Km. Complexity further increases to $\mathcal{O}((Km)^2 Kn) = \mathcal{O}(K^3 m^2 n)$.

Finally, when we introduce gappy state phrases of the type $s_i \star s_j$, the number of such phrases is $\mathcal{O}(m^2)$, since we may choose a start and end point independently. Thus, the total complexity rises to $\mathcal{O}((Km + m^2)^2 Kn) = \mathcal{O}(Km^4 n)$. Although this is less than the $\mathcal{O}(n^6)$ complexity of exact ITG (Inversion Transduction Grammar) model (Wu, 1997), a quintic algorithm is often quite slow.

The pruned lists of allowed phrases limit this complexity. The model is allowed to use observation (contiguous) and state (contiguous and gappy) phrases only from these lists. The number of phrases that match any given sentence pair from these pruned lists is very small ($\sim$ 2 to 5). If the number of phrases in the lists that match the observation and state side of a given sentence pair are small constants, the complexity remains $\mathcal{O}(m^2 n)$, equal to that of word-based models.

7 Results

We evaluate our models based on both word alignment and end-to-end translation with two language pairs: English-French and English-German. For French-English, we use the Hansards NAACL 2003 shared-task dataset, which contains nearly 1.1 million training sentence pairs. We also evaluated on German-English Europarl data from WMT2010, with nearly 1.6 million training sentence pairs. The model from Liang et al. (2006) is our word-based baseline.

7.1 Training Regimen

Our training regimen begins with both the forward (F→E) and backward (E→F) iterations of Model 1 run independently (i.e. without agreement). Next, we train several iterations of the forward and backward word-to-word HMMs, again with independent training. We do not use agreement during word alignment since it tends to produce sparse 1-1 alignments, which in turn leads to low phrase emission probabilities in the gappy model.

Initializing the emission probabilities of the semi-Markov model is somewhat complicated, since the word-based models do not assign any mass to the phrasal or gapped configurations. Therefore we use a heuristic method. We first retrieve the Viterbi alignments of the forward and backward

word-to-word HMM aligners. For phrasal correspondences, we combine these forward and backward Viterbi alignments using a common heuristic (Union, Intersection, Refined, or Grow-Diag-Final), and extract tight phrase-pairs (no unaligned words on the boundary) from this alignment set. We found that Grow-Diag-Final was most effective in our experiments. The counts gathered from this phrase extraction are used to initialize phrasal translation probabilities. For gappy states in a forward (F→E) model, we use alignments from the backward (E→F) model. If a state s_k is aligned to two non-consecutive observations o_i and o_j such that s_k is not aligned to any other observation, and at least one observation between o_i and o_j is aligned to a non-null state other than s_k, then we reverse this link to get $o_i \ \star \ o_j \sim s_k$ and use it as a gapped-state-phrase instance for adding fractional counts. Given these approximate fractional counts, we perform a standard MLE M-step to initialize the emission probability distributions. The distortion probabilities from the word-based model are used without changes.

7.2 Alignment Results (F1)

The validation and test sentences have been hand-aligned (see Och and Ney (2003)) and are marked with both sure and possible alignments. For French-English, following Liang et al. (2006), we lowercase all words, and use the validation set plus the first 100 test sentences as our development set and the remaining 347 test-sentences as our test-set for final F1 evaluation.[5] In German-English, we have a development set of 102 sentences, and a test set of 258 sentences, also annotated with a set of sure and possible alignments. Given a predicted alignment A, precision and recall are computed using sure alignments S and possible alignments P (where $S \subseteq P$) as in Och and Ney (2003):

$$Precision = \frac{|A \cap P|}{|A|} \times 100\%$$

$$Recall = \frac{|A \cap S|}{|S|} \times 100\%$$

[5]We report F1 rather than AER because AER appears not to correlate well with translation quality.(Fraser and Marcu, 2007)

Language pair	Word-to-word	Gappy
French-English	34.0	34.5
German-English	19.3	19.8

Table 2: BLEU results on German-English and French-English.

$$AER = \left(1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}\right) \times 100\%$$

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \times 100\%$$

Many free parameters were tuned to optimize alignment F1 on the development set, including the number of iterations of each Model 1, HMM, and Gappy; the NULL weight p_0, the number of contiguous and gappy phrases to include, and the maximum phrase length. Five iterations of all models, $p_0 = 0.3$, using the top 1000 contiguous phrases and the top 200 gappy phrases, maximum phrase length of 5, and penalties $\eta = \kappa = 1$ produced competitive results. Note that by setting η and κ to one, we have effectively removed the penalty altogether without affecting our results. In Table 1 we see a consistent improvement with the addition of contiguous phrases, and some additional gains with gappy phrases.

7.3 Translation Results (BLEU)

We assembled a phrase-based system from the alignments (using only contiguous phrases consistent with the potentially gappy alignment), with 4 channel models, word and phrase count features, distortion penalty, lexicalized reordering model, and a 5-gram language model, weighted by MERT. The same free parameters from above were tuned to optimize development set BLEU using grid search. The improvements in Table 2 are encouraging, especially as a syntax-based or non-contiguous phrasal system (Galley and Manning, 2010) may benefit more from gappy phrases.

8 Conclusions and Future Work

We have described an algorithm for efficient unsupervised alignment of phrases. Relatively straightforward extensions to the base HMM allow for efficient inference, and agreement between the two

Data	Decoding method	Word-to-word	+Contig phrases	+Gappy phrases
FE 10K	Viterbi	89.7	90.6	90.3
FE 10K	Posterior ≥ 0.1	90.1	90.4	**90.7**
FE 100K	Viterbi	93.0	93.6	93.8
FE 100K	Posterior ≥ 0.1	93.1	93.7	**93.8**
FE All	Viterbi	94.1	94.3	94.3
FE All	Posterior ≥ 0.1	94.2	94.4	**94.5**
GE 10K	Viterbi	76.2	79.6	**79.7**
GE 10K	Posterior ≥ 0.1	76.7	79.3	79.3
GE 100K	Viterbi	81.0	83.0	83.2
GE 100K	Posterior ≥ 0.1	80.7	83.1	**83.4**
GE All	Viterbi	83.0	85.2	85.6
GE All	Posterior ≥ 0.1	83.7	85.3	**85.7**

Table 1: F1 scores of automatic word alignments, evaluated on the test set of the hand-aligned sentence pairs.

models prevents EM from overfitting, even in the absence of harsh penalties. We also allow gappy (non-contiguous) phrases on the state side, which makes agreement more successful but agreement needs approximation of posterior marginals. Using pruned lists of good phrases, we maintain complexity equal to the baseline word-to-word model.

There are several steps forward from this point. Limiting the gap length also prevents combinatorial explosion; we hope to explore this in future work. Clearly a translation system that uses discontinuous mappings at runtime (Chiang, 2007; Galley and Manning, 2010) may make better use of discontinuous alignments. This model can also be applied at the morpheme or character level, allowing joint inference of segmentation and alignment. Furthermore the state space could be expanded and enhanced to include more possibilities: states with multiple gaps might be useful for alignment in languages with template morphology, such as Arabic or Hebrew. More exploration in the model space could be useful – a better distortion model might place a stronger distribution on the likely starting and ending points of phrases.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful suggestions. This project is funded by Microsoft Research.

References

Jesús Andrés-Ferrer and Alfons Juan. 2009. A phrase-based hidden semi-Markov approach to machine translation. In *Proceedings of EAMT*.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1994. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*.

Hal Daumé III and Daniel Marcu. 2004. A phrase-based HMM approach to document/abstract alignment. In *Proceedings of EMNLP*.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. 2006. Why generative phrase models underperform surface heuristics. In *Proceedings of ACL*.

Yonggang Deng and William Byrne. 2005. HMM word and phrase alignment for statistical machine translation. In *Proceedings of HLT-EMNLP*.

Xiangyu Duan, Min Zhang, and Haizhou Li. 2010. Pseudo-word for phrase-based machine translation. In *Proceedings of ACL*.

Alexander Fraser and Daniel Marcu. 2007. Measuring word alignment quality for statistical machine translation. *Computational Linguistics*, 33(3):293–303.

Michel Galley and Christopher D. Manning. 2010. Accurate non-hierarchical phrase-based translation. In *HLT/NAACL*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT-NAACL*.

Philipp Koehn, Franz Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of HLT-NAACL*.

Patrik Lambert and Rafael Banchs. 2006. Grouping multi-word expressions according to part-of-speech in

statistical machine translation. In *Proc. of the EACL Workshop on Multi-Word-Expressions in a Multilingual Context*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of HLT-NAACL*.

Yanjun Ma, Nicolas Stroppa, and Andy Way. 2007. Boostrapping word alignment via word packing. In *Proceedings of ACL*.

Daniel Marcu and Daniel Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29:19–51.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of COLING*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23:377–404.

Andreas Zollmann, Ashish Venugopal, and Stephan Vogel. 2006. Syntax augmented machine translation via chart parsing. In *Processings of the Statistical Machine Translation Workshop at NAACL*.

Translationese and Its Dialects

Moshe Koppel
Department of Computer Science
Bar Ilan University
Ramat-Gan, Israel 52900
moishk@gmail.com

Noam Ordan
Department of Computer Science
University of Haifa
Haifa, Israel 31905
noam.ordan@gmail.com

Abstract

While it is has often been observed that the product of translation is somehow different than non-translated text, scholars have emphasized two distinct bases for such differences. Some have noted interference from the source language spilling over into translation in a source-language-specific way, while others have noted general effects of the process of translation that are independent of source language. Using a series of text categorization experiments, we show that both these effects exist and that, moreover, there is a continuum between them. There are many effects of translation that are consistent among texts translated from a given source language, some of which are consistent even among texts translated from families of source languages. Significantly, we find that even for widely unrelated source languages and multiple genres, differences between translated texts and non-translated texts are sufficient for a learned classifier to accurately determine if a given text is translated or original.

1 Introduction

The products of translation (written or oral) are generally assumed to be ontologically different from non-translated texts. Researchers have emphasized two aspects of this difference. Some (Baker 1993) have emphasized general effects of the process of translation that are independent of source language and regard the collective product of this process in a given target language as an 'interlanguage' (Selinker, 1972), 'third code' (Frawley, 1984) or 'translationese' (Gellerstam, 1986). Others (Toury, 1995) have emphasized the effects of *interference*, the process by which a specific source language leaves distinct marks or fingerprints in the target language, so that translations from different source languages into the same target language may be regarded as distinct dialects of translationese.

We wish to use text categorization methods to set both of these claims on a firm empirical foundation. We will begin by bringing evidence for two claims:

(1) Translations from different source languages into the same target language are sufficiently different from each other for a learned classifier to accurately identify the source language of a given translated text;

(2) Translations from a mix of source languages are sufficiently distinct from texts originally written in the target language for a learned classifier to accurately determine if a given text is translated or original.

Each of these claims has been made before, but our results will strengthen them in a number of ways. Furthermore, we will show that the degree of difference between translations from two source languages reflects the degree of difference between the source languages themselves. Translations from cognate languages differ from non-translated texts in similar ways, while translations from unrelated languages differ from non-translated texts in distinct ways. The same result holds for families of languages.

The outline of the paper is as follows. In the following section, we show that translations from different source languages can be distinguished from each other and that closely related source languages manifest similar forms of interference. In section 3, we show that, in a corpus involving five European languages, we can distinguish translationese from non-translated text and we consider some salient markers of translationese. In section

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1318–1326,
Portland, Oregon, June 19-24, 2011.

4, we consider the extent to which markers of translationese cross over into non-European languages as well as into different genres. Finally, we consider possible applications and implications for future studies.

2 Interference Effects in Translationese

In this section, we perform several text categorization experiments designed to show the extent to which interference affects (both positively and negatively) our ability to classify documents.

2.1 The Europarl Corpus

The main corpus we will use throughout this paper is Europarl (Koehn, 2005), which consists of transcripts of addresses given in the European Parliament. The full corpus consists of texts translated into English from 11 different languages (and vice versa), as well as texts originally produced in English. For our purposes, it will be sufficient to use translations from five languages (Finnish, French, German, Italian and Spanish), as well as original English. We note that this corpus constitutes a *comparable corpus* (Laviosa, 1997), since it contains (1) texts written originally in a certain language (English), as well as (2) texts translated into that same language, matched for genre, domain, publication timeframe, etc. Each of the five translated components is a text file containing just under 500,000 words; the original English component is a file of the same size as the aggregate of the other five.

The five source languages we use were selected by first eliminating several source languages for which the available text was limited and then choosing from among the remaining languages, those of varying degrees of pairwise similarity. Thus, we select three cognate (Romance) languages (French, Italian and Spanish), a fourth less related language (German), and a fifth even further removed (Finnish). As will become clear, the motivation is to see whether the distance between the languages impacts the distinctiveness of the translation product.

We divide each of the translated corpora into 250 equal chunks, paying no attention to natural units within the corpus. Similarly, we divide the original English corpus into 1250 equal chunks. We set aside 50 chunks from each of the translated corpora and 250 chunks from the original English corpus for development purposes (as will be explained below). The experiments described below use the remaining 1000 translated chunks and 1000 original English chunks.

2.2 Identifying source language

Our objective in this section is to measure the extent to which translations are affected by source language. Our first experiment will be to use text categorization methods to learn a classifier that categorizes translations according to source language. We will check the accuracy of such classifiers on out-of-sample texts. High accuracy would reflect that there are exploitable differences among translations of otherwise comparable texts that differ only in terms of source language.

The details of the experiment are as follows. We use the 200 chunks from each translated corpus, as described above. We use as our feature set a list of 300 function words taken from LIWC (Pennebaker, 2001) and represent each chunk as a vector of size 300 in which each entry represents the frequency of the corresponding feature in the chunk. The restriction to function words is crucial; we wish to rely only on stylistic differences rather than content differences that might be artifacts of the corpus.

We use Bayesian logistic regression (Madigan, 2005) as our learning method in order to learn a classifier that classifies a given text into one of five classes representing the different source languages. We use 10-fold cross-validation as our testing method.

We find that 92.7% of documents are correctly classified.

In Table 1 we show the confusion matrix for the five languages. As can be seen, there are more mistakes across the three cognate languages than between those three languages and German and still fewer mistakes involving the more distant Finnish language.

	It	Fr	Es	De	Fi
It	169	19	8	4	0
Fr	18	161	12	8	1
Es	3	11	172	11	3
De	4	12	3	178	3
Fi	0	1	2	5	192

Table 1: Confusion matrix for 10-fold cross validation experiment to determine source language of texts translated into English

This result strengthens that of van Halteren (2008) in a similar experiment. Van Halteren, also using Europarl (but with Dutch as the fifth source language, rather than Finnish), obtained accuracy of 87.2%-96.7% for a two-way decision on source language, and 81.5%-87.4% for a six-way decision (including the original which has no source language). Significantly, though, van Halteren's feature set included content words and he notes that many of the most salient differences reflected differences in thematic emphasis. By restricting our feature set to function words, we neutralize such effects.

In Table 2, we show the two words most over-represented and the two words most under-represented in translations from each source language (ranked according to an unpaired T-test). For each of these, the difference between frequency of use in the indicated language and frequency of use in the other languages in aggregate is significant at $p<0.01$.

	over-represented	**under-represented**
Fr	*of, finally*	*here, also*
It	*upon, moreover*	*also, here*
Es	*with, therefore*	*too, then*
De	*here, then*	*of, moreover*
Fi	*be, example*	*me, which*

Table 2: Most salient markers of translations from each source language.

The two most underrepresented words for French and Italian, respectively, are in fact identical. Furthermore, the word *too* which is underrepresented for Spanish is a near synonym of *also* which appears in both French and Spanish. This suggests the possibility that interference effects in cognate languages such as French, Italian and Spanish might be similar. We will see presently that this is in fact the case.

When a less related language is involved we see the opposite picture. For German, both underrepresented items appear as overrepresented in the Romance languages, and, conversely, underrepresented items in the Romance languages appear as overrepresented items for German. This may cast doubt on the idea that all translations share universal properties and that at best we may claim that particular properties are shared by closely related languages but not others. In the experiments presented in the next subsection, we'll find that translationese is gradable: closely related languages share more features, yet even further removed languages share enough properties to hold the general translationese hypothesis as valid.

2.3 Identifying translationese per source language

We now wish to measure in a subtler manner the extent to which interference affects translation. In this experiment, the challenge is to learn a classifier that classifies a text as belonging to one of only two classes: original English (O) or translated-into-English (T). The catch is that all our training texts for the class T will be translations from some fixed source language, while all our test documents in T will be translations from a different source language. What accuracy can be achieved in such an experiment? The answer to this question will tell us a great deal about how much of translationese is general and how much of it is language dependent. If accuracy is close to 100%, translationese is purely general (Baker, 1993). (We already know from the previous experiment that that's not the case.). If accuracy is near 50%, there are no general effects, just language-dependent ones. Note that, whereas in our first experiment above pair-specific interference facilitated good classification, in this experiment pair-specific interference is an impediment to good classification.

The details of the experiment are as follows. We create, for example, a "French" corpus consisting of the 200 chunks of text translated from French and 200 original English texts. We similarly create a corpus for each of the other source languages, taking care that each of the 1000 original English texts appears in exactly one of the corpora. As above, we represent each chunk in terms of frequencies of function words. Now, using Bayesian logistic regression, we learn a classifier that distinguishes T from O in the French corpus. We then apply this learned classifier to the texts in, for example, the equivalent "Italian" corpus to see if we can classify them as translated or original. We repeat this for each of the 25 ⟨train_corpus, test_corpus⟩ pairs.

In Table 3, we show the accuracy obtained for each such pair. (For the case where the training corpus and testing corpus are identical – the di-

agonal of the matrix – we show results for ten-fold cross-validation.)

We note several interesting facts. First, results of cross-validation within each corpus are very strong. For any given source language, it is quite easy to distinguish translations from original English. This corroborates results obtained by Baroni and Bernardini (2006), Ilisei et al. (2010), Kurokawa et al. (2009) and van Halteren (2008), which we will discuss below.

We note further, that for the cases where we train on one source language and test on another, results are far worse. This clearly indicates that interference effects from one source language might be misleading when used to identify translations from a different language. Thus, for example, in the Finnish corpus, the word *me* is a strong indicator of original English (constituting 0.0003 of tokens in texts translated from Finnish as opposed to 0.0015 of tokens in original English texts), but in the German corpus, *me* is an indicator of translated text (constituting 0.0020 of tokens in text translated from German).

The most interesting result that can be seen in this table is that the accuracy obtained when training using language x and testing using language y depends precisely on the degree of similarity between x and y. Thus, for training and testing within the three cognate languages, results are fairly strong, ranging between 84.5% and 91.5%. For training/testing on German and testing/training on one of the other European languages, results are worse, ranging from 68.5% to 83.3%. Finally, for training/testing on Finnish and testing/training on any of the European languages, results are still worse, hovering near 60% (with the single unexplained outlier for training on German and testing on Finnish).

Finally, we note that even in the case of training or testing on Finnish, results are considerably better than random, suggesting that despite the confounding effects of interference, some general properties of translationese are being picked up in each case. We explore these in the following section.

3 General Properties of Translationese

Having established that there are source-language-dependent effects on translations, let's now consider source-language-independent effects on translation.

Train

	It	Fr	Es	De	Fi
It	98.3	91.5	86.5	71.3	61.5
Fr	91	97	86.5	68.5	60.8
Es	84.5	88.3	95.8	76.3	59.5
De	82	83.3	78.5	95	80.8
Fi	56	60.3	56	62.3	97.3

Table 3: Results of learning a T vs. O classifier using one source language and testing it using another source language

3.1 Identifying translationese

In order to identify general effects on translation, we now consider the same two-class classification problem as above, distinguishing T from O, except that now the translated texts in both our train and test data will be drawn from multiple source languages. If we succeed at this task, it must be because of features of translationese that cross source-languages.

The details of our experiment are as follows. We use as our translated corpus, the 1000 translated chunks (200 from each of five source languages) and as our original English corpus all 1000 original English chunks. As above, we represent each chunk in terms of function words frequencies. We use Bayesian logistic regression to learn a two-class classifier and test its accuracy using ten-fold cross-validation.

Remarkably, we obtain accuracy of 96.7%.

This result extends and strengthens results reported in some earlier studies. Ilisei et al. (2010), Kurokawa (2009) and van Halteren (2008) each obtained above 90% accuracy in distinguishing translation from original. However, in each case the translations were from a single source language. (Van Halteren considered multiple source languages, but each learned classifier used only one of them.) Thus, those results do not prove that translationese has distinctive source-language-independent features. To our knowledge, the only earlier work that used a learned classifier to identify translations in which both test and train sets involved multiple source languages is Baroni and Bernardini (2006), in which the target language was Italian and the source languages were known to be varied. The actual distribution of source languages was, however, not known to the researchers. They obtained accuracy of 86.7%. Their result was obtained using combinations of lexical and syntactic features.

3.2 Some distinguishing features

Let us now consider some of the most salient function words for which frequency of usage in T differs significantly from that in O. While there are many such features, we focus on two categories of words that are most prominent among those with the most significant differences.

First, we consider animate pronouns. In Table 4, we show the frequencies of animate pronouns in O and T, respectively (the possessive pronouns, *mine*, *yours* and *hers*, not shown, are extremely rare in the corpus). As can be seen, all pronouns are under-represented in T; for most (bolded), the difference is significant at $p<0.01$.

By contrast, the word *the* is significantly overrepresented in T (15.32% in T vs. 13.73% in O; significant at $p<0.01$).

word	freq O	freq T
I	**2.552%**	2.148%
we	**2.713%**	2.344%
you	0.479%	0.470%
he	**0.286%**	0.115%
she	**0.081%**	0.039%
me	0.148%	0.141%
us	**0.415%**	0.320%
him	**0.066%**	0.033%
her	**0.091%**	0.056%
my	**0.462%**	0.345%
our	**0.696%**	0.632%
your	0.119%	0.109%
his	**0.218%**	0.123%

Table 4: Frequency of pronouns in O and T in the Europarl corpus. Bold indicates significance at $p<0.01$.

In Table 5, we consider cohesive markers, tagged as adverbs (Schmid, 2004). (These are adverbs that can appear at the beginning of a sentence followed immediately by a comma.)

word	freq O	freq T
therefore	0.153%	**0.287%**
thus	0.015%	**0.041%**
consequently	0.006%	**0.014%**
hence	0.007%	**0.013%**
accordingly	0.006%	**0.011%**
however	0.216%	**0.241%**
nevertheless	0.019%	**0.045%**
also	0.460%	**0.657%**
furthermore	0.012%	**0.048%**
moreover	0.008%	**0.036%**
indeed	**0.098%**	0.053%
actually	**0.065%**	0.042%

Table 5: Frequency of cohesive adverbs in O and T in the Europarl corpus. Bold indicates significance at $p<0.01$.

We note that the preponderance of such cohesive markers are significantly more frequent in translations. In fact, we also find that a variety of phrases that serve the same purpose as cohesive adverbs, such as *in fact* and *as a result* are significantly more frequent in translationese.

The general principle underlying these phenomena is subject to speculation. Previous researchers have noted the phenomenon of *explicitation*, according to which translators tend to render implicit utterances in the source text into explicit utterances in the target text (Blum-Kulka, 1986, Laviosa-Braithwaite, 1998), for example by filling out elliptical expressions or adding connectives to increase cohesion of the text (Laviosa-Braithwaite, 1998). It is plausible that the use of cohesive adverbs is an instantiation of this phenomenon.

With regard to the under-representation of pronouns and the over-representation of *the*, there are a number of possible interpretations. It may be that this too is the result of explicitation, in which anaphora is resolved by replacing pronouns with noun phrases (e.g., *the man* instead of *he*). But it also might be that this is an example of *simplification* (Laviosa- Braithwaite 1998, Laviosa 2002), according to which the translator simplifies the message, the language, or both. Related results confirming the simplification hypothesis were found by Ilisei et al. (2010) on Spanish texts. In particular, they found that type-to-token ratio (*lexical variety*/*richness*), mean sentence length and proportion of grammatical words (*lexical density*/*readability*) are all smaller in translated texts.

We note that Van Halteren (2008) and Kurokawa et al. (2009), who considered lexical features, found cultural differences, like over-representation of *ladies* and *gentlemen* in translated speeches. Such differences, while of general interest, are orthogonal to our purposes in this paper.

3.3 Overriding language-specific effects

We found in Section 2.3 that when we trained in one language and tested in another, classification succeeded to the extent that the source languages used in training and testing, respectively, are related to each other. In effect, general differences between translationese and original English were partially overwhelmed by language-specific differences that held for the training language but not the test language. We thus now revisit that earlier experiment, but restrict ourselves to features that distinguish translationese from original English generally.

To do this, we use the small development corpus described in Section 2.1. We use Bayesian logistic regression to learn a classifier to distinguish between translationese and original English. We select the 10 highest-weighted function-word markers for T and the 10 highest-weighted function-word markers for O in the development corpus. We then rerun our train-on-source-language-x, test-on-source-language-y experiment using this restricted set as our feature set. We now find that even in the difficult case where we train on Finnish and test on another language (or vice versa), we succeed at distinguishing translationese from original English with accuracy above 80%. This considerably improves the earlier results shown in Table 3. Thus, a bit of feature engineering facilitates learning a good classifier for T vs. O even across source languages.

4 Other Genres and Language Families

We have found both general and language-specific differences between translationese and original English in one large corpus. It might be wondered whether the phenomena we have found hold in other genres and for a completely different set of source languages. To test this, we consider a second corpus.

4.1 The IHT corpus

Our second corpus includes three translated corpora, each of which is an on-line local supplement to the International Herald Tribune (IHT): *Kathimerini* (translated from Greek), *Ha'aretz* (translated from Hebrew), and the *JoongAng Daily* (translated from Korean). In addition, the corpus includes original English articles from the IHT. Each of the four components contains four different domains balanced roughly equally: news (80,000 words), arts and leisure (50,000), business and finance (50,000), and opinion (50,000) and each covers the period from April-September 2004. Each component consists of about 230,000 tokens. (Unlike for our Europarl corpus, the amount of English text available is not equal to the aggregate of the translated corpora, but rather equal to each of the individual corpora.)

It should be noted that the IHT corpus belongs to the writing modality while the Europarl corpus belongs to the speaking modality (although possibly post-edited). Furthermore, the source languages (Hebrew, Greek and Korean) in the IHT corpus are more disparate than those in the Europarl corpus.

Our first objective is to confirm that the results we obtained earlier on the Europarl corpus hold for the IHT corpus as well.

Perhaps more interestingly, our second objective is to see if the gradability phenomenon observed earlier (Table 3) generalizes to families of languages. Our first hypothesis is that a classifier for identifying translationese that is trained on Europarl will succeed only weakly to identify translationese in IHT. But our second hypothesis is that there are sufficient general properties of translationese that cross language families and genres that a learned classifier can accurately identify translationese even on a test corpus that includes both corpora, spanning eight disparate languages across two distinct genres.

4.2 Results on IHT corpus

Running essentially the same experiments as described for the Europarl corpus, we obtain the following results.

First of all, we can determine source language with accuracy of 86.5%. This is a somewhat weaker result than the 92.7% result obtained on Europarl, especially considering that there are only three classes instead of five. The difference is most likely due to the fact that the IHT corpus is about half the size of the Europarl corpus. Nevertheless, it is clear that source language strongly affects translationese in this corpus.

Second, as can be seen in Table 6, we find that the gradability phenomenon occurs in this corpus as well. Results are strongest when the train and

test corpora involve the same source language and trials involving Korean, the most distant language, are somewhat weaker than those across Greek and Hebrew.

Train	Gr	He	Ko
Gr	89.8	73.4	64.8
He	82.0	86.3	65.5
Ko	73.0	72.5	85.0

Table 6: Results of learning a T vs. O classifier using one source language and testing it using another source language

Third, we find in ten-fold cross-validation experiments that we can distinguish translationese from original English in the IHT corpus with accuracy of 86.3%. Thus, despite the great distance between the three source languages in this corpus, general differences between translationese and original English are sufficient to facilitate reasonably accurate identification of translationese.

4.3 Combining the corpora

First, we consider whether a classifier learned on the Europarl corpus can be used to identify translationese in the IHT corpus, and vice versa. It would be consistent with our findings in Section 2.3, that we would achieve better than random results but not high accuracy, since there are no doubt features common to translations from the five European languages of Europarl that are distinct from those of translations from the very different languages in IHT.

In fact, we find that training on Europarl and testing on IHT yields accuracy of 64.8%, while training on IHT and testing on Europarl yields accuracy of 58.8%. The weak results reflect both differences between the families of source languages involved in the respective corpora, as well as genre differences. Thus, for example, we find that of the pronouns shown in Table 4 above, only *he* and *his* are significantly under-represented in translationese in the IHT corpus. Thus, that effect is specific either to the genre of Europarl or to the European languages considered there.

Now, we combine the two corpora and check if we can identify translationese across two genres and eight languages. We run the same experiments as described above, using 200 texts from each of the eight source languages and 1600 non-translated English texts, 1000 from Europarl and 600 from IHT.

In 10-fold cross-validation, we find that we can distinguish translationese from non-translated English with accuracy of 90.5%.

This shows that there are features of translationese that cross genres and widely disparate languages. Thus, for one prominent example, we find that, as in Europarl, the word *the* is over-represented in translationese in IHT (15.36% in T vs. 13.31% in O; significant at $p<0.01$). In fact, the frequencies across corpora are astonishingly consistent.

To further appreciate this point, let's look at the frequencies of cohesive adverbs in the IHT corpus.

We find essentially, the same pattern in IHT as we did in Europarl. The preponderance of cohesive adverbs are over-represented in translationese, most of them with differences significant at $p<0.01$. Curiously, the word *actually* is a counter-example in both corpora.

word	freq O	freq T
therefore	0.011%	**0.031%**
thus	0.011%	**0.027%**
consequently	0.000%	**0.004%**
hence	0.003%	0.007%
accordingly	0.003%	0.003%
however	0.078%	**0.129%**
nevertheless	0.008%	**0.018%**
also	0.305%	**0.453%**
furthermore	0.003%	**0.011%**
moreover	0.009%	0.008%
indeed	0.018%	0.024%
actually	0.032%	0.018%

Table 7: Frequency of cohesive adverbs in O and T in the IHT corpus. Bold indicates significance at $p<0.01$.

5 Conclusions

We have found that we can learn classifiers that determine source language given a translated text, as well as classifiers that distinguish translated text from non-translated text in the source language. These text categorization experiments suggest that both source language and the mere fact of being

translated play a crucial role in the makeup of a translated text.

It is important to note that our learned classifiers are based solely on function words, so that, unlike earlier studies, the differences we find are unlikely to include cultural or thematic differences that might be artifacts of corpus construction.

In addition, we find that the exploitability of differences between translated texts and non-translated texts are related to the difference between source languages: translations from similar source languages are different from non-translated texts in similar ways.

Linguists use a variety of methods to quantify the extent of differences and similarities between languages. For example, Fusco (1990) studies translations between Spanish and Italian and considers the impact of structural differences between the two languages on translation quality. Studying the differences and distance between languages by comparing translations into the same language may serve as another way to deepen our typological knowledge. As we have seen, training on source language x and testing on source language y provides us with a good estimation of the distance between languages, in accordance with what we find in standard works on typology (cf. Katzner, 2002).

In addition to its intrinsic interest, the finding that the distance between languages is directly correlated with our ability to distinguish translations from a given source language from non-translated text is of great importance for several computational tasks. First, translations can be studied in order to shed new light on the differences between languages and can bear on attested techniques for using cognates to improve machine translation (Kondrak & Sherif, 2006). Additionally, given the results of our experiments, it stands to reason that using translated texts, especially from related source languages, will prove beneficial for constructing language models and will outperform results obtained from non-translated texts. This, too, bears on the quality of machine translation.

Finally, we find that there are general properties of translationese sufficiently strong that we can identify translationese even in a combined corpus that is comprised of eight very disparate languages across two distinct genres, one spoken and the other written. Prominent among these properties is the word *the*, as well as a number of cohesive adverbs, each of which is significantly over-represented in translated texts.

References

Mona Baker. 1993. Corpus linguistics and translation studies: Implications and applications. In Gill Francis Mona Baker and Elena Tognini Bonelli, editors, *Text and technology: in honour of John Sinclair*, pages 233-252. John Benjamins, Amsterdam.

Marco Baroni and Silvia Bernardini. 2006. A new approach to the study of Translationese: Machine-learning the difference between original and translated text. *Literary and Linguistic Computing*, 21(3):259-274.

Shoshan Blum-Kulka. Shifts of cohesion and coherence in translation. 1986. In Juliane House and Shoshana Blum-Kulka (Eds), *Interlingual and Intercultural Communication* (17-35). Tübingen: Günter Narr Verlag.

William Frawley. 1984. Prolegomenon to a theory of translation. In William Frawley (ed), *Translation. Literary, Linguistic and Philosophical Perspectives* (179-175). Newark: University of Delaware Press.

Maria Antonietta Fusco. 1990. Quality in conference interpreting between cognate languages: A preliminary approach to the Spanish-Italian case. *The Interpreters' Newsletter*, 3, 93-97.

Martin Gellerstam. 1986. Translationese in Swedish novels translated from English, in Lars Wollin & Hans Lindquist (eds.), *Translation Studies in Scandinavia* (88-95). Lund: CWK Gleerup.

Iustina Ilisei, Diana Inkpen, Gloria Corpas Pastor, and Ruslan Mitkov. Identification of translationese: A machine learning approach. In Alexander F. Gelbukh, editor, Proceedings of CICLing-2010: *Computational Linguistics and Intelligent Text Processing, 11th International, volume 6008 of Lecture Notes in Computer Science*, pages 503-511. Springer, 2010.

Kenneth Katzner. 2002. *The Languages of the World*. Routledge.

Grzegorz Kondrak and Tarek Sherif. 2006. Evaluation of several phonetic similarity algorithms on the task of cognate identification. In *Proceedings of the Workshop on Linguistic Distances* (LD '06). 43-50.

David Kurokawa, Cyril Goutte, and Pierre Isabelle. 2009. Automatic detection of translated text and its impact on machine translation. In Proceedings of MT-Summit XII.

Sara Laviosa: 1997. *How Comparable can 'Comparable Corpora' Be?*. Target, 9 (2), pp. 289-319.

Sara Laviosa-Braithwaite. 1998. In Mona Baker (ed.) *Routledge Encyclopedia of Translation Studies*. London/New York: Routledge, pp.288-291.

Sara Laviosa. 2002. *Corpus-based Translation Studies. Theory, Findings, Applications*. Amsterdam/New York: Rodopi.

David Madigan, Alexander Genkin, David D. Lewis and Dmitriy Fradkin 2005. Bayesian Multinomial Logistic Regression for Author Identification, In Maxent Conference, 509-516.

James W. Pennebaker, Martha E. Francis, and Roger J. Booth. 2001. *Linguistic Inquiry and Word Count (LIWC): LIWC2001 Manual*. Erlbaum Publishers, Mahwah, NJ, USA.

Helmut Schmid. Probabilistic Part-of-Speech Tagging Using Decision Trees. 2004. In *Proceedings of International Conference on New Methods in Language Processing*.

Larry Selinker.1972. Interlanguage. *International Review of Applied Linguistics*. 10, 209-241.

Gideon Toury. 1995. *Descriptive Translation Studies and beyond*. John Benjamins, Amsterdam / Philadelphia.

Hans van Halteren. 2008. Source language markers in EUROPARL translations. In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 937-944.

Rare Word Translation Extraction from Aligned Comparable Documents

Emmanuel Prochasson and Pascale Fung
Human Language Technology Center
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
{eemmanuel,pascale}@ust.hk

Abstract

We present a first known result of high precision rare word bilingual extraction from comparable corpora, using aligned comparable documents and supervised classification. We incorporate two features, a context-vector similarity and a co-occurrence model between words in aligned documents in a machine learning approach. We test our hypothesis on different pairs of languages and corpora. We obtain very high F-Measure between 80% and 98% for recognizing and extracting correct translations for rare terms (from 1 to 5 occurrences). Moreover, we show that our system can be trained on a pair of languages and test on a different pair of languages, obtaining a F-Measure of 77% for the classification of Chinese-English translations using a training corpus of Spanish-French. Our method is therefore even potentially applicable to low resources languages without training data.

1 Introduction

Rare words have long been a challenge to translate automatically using statistical methods due to their low occurrences. However, the *Zipf's Law* claims that, for any corpus of natural language text, the frequency of a word w_n (n being its rank in the frequency table) will be roughly twice as high as the frequency of word w_{n+1}. The logical consequence is that in any corpus, there are very few frequent words and many rare words.

We propose a novel approach to extract rare word translations from comparable corpora, relying on two main features.

The first feature is the *context-vector similarity* (Fung, 2000; Chiao and Zweigenbaum, 2002; Laroche and Langlais, 2010): each word is characterized by its context in both source and target corpora, words in translation should have similar context in both languages.

The second feature follows the assumption that specific terms and their translations should appear together often in documents on the same topic, and rarely in non-related documents. This is the general assumption behind early work on bilingual lexicon extraction from parallel documents using sentence boundary as the context window size for co-occurrence computation, we suggest to extend it to aligned comparable documents using document as the context window. This document context is too large for co-occurrence computation of functional words or high frequency content words, but we show through observations and experiments that this window size is appropriate for rare words.

Both these features are unreliable when the number of occurrences of words are low. We suggest however that they are complementary and can be used together in a machine learning approach. Moreover, we suggest that the model trained for one pair of languages can be successfully applied to extract translations from another pair of languages.

This paper is organized as follows. In the next section, we discuss the challenge of rare lexicon extraction, explaining the reasons why classic approaches on comparable corpora fail at dealing with rare words. We then discuss in section 3 the concept of *aligned comparable documents* and how we exploited those documents for bilingual lexicon extraction in section 4. We present our resources and implementation in section 5 then carry out and comment several experiments in section 6.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1327–1335,
Portland, Oregon, June 19-24, 2011.

2 The challenge of rare lexicon extraction

There are few previous works focusing on the extraction of rare word translations, especially from comparable corpora. One of the earliest works is from (Pekar et al., 2006). They emphasized the fact that the context-vector based approach, used for processing comparable corpora, *perform quite unreliably on all but the most frequent words*. In a nutshell[1], this approach proceeds by gathering the context of words in source and target languages inside *context-vectors*, then compares source and target context-vectors using similarity measures. In a monolingual context, such an approach is used to automatically get synonymy relationship between words to build thesaurus (Grefenstette, 1994). In the multilingual case, it is used to extract translations, that is, pairs of words with the same meaning in source and target corpora. It relies on the *Firthien* hypothesis that *you shall know a word by the company it keeps* (Firth, 1957).

To show that the frequency of a word influences its alignment, (Pekar et al., 2006) used six pairs of comparable corpora, ranking translations according to their frequencies. The less frequent words are ranked around 100-160 by their algorithm, while the most frequent ones typically appear at rank 20-40.

We ran a similar experiment using a French-English comparable corpus containing medical documents, all related to the topic of *breast cancer*, all manually classified as *scientific discourse*. The French part contains about 530,000 words while the English part contains about 7.4 millions words. For this experiment though, we sampled the English part to obtain a 530,000-words large corpus, matching the size of the French part.

Using an implementation of the context-vector similarity, we show in figure 1 that frequent words (above 400 occurrences in the corpus) reach a 60% precision whereas rare words (below 15 occurrences) are correctly aligned in only 5% of the time.

These results can be explained by the fact that, for the vector comparison to be efficient, the information they store has to be relevant and discriminatory. If there are not enough occurrences of a word, it is impossible to get a precise description of the *typical* context of this word, and therefore its description is likely to be very different for source and target words in translation.

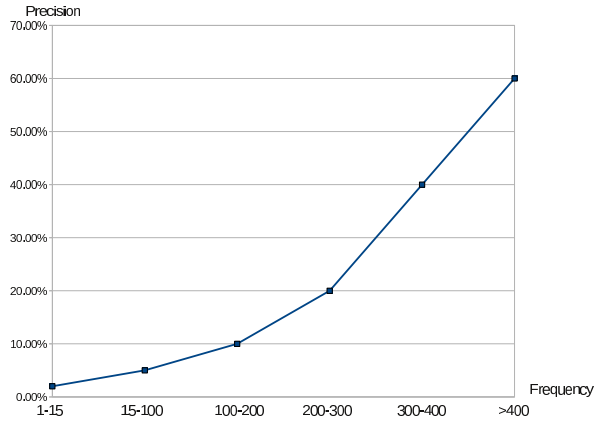


Figure 1: Results for context-vector based translations extraction with respect to word frequency. The vertical axis is the amount of correct translations found for Top_1, and the horizontal axis is the word occurrences in the corpus.

We confirmed this result with another observation on the full English part of the previous corpus, randomly split in 14 samples of the same size. The context-vectors for very frequent words, such as *cancer* (between 3,000 and 4,000 occurrences in each sample) are very similar across the subsets. Less frequent words, such as *abnormality* (between 70 and 16 occurrences in each sample) have very unstable context-vectors, hence a lower similarity across the subsets. This observation actually indicates that it will be difficult to align *abnormality* with itself.

3 Aligned comparable documents

A pair of *aligned comparable documents* is a particular case of comparable corpus: two comparable documents share the same topic and domain; they both relate the same information but are not mutual translations; although they might share parallel chunks (Munteanu and Marcu, 2005) – paragraphs, sentences or phrases – in the general case they were written independently. These comparable documents, when concatenated together in order, form an aligned comparable corpus.

[1]Detailed presentations can be found for example in (Fung, 2000; Chiao and Zweigenbaum, 2002; Laroche and Langlais, 2010).

Examples of such aligned documents can be found, for example in (Munteanu and Marcu, 2005): they aligned comparable documents with close publication dates. (Tao and Zhai, 2005) used an iterative, bootstrapping approach to align comparable documents using examples of already aligned corpora. (Smith et al., 2010) aligned documents from Wikipedia following the interlingual links provided on articles.

We take advantage of this alignment between documents: by looking at *what is common between two aligned documents* and *what is different in other documents*, we obtain more precise information about terms than when using a larger comparable corpus without alignment. This is especially interesting in the case of rare lexicon as the classic context-vector similarity is not discriminatory enough and fails at raising interesting translation for rare words.

4 Rare word translations from aligned comparable documents

4.1 Co-occurrence model

Different approaches have been proposed for bilingual lexicon extraction from parallel corpora, relying on the assumption that a word has one sense, one translation, no missing translation, and that its translation appears in aligned parallel sentences (Fung, 2000). Therefore, translations can be extracted by comparing the distribution of words across the sentences. For example, (Gale and Church, 1991) used a derivative of the χ^2 statistics to evaluate the association between words in aligned region of parallel documents. Such association scores evaluate the strength of the relation between events. In the case of parallel sentences and lexicon extraction, they measure how often two words appear in aligned sentences, and how often one appears without the other. More precisely, they will compare their number of co-occurrences against the expected number of co-occurrences under the null-hypothesis that words are randomly distributed. If they appear together more often than expected, they are considered as associated (Evert, 2008).

We focus in this work on *rare words*, more precisely on specialized terminology. We define them as the set of terms that appear from 1 (hapaxes) to 5 times. We use a strategy similar to the one applied on parallel sentences, but rely on *aligned documents*. Our hypothesis is very similar: words in translation should appear in aligned comparable documents. We used the *Jaccard similarity* (eq. 1) to evaluate the association between words among aligned comparable documents. In the general case, this measure would not give relevant scores due to frequency issue: it produces the same scores for two words that appear always together, and never one without the other, disregarding the fact that they appear 500 times or one time only. Other association scores generally rely on occurrence and co-occurrence counts to tackle this issue (such as the log-likelihood, eq. 2). In our case, the number of co-occurrences will be limited by the number of occurrences of the words, from 1 to 5. Therefore, the Jaccard similarity efficiently reflects what we want to observe.

$$J(w_i, w_j) = \frac{|A_i \cap A_j|}{|A_i \cup A_j|}; A_i = \{d : w_i \in d\} \quad (1)$$

A score of 1 indicates a perfect association (words always appear together, never one without the other), the more one word appears without the other, the lower the score.

4.2 Context-vector similarity

We implemented the context-vector similarity in a way similar to (Morin et al., 2007). In all experiments, we used the same set of parameters, as they yielded the best results on our corpora. We built the context-vectors using nouns only as seed lexicon, with a window size of 20. Source context-vectors are translated in the target language using the resources presented in the next section. We used the log-likelihood (Dunning, 1993, eq. 2) for context-vector normalization (O is the observed number of co-occurrence in the corpus, E is the expected number of co-occurrences under the null hypothesis). We used the Cosine similarity (eq. 3) for context-vector comparisons.

$$ll(w_i, w_j) = 2 \sum_{ij} O_{ij} log \frac{O_{ij}}{E_{ij}} \quad (2)$$

$$Cosine(A, B) = \frac{A \cdot B}{\|A\|^2 + \|B\|^2 - A \cdot B} \quad (3)$$

4.3 Binary classification of rare translations

We suggest to incorporate both the context-vector similarity and the co-occurrence features in a machine learning approach. This approach consists of training a classifier on positive examples of translation pairs, and negative examples of non-translations pairs. The trained model (in our case, a decision tree) is then used to tag an unknown pair of words as either ”Translation” or ”Non-Translation”.

One potential problem for building the training set, as pointed out for example by (Zhao and Ng, 2007) is this: we have a limited number of positive examples, but a very large amount of *non-translation* examples as obviously is the case for rare word translations in any training corpus. Including two many negative examples in the training set would lead the classifier to label every pairs as ”Non-Translation”.

To tackle this problem, (Zhao and Ng, 2007) tuned the imbalance of positive/negative ratio by resampling the positive examples in the training set. We chose to reduce the set of negative examples, and found that a ratio of five negative examples to one positive is optimal in our case. A lower ratio improves precision but reduces recall for the ”Translation” class.

It is also desirable that the classifier focuses on discriminating between confusing pairs of translations. As most of the negative examples have a null co-occurrence score and a null context-vector similarity, they are excluded from the training set. The negative examples are randomly chosen among those that fulfill the following constraints:

- non-null features ;
- ratio of number of occurrences between source/target words higher than 0.2 and lower than 5.

We use the J48 decision tree algorithm, in the *Weka* environment (Hall et al., 2009). Features are computed using the Jaccard similarity (section 3) for the co-occurrence model, and the implementation of the context-vector similarity presented in section 4.2.

4.4 Extension to another pair of languages

Even though the context vector similarity has been shown to achieve different accuracy depending on the pair of languages involved, the co-occurrence model is totally language independent. In the case of binary classification of translations, the two models are complementary to each other: word pairs with null co-occurrence are not considered by the context model while the context vector model gives more semantic information than the co-occurrence model.

For these reasons, we suggest that it is possible to use a decision tree trained on one pair of languages to extract translations from another pair of languages. A similar approach is proposed in (Alfonseca et al., 2008): they present a word decomposition model designed for German language that they successfully applied to other compounding languages. Our approach consists in training a decision tree on a pair of languages and applying this model to the classification of unknown pairs of words in another pair of languages. Such an approach is especially useful for prospecting new translations from less known languages, using a well known language as training.

We used the same algorithms and same features as in the previous sections, but used the data computed from one pair of languages as the training set, and the data computed from another pair of languages as the testing set.

5 Experimental setup

5.1 Corpora

We built several corpora using two different strategies. The first set was built using Wikipedia and the interlingual links available on articles (that points to another version of the same article in another language). We started from the list of all French articles[2] and randomly selected articles that provide a link to Spanish and English versions. We downloaded those, and clean them by removing the wikipedia formatting tags to obtain raw UTF8 texts. Articles were not selected based on their sizes, the vocabulary used, nor a particular topic. We obtained about 20,000 aligned documents for each language.

A second set was built using an in-house system

[2] Available on `http://download.wikimedia.org/`.

	[WP] French	[WP] English	[WP] Es	[CLIR] En	[CLIR] Zh
#documents	20,169	20,169	20,169	15,3247	15,3247
#tokens	4,008,284	5,470,661	2,741,789	1,334,071	1,228,330
#unique tokens	120,238	128,831	103,398	30,984	60,015

Table 1: Statistics for all parts of all corpora.

(unpublished) that seeks for comparable and parallel documents from the web. Starting from a list of Chinese documents (in this case, mostly news articles), we automatically selected English target documents using Cross Language Information Retrieval. About 85% of the paired documents obtained are direct translations (header/footer of web pages apart). However, they will be processed just like aligned comparable documents, that is, we will not take advantage of the structure of the parallel contents to improve accuracy, but will use the exact same approach that we applied for the Wikipedia documents. We gathered about 15,000 pairs of documents employing this method.

All corpora were processed using Tree-Tagger[3] for segmentation and Part-of-Speech tagging. We focused on nouns only and discarded all other tokens. We would record the lemmatized form of tokens when available, otherwise we would record the original form. Table 1 summarizes main statistics for each corpus; [WP] refers to the Wikipedia corpora, [CLIR] to the Chinese-English corpora extracted through cross language information retrieval.

5.2 Dictionaries

We need a bilingual seed lexicon for the context-vector similarity. We used a French-English lexicon obtained from the Web. It contains about 67,000 entries. The Spanish-English and Spanish-French dictionaries were extracted from the linguistic resources of the Apertium project[4]. We obtained approximately 22,500 Spanish-English translations and 12,000 for Spanish-French. Finally, for Chinese-English we used the LDC2002L27 resource from the Linguistic Data Consortium[5] with about 122,000 entries.

[3] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html

[4] http://www.apertium.org

[5] http://www.ldc.upenn.edu

5.3 Evaluation lists

To evaluate our approach, we needed evaluation lists of terms for which translations are already known. We used the Medical Subject Headlines, from the UMLS meta-thesaurus[6] which provides a lexicon of specialized, medical terminology, notably in Spanish, English and French. We used the LDC lexicon presented in the previous section for Chinese-English.

From these resources, we selected all the source words that appears from 1 to 5 times in the corpora in order to build the evaluation lists.

5.4 Oracle translations

We looked at the corpora to evaluate how many translation pairs from the evaluation lists can be found across the aligned comparable documents. Those translations are hereafter the *oracle translations*. For French/English, French/Spanish and English/Spanish, about 60% of the translation pairs can be found. For Chinese/English, this ratio reaches 45%. The main reason for this lower result is the inaccuracy of the segmentation tool used to process Chinese. Segmentation tools usually rely on a training corpus and typically fail at handling rare words which, by definition, were unlikely to be found in the training examples. Therefore, some rare Chinese tokens found in our corpus are the results of faulty segmentation, and the translation of those faulty words can not be found in related documents. We encountered the same issue but at a much lower degree for other languages because of spelling mistakes and/or improper Part-of-Speech tagging.

6 Experiments

We ran three different experiments. Experiment I compares the accuracy of the context-vector similarity and the co-occurrence model. Experiment II uses supervised classification with both features.

[6] http://www.nlm.nih.gov/research/umls/

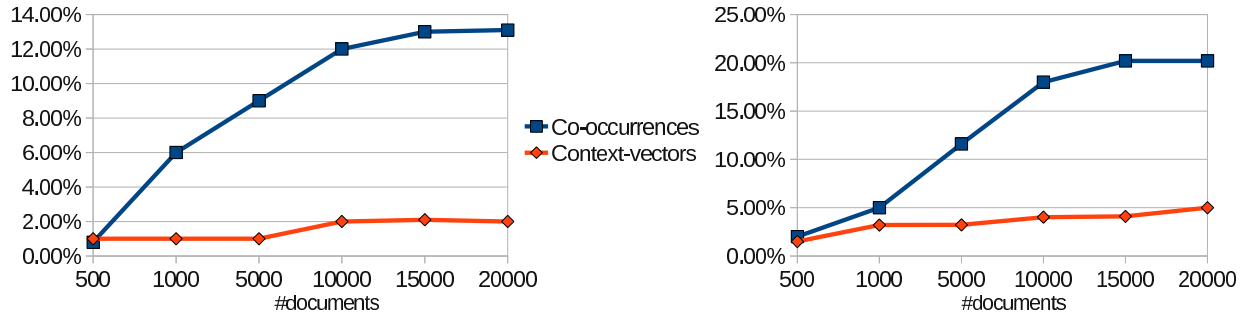


Figure 2: Experiment I: comparison of accuracy obtained for the Top_{10} with the context-vector similarity and the co-occurrence model, for hapaxes (left) and words that appear 2 to 5 times (right).

Experiment III extracts translation from a pair of languages, using a classifier trained on another pair of languages.

6.1 Experiment I: co-occurrence model vs. context-vector similarity

We split the French-English part of the Wikipedia corpus into different samples: the first sample contains 500 pairs of documents. We then aggregated more documents to this initial sample to test different sizes of corpora. We built the sample in order to ensure hapaxes in the whole corpus are hapaxes in all subsets. That is, we ensured the 431 hapaxes in the evaluation lists are represented in the 500 documents subset.

We extracted translations in two different ways:

1. using the co-occurrence model;
2. using the context-vector based approach, with the same evaluation lists.

The accuracy is computed on 1,000 pairs of translations from the set of oracle translations, and measures the amount of correct translations found for the 10 best ranks (Top_{10}) after ranking the candidates according to their score (context-vector similarity or co-occurrence model). The results are presented in figure 2.

We can draw two conclusions out of these results. First, the size of the corpus influences the quality of the bilingual lexicon extraction when using the co-occurrence model. This is especially interesting with hapaxes, for which frequency does not change with the increase of the size of the corpora. The accuracy is improved by adding more information to the corpus, even if this additional information does not cover the pairs of translations we are looking for. The added documents will weaken the association of incorrect translations, without changing the association for rare terms translations. For example, the precision for hapaxes using the co-occurrence model ranges from less than 1% when using only 500 pairs of documents, to about 13% when using all documents. The second conclusion is that the co-occurrence model outperforms the context-vector similarity.

However, both these approaches still perform poorly. In the next experiment, we propose to combine them using supervised classification.

6.2 Experiment II: binary classification of translation

For each corpus or combination of corpora – English-Spanish, English-French, Spanish-French and Chinese-English, we ran three experiments, using the following features for supervised learning of translations:

- the context-vector similarity;
- the co-occurrence model;
- both features together.

The parameters are discussed in section 4.3. We used all the oracle translations to train the positive values. Results are presented in table 2, they are computed using a 10-folds cross validation. Class T refers to "Translation", $\neg T$ to "Non-Translation". The evaluation of precision/recall/F-Measure for the class "Translation" are given in equation 4 to 6.

	Precision	Recall	F-Measure	Cl.
English-Spanish				
context-vectors	0.0%	0.0%	0.0%	T
	83.3%	99.9%	90.8%	$\neg T$
co-occ. model	66.2%	44.2%	53.0%	T
	89.5%	95.5%	92.4%	$\neg T$
both	**98.6%**	**88.6%**	**93.4%**	T
	97.8%	99.8%	98.7%	$\neg T$
French-English				
context-vectors	76.5%	10.3%	18.1%	T
	90.9%	99.6%	95.1%	$\neg T$
co-occ. model	85.7%	1.2%	2.4%	T
	90.1%	100%	94.8%	$\neg T$
both	**81.0%**	**80.2%**	**80.6%**	T
	94.9%	98.7%	96.8%	$\neg T$
French-Spanish				
context-vectors	0.0%	0.0%	0.0%	T
	81.0%	100%	89.5%	$\neg T$
co-occ. model	64.2%	46.5%	53.9%	T
	88.2%	93.9%	91.0%	$\neg T$
both	**98.7%**	**94.6%**	**96.7%**	T
	98.8%	99.7%	99.2%	$\neg T$
Chinese-English				
context-vectors	69.6%	13.3%	22.3%	T
	91.0%	93.1%	92.1%	$\neg T$
co-occ. model	73.8%	32.5%	45.1%	T
	85.2%	97.1%	90.8%	$\neg T$
both	**86.7%**	**74.7%**	**80.3%**	T
	96.3%	98.3%	97.3%	$\neg T$

Table 2: Experiment II: results of binary classification for "Translation" and "Non-Translation".

$$precision_T = \frac{|T \cap oracle|}{|T|} \quad (4)$$

$$recall_T = \frac{|T \cap oracle|}{|oracle|} \quad (5)$$

$$FMeasure = 2 \times \frac{precision \times recall}{precision + recall} \quad (6)$$

These results show first that one feature is generally not discriminatory enough to discern correct translation and non-translation pairs. For example with Spanish-English, by using context-vector similarity only, we obtained very high recall/precision for the classification of "Non-Translation", but null precision/recall for the classification of "Translation". In some other cases, we obtained high precision but poor recall with one feature only, which is not a usefully result as well since most of the correct translations are still labeled as "Non-Translation".

However, when using both features, the precision is strongly improved up to 98% (English-Spanish or French-Spanish) with a high recall of about 90% for class T. We also achieved about 86%/75% precision/recall in the case of Chinese-English, even though they are very distant languages. This last result is also very promising since it has been obtained from a fully automatically built corpus. Table 3 shows some examples of correctly labeled "Translation".

The decision trees obtained indicate that, in general, word pairs with very high co-occurrence model scores are translations, and that the context-vector similarity disambiguate candidates with lower co-occurrence model scores. Interestingly, the trained decision trees are very similar between the different pairs of languages, which inspired the next experiment.

6.3 Experiment III: extension to another pair of languages

In the last experiment, we focused on using the knowledge acquired with a given pair of languages to recognize proper translation pairs using a different pair of languages. For this experiment, we used the data from one corpus to train the classifier, and used the data from another combination of languages as the test set. Results are displayed in table 4.

These last results are of great interest because they show that translation pairs can be correctly classified even with a classifier trained on another pair of languages. This is very promising because it allows one to prospect new languages using knowledge acquired on a known pairs of languages. As an example, we reached a 77% F-Measure for Chinese-English alignment using a classifier trained on Spanish-French features. This not only confirms the precision/recall of our approach in general, but also shows that the model obtained by training tends to be very stable and accurate across different pairs of languages and different corpora.

	Tested with			
Trained with	Sp-En	Sp-Fr	Fr-En	Zh-En
Sp-En	98.6/88.8/93.5	98.7/94.9/96.8	91.5/48.3/63.2	99.3/63.0/77.1
Sp-Fr	89.5/77.9/83.9	90.4/82.9/86.5	75.4/53.5/62.6	98.7/63.3/77.1
Fr-En	89.5/77.9/83.9	90.4/82.9/86.5	85.2/80.0/82.6	81.0/87.6/84.2
Zh-En	96.6/89.2/92.7	97.7/94.9/96.3	81.1/50.9/62.5	97.4/65.1/78.1

Table 4: Experiment III: Precision/Recall/F-Measure for label "Translation", obtained for all training/testing set combinations.

English	French
myometrium	myomètre
lysergide	lysergide
hyoscyamus	jusquiame
lysichiton	lysichiton
brassicaceae	brassicacées
yarrow	achillée
spikemoss	sélaginelle
leiomyoma	fibromyome
ryegrass	ivraie
English	Spanish
spirometry	espirometría
lolium	lolium
omentum	epiplón
pilocarpine	pilocarpina
chickenpox	varicela
bruxism	bruxismo
psittaciformes	psittaciformes
commodification	mercantilización
talus	astrágalo
English	Chinese
hooliganism	流氓
kindergarten	幼儿园
oyster	牡蛎
fascism	法西斯主义
taxonomy	分类学
mongolian	蒙古人
subpoena	传票
rupee	卢比
archbishop	大主教
serfdom	农奴
typhoid	伤寒

Table 3: Experiment II and III: examples of rare word translations found by our algorithm. Note that even though some words such as "*kindergarten*" are not rare in general, they occur with very low frequency in the test corpus.

7 Conclusion

We presented a new approach for extracting translations of rare words among aligned comparable documents. To the best of our knowledge, this is one of the first high accuracy extraction of rare lexicon from non-parallel documents. We obtained a F-Measure ranging from about 80% (French-English, Chinese-English) to 97% (French-Spanish). We also obtained good results for extracting lexicon for a pair of languages, using a decision tree trained with the data computed on another pair of languages. We yielded a 77% F-Measure for the extraction of Chinese-English lexicon, using Spanish-French for training the model.

On top of these promising results, our approach presents several other advantages. First, we showed that it works well on automatically built corpora which require minimal human intervention. Aligned comparable documents can easily be collected and are available in large volumes. Moreover, the proposed machine learning method incorporating both context-vector and co-occurrence model has shown to give good results on pairs of languages that are very different from each other, such as Chinese-English. It is also applicable across different training and testing language pairs, making it possible for us to find rare word translations even for languages without training data. The co-occurrence model is completely language independent and have been shown to give good results on various pairs of languages, including Chinese-English.

Acknowledgments

The authors would like to thank Emmanuel Morin (LINA CNRS 6241) for providing us the comparable corpus used for the experiment in section 2, Simon Shi for extracting and providing the corpus

described in section 5.1, and the anonymous reviewers for their valuable comments. This research is partly supported by ITS/189/09 AND BBNX02-20F00310/11PN.

References

Enrique Alfonseca, Slaven Bilac, and Stefan Pharies. 2008. Decompounding query keywords from compounding languages. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL'08)*, pages 253–256.

Yun-Chuang Chiao and Pierre Zweigenbaum. 2002. Looking for candidate translational equivalents in specialized, comparable corpora. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, pages 1208–1212.

Ted Dunning. 1993. Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, 19(1):61–74.

Stefan Evert. 2008. Corpora and collocations. In A. Ludeling and M. Kyto, editors, *Corpus Linguistics. An International Handbook*, chapter 58. Mouton de Gruyter, Berlin.

John Firth. 1957. *A synopsis of linguistic theory 1930-1955*. Studies in Linguistic Analysis, Philological. Longman.

Pascale Fung. 2000. A statistical view on bilingual lexicon extraction–from parallel corpora to non-parallel corpora. In Jean Véronis, editor, *Parallel Text Processing*, page 428. Kluwer Academic Publishers.

William A. Gale and Kenneth W. Church. 1991. Identifying word correspondence in parallel texts. In *Proceedings of the workshop on Speech and Natural Language*, HLT'91, pages 152–157, Morristown, NJ, USA. Association for Computational Linguistics.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publisher.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: An update. *SIGKDD Explorations*, 11.

Audrey Laroche and Philippe Langlais. 2010. Revisiting context-based projection methods for term-translation spotting in comparable corpora. In *23rd International Conference on Computational Linguistics (Coling 2010)*, pages 617–625, Beijing, China, Aug.

Emmanuel Morin, Béatrice Daille, Koichi Takeuchi, and Kyo Kageura. 2007. Bilingual Terminology Mining – Using Brain, not brawn comparable corpora. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07)*, pages 664–671, Prague, Czech Republic.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving Machine Translation Performance by Exploiting Non-Parallel Corpora. *Computational Linguistics*, 31(4):477–504.

Viktor Pekar, Ruslan Mitkov, Dimitar Blagoev, and Andrea Mulloni. 2006. Finding translations for low-frequency words in comparable corpora. *Machine Translation*, 20(4):247–266.

Jason R. Smith, Chris Quirk, and Kristina Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL*, pages 403–411.

Tao Tao and ChengXiang Zhai. 2005. Mining comparable bilingual text corpora for cross-language information integration. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 691–696, New York, NY, USA. ACM.

Shanheng Zhao and Hwee Tou Ng. 2007. Identification and resolution of Chinese zero pronouns: A machine learning approach. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, Prague, Czech Republic.

Using Bilingual Parallel Corpora for Cross-Lingual Textual Entailment

Yashar Mehdad
FBK - irst and Uni. of Trento
Povo (Trento), Italy
mehdad@fbk.eu

Matteo Negri
FBK - irst
Povo (Trento), Italy
negri@fbk.eu

Marcello Federico
FBK - irst
Povo (Trento), Italy
federico@fbk.eu

Abstract

This paper explores the use of bilingual parallel corpora as a source of lexical knowledge for cross-lingual textual entailment. We claim that, in spite of the inherent difficulties of the task, phrase tables extracted from parallel data allow to capture both lexical relations between single words, and contextual information useful for inference. We experiment with a phrasal matching method in order to: *i)* build a system portable across languages, and *ii)* evaluate the contribution of lexical knowledge in isolation, without interaction with other inference mechanisms. Results achieved on an English-Spanish corpus obtained from the RTE3 dataset support our claim, with an overall accuracy above average scores reported by RTE participants on monolingual data. Finally, we show that using parallel corpora to extract paraphrase tables reveals their potential also in the monolingual setting, improving the results achieved with other sources of lexical knowledge.

1 Introduction

Cross-lingual Textual Entailment (CLTE) has been proposed by (Mehdad et al., 2010) as an extension of Textual Entailment (Dagan and Glickman, 2004) that consists in deciding, given two texts T and H *in different languages*, if the meaning of H can be inferred from the meaning of T. The task is inherently difficult, as it adds issues related to the multilingual dimension to the complexity of semantic inference at the textual level. For instance, the reliance of current monolingual TE systems on lexical resources (*e.g.* WordNet, VerbOcean, FrameNet) and deep processing components (*e.g.* syntactic and semantic parsers, co-reference resolution tools, temporal expressions recognizers and normalizers) has to confront, at the cross-lingual level, with the limited availability of lexical/semantic resources covering multiple languages, the limited coverage of the existing ones, and the burden of integrating language-specific components into the same cross-lingual architecture.

As a first step to overcome these problems, (Mehdad et al., 2010) proposes a "basic solution", that brings CLTE back to the monolingual scenario by translating H into the language of T. Despite the advantages in terms of modularity and portability of the architecture, and the promising experimental results, this approach suffers from one main limitation which motivates the investigation on alternative solutions. Decoupling machine translation (MT) and TE, in fact, ties CLTE performance to the availability of MT components, and to the quality of the translations. As a consequence, on one side translation errors propagate to the TE engine hampering the entailment decision process. On the other side such unpredictable errors reduce the possibility to control the behaviour of the engine, and devise *ad-hoc* solutions to specific entailment problems.

This paper investigates the idea, still unexplored, of a tighter integration of MT and TE algorithms and techniques. Our aim is to embed cross-lingual processing techniques inside the TE recognition process in order to avoid any dependency on external MT components, and eventually gain full control of the system's behaviour. Along this direction, we

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1336–1345,
Portland, Oregon, June 19-24, 2011.

start from the acquisition and use of lexical knowledge, which represents the basic building block of any TE system. Using the basic solution proposed by (Mehdad et al., 2010) as a term of comparison, we experiment with different sources of multilingual lexical knowledge to address the following questions:

(1) What is the potential of the existing multilingual lexical resources to approach CLTE? To answer this question we experiment with lexical knowledge extracted from bilingual dictionaries, and from a multilingual lexical database. Such experiments show two main limitations of these resources, namely: *i)* their limited coverage, and *ii)* the difficulty to capture contextual information when only associations between single words (or at most named entities and multiword expressions) are used to support inference.

(2) Does MT provide useful resources or techniques to overcome the limitations of existing resources? We envisage several directions in which inputs from MT research may enable or improve CLTE. As regards the resources, phrase and paraphrase tables extracted from bilingual parallel corpora can be exploited as an effective way to capture both lexical relations between single words, and contextual information useful for inference. As regards the algorithms, statistical models based on co-occurrence observations, similar to those used in MT to estimate translation probabilities, may contribute to estimate entailment probabilities in CLTE. Focusing on the resources direction, the main contribution of this paper is to show that the lexical knowledge extracted from parallel corpora allows to significantly improve the results achieved with other multilingual resources.

(3) In the cross-lingual scenario, can we achieve results comparable to those obtained in monolingual TE? Our experiments show that, although CLTE seems intrinsically more difficult, the results obtained using phrase and paraphrase tables are better than those achieved by average systems on monolingual datasets. We argue that this is due to the fact that parallel corpora are a rich source of cross-lingual paraphrases with no equivalents in monolingual TE.

(4) Can parallel corpora be useful also for monolingual TE? To answer this question, we experiment on monolingual RTE datasets using paraphrase tables extracted from bilingual parallel corpora. Our results improve those achieved with the most widely used resources in monolingual TE, namely WordNet, Verbocean, and Wikipedia.

The remainder of this paper is structured as follows. Section 2 shortly overviews the role of lexical knowledge in textual entailment, highlighting a gap between TE and CLTE in terms of available knowledge sources. Sections 3 and 4 address the first three questions, giving motivations for the use of bilingual parallel corpora in CLTE, and showing the results of our experiments. Section 5 addresses the last question, reporting on our experiments with paraphrase tables extracted from phrase tables on the monolingual RTE datasets. Section 6 concludes the paper, and outlines the directions of our future research.

2 Lexical resources for TE and CLTE

All current approaches to monolingual TE, either syntactically oriented (Rus et al., 2005), or applying logical inference (Tatu and Moldovan, 2005), or adopting transformation-based techniques (Kouleykov and Magnini, 2005; Bar-Haim et al., 2008), incorporate different types of lexical knowledge to support textual inference. Such information ranges from *i)* lexical paraphrases (textual equivalences between terms) to *ii)* lexical relations preserving entailment between words, and *iii)* word-level similarity/relatedness scores. WordNet, the most widely used resource in TE, provides all the three types of information. Synonymy relations can be used to extract lexical paraphrases indicating that words from the text and the hypothesis entail each other, thus being interchangeable. Hypernymy/hyponymy chains can provide entailment-preserving relations between concepts, indicating that a word in the hypothesis can be replaced by a word from the text. Paths between concepts and glosses can be used to calculate similarity/relatedness scores between single words, that contribute to the computation of the overall similarity between the text and the hypothesis.

Besides WordNet, the RTE literature documents the use of a variety of lexical information sources (Bentivogli et al., 2010; Dagan et al., 2009). These include, just to mention the most popular

ones, DIRT (Lin and Pantel, 2001), VerbOcean (Chklovski and Pantel, 2004), FrameNet (Baker et al., 1998), and Wikipedia (Mehdad et al., 2010; Kouylekov et al., 2009). DIRT is a collection of statistically learned inference rules, that is often integrated as a source of lexical paraphrases and entailment rules. VerbOcean is a graph of fine-grained semantic relations between verbs, which are frequently used as a source of precise entailment rules between predicates. FrameNet is a knowledge-base of frames describing prototypical situations, and the role of the participants they involve. It can be used as an alternative source of entailment rules, or to determine the semantic overlap between texts and hypotheses. Wikipedia is often used to extract probabilistic entailment rules based word similarity/relatedness scores.

Despite the consensus on the usefulness of lexical knowledge for textual inference, determining the actual impact of these resources is not straightforward, as they always represent one component in complex architectures that may use them in different ways. As emerges from the ablation tests reported in (Bentivogli et al., 2010), even the most common resources proved to have a positive impact on some systems and a negative impact on others. Some previous works (Bannard and Callison-Burch, 2005; Zhao et al., 2009; Kouylekov et al., 2009) indicate, as main limitations of the mentioned resources, their limited coverage, their low precision, and the fact that they are mostly suitable to capture relations mainly between single words.

Addressing CLTE we have to face additional and more problematic issues related to: *i)* the stronger need of lexical knowledge, and *ii)* the limited availability of multilingual lexical resources. As regards the first issue, it's worth noting that in the monolingual scenario simple "bag of words" (or "bag of n-grams") approaches are *per se* sufficient to achieve results above baseline. In contrast, their application in the cross-lingual setting is not a viable solution due to the impossibility to perform direct lexical matches between texts and hypotheses in different languages. This situation makes the availability of multilingual lexical knowledge a necessary condition to bridge the language gap. However, with the only exceptions represented by WordNet and Wikipedia, most of the aforementioned resources are available only for English. Multilingual lexical databases aligned with the English WordNet (*e.g.* MultiWordNet (Pianta et al., 2002)) have been created for several languages, with different degrees of coverage. As an example, the 57,424 synsets of the Spanish section of MultiWordNet aligned to English cover just around 50% of the WordNet's synsets, thus making the coverage issue even more problematic than for TE. As regards Wikipedia, the cross-lingual links between pages in different languages offer a possibility to extract lexical knowledge useful for CLTE. However, due to their relatively small number (especially for some languages), bilingual lexicons extracted from Wikipedia are still inadequate to provide acceptable coverage. In addition, featuring a bias towards named entities, the information acquired through cross-lingual links can at most complement the lexical knowledge extracted from more generic multilingual resources (*e.g* bilingual dictionaries).

3 Using Parallel Corpora for CLTE

Bilingual parallel corpora represent a possible solution to overcome the inadequacy of the existing resources, and to implement a portable approach for CLTE. To this aim, we exploit parallel data to: *i)* learn alignment criteria between phrasal elements in different languages, *ii)* use them to automatically extract lexical knowledge in the form of *phrase tables*, and *iii)* use the obtained phrase tables to create monolingual *paraphrase tables*.

Given a cross-lingual T/H pair (with the text in l_1 and the hypothesis in l_2), our approach leverages the vast amount of lexical knowledge provided by phrase and paraphrase tables to map H into T. We perform such mapping with two different methods. The **first method** uses a single phrase table to directly map phrases extracted from the hypothesis to phrases in the text. In order to improve our system's generalization capabilities and increase the coverage, the **second method** combines the phrase table with two monolingual paraphrase tables (one in l_1, and one in l_2). This allows to:

1. use the paraphrase table in l_2 to find paraphrases of phrases extracted from H;
2. map them to entries in the phrase table, and extract their equivalents in l_1;

3. use the paraphrase table in l_1 to find paraphrases of the extracted fragments in l_1;

4. map such paraphrases to phrases in T.

With the second method, phrasal matches between the text and the hypothesis are indirectly performed through paraphrases of the phrase table entries.

The final entailment decision for a T/H pair is assigned considering a model learned from the similarity scores based on the identified phrasal matches. In particular, "YES" and "NO" judgements are assigned considering the proportion of words in the hypothesis that are found also in the text. This way to approximate entailment reflects the intuition that, as a directional relation between the text and the hypothesis, the full content of H has to be found in T.

3.1 Extracting Phrase and Paraphrase Tables

Phrase tables (PHT) contain pairs of corresponding phrases in two languages, together with association probabilities. They are widely used in MT as a way to figure out how to translate input in one language into output in another language (Koehn et al., 2003). There are several methods to build phrase tables. The one adopted in this work consists in learning phrase alignments from a word-aligned bilingual corpus. In order to build English-Spanish phrase tables for our experiments, we used the freely available Europarl V.4, News Commentary and United Nations Spanish-English parallel corpora released for the WMT10[1]. We run TreeTagger (Schmid, 1994) for tokenization, and used the Giza++ (Och and Ney, 2003) to align the tokenized corpora at the word level. Subsequently, we extracted the bilingual phrase table from the aligned corpora using the Moses toolkit (Koehn et al., 2007). Since the resulting phrase table was very large, we eliminated all the entries with identical content in the two languages, and the ones containing phrases longer than 5 words in one of the two sides. In addition, in order to experiment with different phrase tables providing different degrees of coverage and precision, we extracted 7 phrase tables by pruning the initial one on the direct phrase translation probabilities of $0.01, 0.05, 0.1, 0.2, 0.3, 0.4$ and 0.5. The resulting phrase tables range from 76 to 48 million entries, with an average of 3.9 words per phrase.

Paraphrase tables (PPHT) contain pairs of corresponding phrases in the same language, possibly associated with probabilities. They proved to be useful in a number of NLP applications such as natural language generation (Iordanskaja et al., 1991), multidocument summarization (McKeown et al., 2002), automatic evaluation of MT (Denkowski and Lavie, 2010), and TE (Dinu and Wang, 2009).

One of the proposed methods to extract paraphrases relies on a pivot-based approach using phrase alignments in a bilingual parallel corpus (Bannard and Callison-Burch, 2005). With this method, all the different phrases in one language that are aligned with the same phrase in the other language are extracted as paraphrases. After the extraction, pruning techniques (Snover et al., 2009) can be applied to increase the precision of the extracted paraphrases.

In our work we used available[2] paraphrase databases for English and Spanish which have been extracted using the method previously outlined. Moreover, in order to experiment with different paraphrase sets providing different degrees of coverage and precision, we pruned the main paraphrase table based on the probabilities, associated to its entries, of $0.1, 0.2$ and 0.3. The number of phrase pairs extracted varies from 6 million to about 80000, with an average of 3.2 words per phrase.

3.2 Phrasal Matching Method

In order to maximize the usage of lexical knowledge, our entailment decision criterion is based on similarity scores calculated with a phrase-to-phrase matching process.

A phrase in our approach is an *n-gram* composed of up to 5 consecutive words, excluding punctuation. Entailment decisions are estimated by combining phrasal matching scores ($Score_n$) calculated for each level of *n-grams* , which is the number of *1-grams*, *2-grams*,..., *5-grams* extracted from H that match with *n-grams* in T. Phrasal matches are performed either at the level of tokens, lemmas, or stems, can be of two types:

[1] http://www.statmt.org/wmt10/

[2] http://www.cs.cmu.edu/ alavie/METEOR

1. **Exact**: in the case that two phrases are identical at one of the three levels (token, lemma, stem);

2. **Lexical**: in the case that two different phrases can be mapped through entries of the resources used to bridge T and H (*i.e.* phrase tables, paraphrases tables, dictionaries or any other source of lexical knowledge).

For each phrase in H, we first search for exact matches at the level of token with phrases in T. If no match is found at a token level, the other levels (lemma and stem) are attempted. Then, in case of failure with exact matching, lexical matching is performed at the same three levels. To reduce redundant matches, the lexical matches between pairs of phrases which have already been identified as exact matches are not considered.

Once matching for each *n-gram* level has been concluded, the number of matches (M_n) and the number of phrases in the hypothesis (Nn) are used to estimate the portion of phrases in H that are matched at each level (*n*). The phrasal matching score for each *n-gram* level is calculated as follows:

$$Score_n = \frac{M_n}{Nn}$$

To combine the phrasal matching scores obtained at each *n-gram* level, and optimize their relative weights, we trained a Support Vector Machine classifier, SVMlight (Joachims, 1999), using each score as a feature.

4 Experiments on CLTE

To address the first two questions outlined in Section 1, we experimented with the phrase matching method previously described, contrasting the effectiveness of lexical information extracted from parallel corpora with the knowledge provided by other resources used in the same way.

4.1 Dataset

The dataset used for our experiments is an English-Spanish entailment corpus obtained from the original RTE3 dataset by translating the English hypothesis into Spanish. It consists of 1600 pairs derived from the RTE3 development and test sets (800+800). Translations have been generated by the CrowdFlower[3] channel to Amazon Mechanical Turk[4] (MTurk), adopting the methodology proposed by (Negri and Mehdad, 2010). The method relies on translation-validation cycles, defined as separate jobs routed to MTurk's workforce. Translation jobs return one Spanish version for each hypothesis. Validation jobs ask multiple workers to check the correctness of each translation using the original English sentence as reference. At each cycle, the translated hypothesis accepted by the majority of trustful validators[5] are stored in the CLTE corpus, while wrong translations are sent back to workers in a new translation job. Although the quality of the results is enhanced by the possibility to automatically weed out untrusted workers using gold units, we performed a manual quality check on a subset of the acquired CLTE corpus. The validation, carried out by a Spanish native speaker on 100 randomly selected pairs after two translation-validation cycles, showed the good quality of the collected material, with only 3 minor "errors" consisting in controversial but substantially acceptable translations reflecting regional Spanish variations.

The T-H pairs in the collected English-Spanish entailment corpus were annotated using TreeTagger (Schmid, 1994) and the Snowball stemmer[6] with token, lemma, and stem information.

4.2 Knowledge sources

For comparison with the extracted phrase and paraphrase tables, we use a large bilingual dictionary and MultiWordNet as alternative sources of lexical knowledge.

Bilingual dictionaries (DIC) allow for precise mappings between words in H and T. To create a large bilingual English-Spanish dictionary we processed and combined the following dictionaries and bilingual resources:
- XDXF Dictionaries[7]: 22,486 entries.

[3] http://crowdflower.com/

[4] https://www.mturk.com/mturk/

[5] Workers' trustworthiness can be automatically determined by means of hidden gold units randomly inserted into jobs.

[6] http://snowball.tartarus.org/

[7] http://xdxf.revdanica.com/

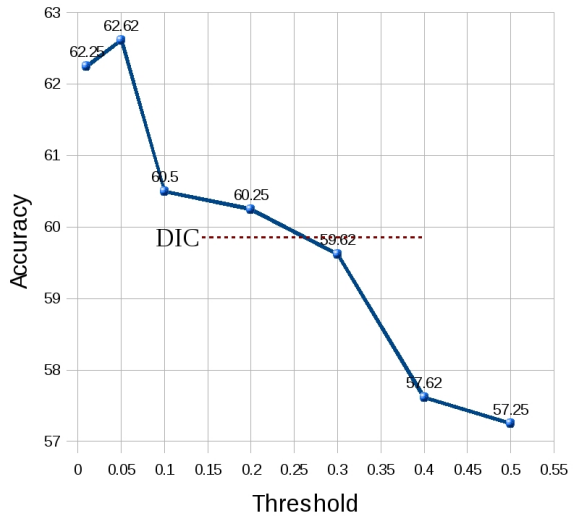


Figure 1: Accuracy on CLTE by pruning the phrase table with different thresholds.

- Universal dictionary database[8]: 9,944 entries.
- Wiktionary database[9]: 5,866 entries.
- Omegawiki database[10]: 8,237 entries.
- Wikipedia interlanguage links[11]: 7,425 entries.

The resulting dictionary features 53,958 entries, with an average length of 1.2 words.

MultiWordNet (MWN) allows to extract mappings between English and Spanish words connected by entailment-preserving semantic relations. The extraction process is dataset-dependent, as it checks for synonymy and hyponymy relations only between terms found in the dataset. The resulting collection of cross-lingual words associations contains 36,794 pairs of lemmas.

4.3 Results and Discussion

Our results are calculated over 800 test pairs of our CLTE corpus, after training the SVM classifier over 800 development pairs. This section reports the percentage of correct entailment assignments (accuracy), comparing the use of different sources of lexical knowledge.

Initially, in order to find a reasonable trade-off between precision and coverage, we used the 7 phrase tables extracted with different pruning thresholds (see Section 3.1). Figure 1 shows that with the pruning threshold set to 0.05, we obtain the highest result of 62.62% on the test set. The curve demonstrates that, although with higher pruning thresholds we retain more reliable phrase pairs, their smaller number provides limited coverage leading to lower results. In contrast, the large coverage obtained with the pruning threshold set to 0.01 leads to a slight performance decrease due to probably less precise phrase pairs.

MWN	DIC	PHT	PPHT	Acc.	δ
x				55.00	0.00
	x			59.88	+4.88
		x		**62.62**	+7.62
		x	x	**62.88**	+7.88

Table 1: Accuracy results on CLTE using different lexical resources.

Once the threshold has been set, in order to prove the effectiveness of information extracted from bilingual corpora, we conducted a series of experiments using the different resources mentioned in Section 4.2.

As it can be observed in Table 1, the highest results are achieved using the phrase table, both alone and in combination with paraphrase tables (62.62% and 62.88% respectively). These results suggest that, with appropriate pruning thresholds, the large number and the longer entries contained in the phrase and paraphrase tables represent an effective way to: *i)* obtain high coverage, and *ii)* capture cross-lingual associations between multiple lexical elements. This allows to overcome the bias towards single words featured by dictionaries and lexical databases.

As regards the other resources used for comparison, the results show that dictionaries substantially outperform MWN. This can be explained by the low coverage of MWN, whose entries also represent weaker semantic relations (preserving entailment, but with a lower probability to be applied) than the direct translations between terms contained in the dictionary.

Overall, our results suggest that the lexical knowledge extracted from parallel data can be successfully used to approach the CLTE task.

[8] http://www.dicts.info/
[9] http://en.wiktionary.org/
[10] http://www.omegawiki.org/
[11] http://www.wikipedia.org/

Dataset	WN	VO	WIKI	PPHT	PPHT 0.1	PPHT 0.2	PPHT 0.3	AVG
RTE3	61.88	62.00	61.75	62.88	**63.38**	**63.50**	63.00	62.37
RTE5	62.17	61.67	60.00	61.33	**62.50**	**62.67**	62.33	61.41
RTE3-G	62.62	61.5	60.5	62.88	**63.50**	62.00	61.5	-

Table 2: Accuracy results on monolingual RTE using different lexical resources.

5 Using parallel corpora for TE

This section addresses the third and the fourth research questions outlined in Section 1. Building on the positive results achieved on the cross-lingual scenario, we investigate the possibility to exploit bilingual parallel corpora in the traditional monolingual scenario. Using the same approach discussed in Section 4, we compare the results achieved with English paraphrase tables with those obtained with other widely used monolingual knowledge resources over two RTE datasets.

For the sake of completeness, we report in this section also the results obtained adopting the "basic solution" proposed by (Mehdad et al., 2010). Although it was presented as an approach to CLTE, the proposed method brings the problem back to the monolingual case by translating H into the language of T. The comparison with this method aims at verifying the real potential of parallel corpora against the use of a competitive MT system (Google Translate) in the same scenario.

5.1 Dataset

We experiment with the original RTE3 and RTE5 datasets, annotated with token, lemma, and stem information using the TreeTagger and the Snowball stemmer.

In addition to confront our method with the solution proposed by (Mehdad et al., 2010) we translated the Spanish hypotheses of our CLTE dataset into English using Google Translate. The resulting dataset was annotated in the same way.

5.2 Knowledge sources

We compared the results achieved with paraphrase tables (extracted with different pruning thresholds[12]) with those obtained using the three most widely used English resources for Textual Entailment (Bentivogli et al., 2010), namely:

WordNet (WN). WordNet 3.0 has been used to extract a set of 5396 pairs of words connected by the hyponymy and synonymy relations.

VerbOcean (VO). VerbOcean has been used to extract 18232 pairs of verbs connected by the "stronger-than" relation (*e.g. "kill" stronger-than "injure"*).

Wikipedia (WIKI). We performed Latent Semantic Analysis (LSA) over Wikipedia using the jLSI tool (Giuliano, 2007) to measure the relatedness between words in the dataset. Then, we filtered all the pairs with similarity lower than 0.7 as proposed by (Kouylekov et al., 2009). In this way we obtained 13760 word pairs.

5.3 Results and Discussion

Table 2 shows the accuracy results calculated over the original RTE3 and RTE5 test sets, training our classifier over the corresponding development sets.

The first two rows of the table show that pruned paraphrase tables always outperform the other lexical resources used for comparison, with an accuracy increase up to 3%. In particular, we observe that using 0.2 as a pruning threshold provides a good trade-off between coverage and precision, leading to our best results on both datasets (63.50% for RTE3, and 62.67% for RTE5). It's worth noting that these results, compared with the average scores reported by participants in the two editions of the RTE Challenge (AVG column), represent an accuracy improvement of more than 1%. Overall, these results confirm our claim that increasing the coverage using context sensitive phrase pairs obtained from large parallel corpora, results in better performance not only in CLTE,

[12]We pruned the paraphrase table (PPHT), with probabilities set to 0.1 (PPHT 0.1), 0.2 (PPHT 0.2), and 0.3 (PPHT 0.3)

but also in the monolingual scenario.

The comparison with the results achieved on monolingual data obtained by automatically translating the Spanish hypotheses (RTE3-G row in Table 2) leads to four main observations. First, we notice that dealing with MT-derived inputs, the optimal pruning threshold changes from 0.2 to 0.1, leading to the highest accuracy of 63.50%. This suggests that the noise introduced by incorrect translations can be tackled by increasing the coverage of the paraphrase table. Second, in line with the findings of (Mehdad et al., 2010), the results obtained over the MT-derived corpus are equal to those we achieve over the original RTE3 dataset (*i.e.* 63.50%). Third, the accuracy obtained over the CLTE corpus using combined phrase and paraphrase tables (62.88%, as reported in Table 1) is comparable to the best result gained over the automatically translated dataset (63.50%). In all the other cases, the use of phrase and paraphrase tables on CLTE data outperforms the results achieved on the same data after translation. Finally, it's worth remarking that applying our phrase matching method on the translated dataset without any additional source of knowledge would result in an overall accuracy of 62.12%, which is lower than the result obtained using only phrase tables on cross-lingual data (62.62%). This demonstrates that phrase tables can successfully replace MT systems in the CLTE task.

In light of this, we suggest that extracting lexical knowledge from parallel corpora is a preferable solution to approach CLTE. One of the main reasons is that placing a black-box MT system at the front-end of the entailment process reduces the possibility to cope with wrong translations. Furthermore, the access to MT components is not easy (*e.g.* Google Translate limits the number and the size of queries, while open source MT tools cover few language pairs). Moreover, the task of developing a full-fledged MT system often requires the availability of parallel corpora, and is much more complex than extracting lexical knowledge from them.

6 Conclusion and Future Work

In this paper we approached the cross-lingual Textual Entailment task focusing on the role of lexical knowledge extracted from bilingual parallel corpora. One of the main difficulties in CLTE raises from the lack of adequate knowledge resources to bridge the lexical gap between texts and hypotheses in different languages. Our approach builds on the intuition that the vast amount of knowledge that can be extracted from parallel data (in the form of phrase and paraphrase tables) offers a possible solution to the problem. To check the validity of our assumptions we carried out several experiments on an English-Spanish corpus derived from the RTE3 dataset, using phrasal matches as a criterion to approximate entailment. Our results show that phrase and paraphrase tables allow to: *i)* outperform the results achieved with the few multilingual lexical resources available, and *ii)* reach performance levels above the average scores obtained by participants in the monolingual RTE3 challenge. These improvements can be explained by the fact that the lexical knowledge extracted from parallel data provides good coverage both at the level of single words, and at the level of phrases.

As a further contribution, we explored the application of paraphrase tables extracted from parallel data in the traditional monolingual scenario. Contrasting results with those obtained with the most widely used resources in TE, we demonstrated the effectiveness of paraphrase tables as a mean to overcome the bias towards single words featured by the existing resources.

Our future work will address both the extraction of lexical information from bilingual parallel corpora, and its use for TE and CLTE. On one side, we plan to explore alternative ways to build phrase and paraphrase tables. One possible direction is to consider linguistically motivated approaches, such as the extraction of syntactic phrase tables as proposed by (Yamada and Knight, 2001). Another interesting direction is to investigate the potential of paraphrase patterns (*i.e.* patterns including part-of-speech slots), extracted from bilingual parallel corpora with the method proposed by (Zhao et al., 2009). On the other side we will investigate more sophisticated methods to exploit the acquired lexical knowledge. As a first step, the probability scores assigned to phrasal entries will be considered to perform weighted phrase matching as an improved criterion to approximate entailment.

Acknowledgments

This work has been partially supported by the EC-funded project CoSyne (FP7-ICT-4-24853).

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. *Proceedings of COLING-ACL.*

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with Bilingual Parallel Corpora. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005).*

Roy Bar-haim , Jonathan Berant , Ido Dagan , Iddo Greental , Shachar Mirkin , Eyal Shnarch , and Idan Szpektor. 2008. Efficient semantic deduction and approximate matching over compact parse forests. *Proceedings of the TAC 2008 Workshop on Textual Entailment*.

Luisa Bentivogli, Peter Clark, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2010. The Sixth PASCAL Recognizing Textual Entailment Challenge. *Proceedings of the the Text Analysis Conference (TAC 2010)*.

Timothy Chklovski and Patrick Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*.

Ido Dagan and Oren Glickman. 2004. Probabilistic textual entailment: Generic applied modeling of language variability. *Proceedings of the PASCAL Workshop of Learning Methods for Text Understanding and Mining*.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Journal of Natural Language Engineering* , Volume 15, Special Issue 04, pp i-xvii.

Michael Denkowski and Alon Lavie. 2010. Extending the METEOR Machine Translation Evaluation Metric to the Phrase Level. *Proceedings of Human Language Technologies (HLT-NAACL 2010).*

Georgiana Dinu and Rui Wang. 2009. Inference Rules and their Application to Recognizing Textual Entailment. *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009).*

Claudio Giuliano. 2007. jLSI a tool for latent semantic indexing. *Software available at http://tcc.itc.it/research/textec/tools-resources/jLSI.html.*

Lidija Iordanskaja, Richard Kittredge, and Alain Polg re.. 1991. Lexical selection and paraphrase in a meaning text generation model. *Natural Language Generation in Articial Intelligence and Computational Linguistics*.

Thorsten Joachims. 1999. Making large-scale support vector machine learning practical.

Philipp Koehn, Franz Josef Och, and Daniel Marcu 2003. Statistical Phrase-Based Translation. *Proceedings of HLT/NAACL*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Milen Kouleykov and Bernardo Magnini. 2005. Tree edit distance for textual entailment. *Proceedings of RALNP-2005, International Conference on Recent Advances in Natural Language Processing*.

Milen Kouylekov, Yashar Mehdad, and Matteo Negri. 2010. Mining Wikipedia for Large-Scale Repositories of Context-Sensitive Entailment Rules. *Proceedings of the Language Resources and Evaluation Conference (LREC 2010)*.

Yashar Mehdad, Alessandro Moschitti and Fabio Massimo Zanzotto. 2010. Syntactic/semantic structures for textual entailment recognition. *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2010)*.

Dekang Lin and Patrick Pantel. 2001. DIRT - Discovery of Inference Rules from Text.. *Proceedings of ACM Conference on Knowledge Discovery and Data Mining (KDD-01)*.

Kathleen R. McKeown, Regina Barzilay, David Evans, Vasileios Hatzivassiloglou, Judith L. Klavans, Ani Nenkova, Carl Sable, Barry Schiffman, and Sergey Sigelman. 2002. Tracking and summarizing news on a daily basis with Columbias Newsblaster. *Proceedings of the Human Language Technology Conference.*.

Yashar Mehdad, Matteo Negri, and Marcello Federico. 2010. Towards Cross-Lingual Textual Entailment. *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2010).*

Dan Moldovan and Adrian Novischi. 2002. Lexical chains for question answering. *Proceedings of COLING*.

Matteo Negri and Yashar Mehdad. 2010. Creating a Bilingual Entailment Corpus through Translations with Mechanical Turk: $100 for a 10-day Rush. *Proceedings of the NAACL 2010 Workshop on Creating Speech and Language Data With Amazons Mechanical Turk* .

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):1951.

Emanuele Pianta, Luisa Bentivogli, and Christian Girardi. 2002. MultiWordNet: Developing and Aligned Multilingual Database. *Proceedings of the First International Conference on Global WordNet*.

Vasile Rus, Art Graesser, and Kirtan Desai 2005. Lexico-Syntactic Subsumption for Textual Entailment. *Proceedings of RANLP 2005*.

Helmut Schmid 2005. Probabilistic Part-of-Speech Tagging Using Decision Trees. *Proceedings of the International Conference on New Methods in Language Processing*.

Marta Tatu andDan Moldovan. 2005. A semantic approach to recognizing textual entailment. *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*.

Matthew Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. 2009. Fluency, Adequacy, or HTER? Exploring Different Human Judgments with a Tunable MT Metric. *Proceedings of WMT09.*

Rui Wang and Yi Zhang,. 2009. Recognizing Textual Relatedness with Predicate-Argument Structures. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*.

Kenji Yamada and Kevin Knight 2001. A Syntax-Based Statistical Translation Model. *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Shiqi Zhao, Haifeng Wang, Ting Liu, and Sheng Li. 2009. Extracting Paraphrase Patterns from Bilingual Parallel Corpora. *Journal of Natural Language Engineering* , Volume 15, Special Issue 04, pp 503-526.

Using Large Monolingual and Bilingual Corpora to Improve Coordination Disambiguation

Shane Bergsma, David Yarowsky, Kenneth Church
Deptartment of Computer Science and Human Language Technology Center of Excellence
Johns Hopkins University
sbergsma@jhu.edu, yarowsky@cs.jhu.edu, kenneth.church@jhu.edu

Abstract

Resolving coordination ambiguity is a classic hard problem. This paper looks at coordination disambiguation in complex noun phrases (NPs). Parsers trained on the Penn Treebank are reporting impressive numbers these days, but they don't do very well on this problem (79%). We explore systems trained using three types of corpora: (1) annotated (e.g. the Penn Treebank), (2) bitexts (e.g. Europarl), and (3) unannotated monolingual (e.g. Google N-grams). Size matters: (1) is a million words, (2) is potentially billions of words and (3) is potentially trillions of words. The unannotated monolingual data is helpful when the ambiguity can be resolved through associations among the lexical items. The bilingual data is helpful when the ambiguity can be resolved by the order of words in the translation. We train separate classifiers with monolingual and bilingual features and iteratively improve them via co-training. The co-trained classifier achieves close to 96% accuracy on Treebank data and makes 20% fewer errors than a supervised system trained with Treebank annotations.

1 Introduction

Determining which words are being linked by a coordinating conjunction is a classic hard problem. Consider the pair:

+**ellipsis** rocket$\backslash w_1$ and mortar$\backslash w_2$ attacks$\backslash h$
−**ellipsis** asbestos$\backslash w_1$ and polyvinyl$\backslash w_2$ chloride$\backslash h$

+ellipsis is about both *rocket attacks* and *mortar attacks*, unlike −ellipsis which is not about *asbestos chloride*. We use *h* to refer to the head of the phrase, and w_1 and w_2 to refer to the other two lexical items.

Natural Language Processing applications need to recognize NP ellipsis in order to make sense of new sentences. For example, if an Internet search engine is given the phrase *rocket attacks* as a query, it should rank documents containing *rocket and mortar attacks* highly, even though *rocket* and *attacks* are not contiguous in the document. Furthermore, NPs with ellipsis often require a distinct type of reordering when translated into a foreign language. Since coordination is both complex and productive, parsers and machine translation (MT) systems cannot simply memorize the analysis of coordinate phrases from training text. We propose an approach to recognizing ellipsis that could benefit both MT and other NLP technology that relies on shallow or deep syntactic analysis.

While the general case of coordination is quite complicated, we focus on the special case of complex NPs. Errors in NP coordination typically account for the majority of parser coordination errors (Hogan, 2007). The information needed to resolve coordinate NP ambiguity cannot be derived from hand-annotated data, and we follow previous work in looking for new information sources to apply to this problem (Resnik, 1999; Nakov and Hearst, 2005; Rus et al., 2007; Pitler et al., 2010).

We first resolve coordinate NP ambiguity in a word-aligned parallel corpus. In bitexts, both monolingual and bilingual information can indicate NP structure. We create separate classifiers using monolingual and bilingual feature views. We train the two classifiers using co-training, iteratively improving the accuracy of one classifier by learning from the predictions of the other. Starting from only two

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1346–1355,
Portland, Oregon, June 19-24, 2011.

initial labeled examples, we are able to train a highly accurate classifier using only monolingual features. The monolingual classifier can then be used both within and beyond the aligned bitext. In particular, it achieves close to 96% accuracy on both bitext data and on out-of-domain examples in the Treebank.

2 Problem Definition and Related Tasks

Our system operates over a part-of-speech tagged input corpus. We attempt to resolve the ambiguity in all tag sequences matching the expression:

[DT|PRP$] (N.*|J.*) and [DT|PRP$] (N.*|J.*) N.*
e.g. [the] rocket$\backslash w_1$ and [the] mortar$\backslash w_2$ attacks$\backslash h$

Each example ends with a noun, h. Preceding h are a pair of possibly-conjoined words, w_1 and w_2, either nouns (*rocket* and *mortar*), adjectives, or a mix of the two. We allow determiners or possessive pronouns before w_1 and/or w_2. This pattern is very common. Depending on the domain, we find it in roughly one of every 10 to 20 sentences. We merge identical matches in our corpus into a single example for labeling. Roughly 38% of w_1,w_2 pairs are both adjectives, 26% are nouns, and 36% are mixed.

The task is to determine whether w_1 and w_2 are conjoined or not. When they are not conjoined, there are two cases: 1) w_1 is actually conjoined with w_2 h as a whole (e.g. *asbestos and polyvinyl chloride*), or 2) The conjunction links something higher up in the parse tree, as in, "farmers are getting older$\backslash w_1$ and younger$\backslash w_2$ people$\backslash h$ are reluctant to take up farming." Here, *and* links two separate clauses.

Our task is both narrower and broader than previous work. It is broader than previous approaches that have focused only on conjoined nouns (Resnik, 1999; Nakov and Hearst, 2005). Although pairs of adjectives are usually conjoined (and mixed tags are usually not), this is not always true, as in *older/younger* above. For comparison, we also state accuracy on the noun-only examples (§ 8).

Our task is more narrow than the task tackled by full-sentence parsers, but most parsers do not bracket NP-internal structure at all, since such structure is absent from the primary training corpus for statistical parsers, the Penn Treebank (Marcus et al., 1993). We confirm that standard broad-coverage parsers perform poorly on our task (§ 7).

Vadas and Curran (2007a) manually annotated NP structure in the Penn Treebank, and a few custom NP parsers have recently been developed using this data (Vadas and Curran, 2007b; Pitler et al., 2010). Our task is more narrow than the task handled by these parsers since we do not handle other, less-frequent and sometimes more complex constructions (e.g. *robot arms and legs*). However, such constructions are clearly amenable to our algorithm. In addition, these parsers have only evaluated coordination resolution *within base NPs*, simplifying the task and rendering the aforementioned *older/younger* problem moot. Finally, these custom parsers have only used simple count features; for example, they have not used the paraphrases we describe below.

3 Supervised Coordination Resolution

We adopt a discriminative approach to resolving coordinate NP ambiguity. For each unique coordinate NP in our corpus, we encode relevant information in a feature vector, $\bar{x}$. A classifier scores these vectors with a set of learned weights, $\bar{w}$. We assume N labeled examples $\{(y^1, \bar{x}^1), ..., (y^N, \bar{x}^N)\}$ are available to train the classifier. We use '$y = 1$' as the class label for NPs with ellipsis and '$y = 0$' for NPs without. Since our particular task requires a binary decision, any standard learning algorithm can be used to learn the feature weights on the training data. We use (regularized) logistic regression (a.k.a. maximum entropy) since it has been shown to perform well on a range of NLP tasks, and also because its probabilistic interpretation is useful for co-training (§ 4). In binary logistic regression, the probability of a positive class takes the form of the logistic function:

$$\Pr(y = 1) = \frac{\exp(\bar{w} \cdot \bar{x})}{1 + \exp(\bar{w} \cdot \bar{x})}$$

Ellipsis is predicted if $\Pr(y = 1) > 0.5$ (equivalently, $\bar{w} \cdot \bar{x} > 0$), otherwise we predict no ellipsis.

Supervised classifiers easily incorporate a range of interdependent information into a learned decision function. The cost for this flexibility is typically the need for labeled training data. The more features we use, the more labeled data we need, since for linear classifiers, the number of examples needed to reach optimum performance is at most linear in the

Phrase	Evidence		Pattern
dairy and meat production (ellipsis)	English:	*... production of dairy and meat...*	h *of* w_1 *and* w_2
	English:	*... dairy production and meat production...*	w_1 h *and* w_2 h
	English:	*... meat and dairy production...*	w_2 *and* w_1 h
	Spanish:	*... producción láctea y cárnica...*	h w_1 ... w_2
		→ *production dairy and meat*	
	Finnish:	*... maidon- ja lihantuotantoon...*	w_1- ... w_2h
		→ *dairy- and meatproduction*	
	French:	*... production de produits laitiers et de viande...*	h ... w_1 ... w_2
		→ *production of products dairy and of meat*	
asbestos and polyvinyl chloride (no ellipsis)	English:	*... polyvinyl chloride and asbestos...*	w_2 h *and* w_1
	English:	*... asbestos , and polyvinyl chloride...*	w_1 *,* *and* w_2 h
	English:	*... asbestos and chloride...*	w_1 *and* h
	Portuguese:	*... o amianto e o cloreto de polivinilo...*	w_1 ... h ... w_2
		→ *the asbestos and the chloride of polyvinyl*	
	Italian:	*... l' asbesto e il polivinilcloruro...*	w_1 ... w_2h
		→ *the asbestos and the polyvinylchloride*	

Table 1: Monolingual and bilingual evidence for ellipsis or lack-of-ellipsis in coordination of [w_1 *and* w_2 h] phrases.

number of features (Vapnik, 1998). In § 4, we propose a way to circumvent the need for labeled data.

We now describe the particular monolingual and bilingual information we use for this problem. We refer to Table 1 for canonical examples of the two classes and also to provide intuition for the features.

3.1 Monolingual Features

Count features These real-valued features encode the frequency, in a large auxiliary corpus, of relevant word sequences. Co-occurrence frequencies have long been used to resolve linguistic ambiguities (Dagan and Itai, 1990; Hindle and Rooth, 1993; Lauer, 1995). With the massive volumes of raw text now available, we can look for very specific and indicative word sequences. Consider the phrase *dairy and meat production* (Table 1). A high count in raw text for the paraphrase "*production of dairy and meat*" implies ellipsis in the original example. In the third column of Table 1, we suggest a pattern that generalizes the particular piece of evidence. It is these patterns and other English paraphrases that we encode in our count features (Table 2). We also use (but do not list) count features for the four paraphrases proposed in Nakov and Hearst (2005, § 3.2.3). Such specific paraphrases are more common than one might think. In our experiments, at least 20% of examples have non-zero counts for a 5-gram pattern, while over 70% of examples have counts for a 4-gram pattern.

Our features also include counts for subsequences of the full phrase. High counts for "*dairy production*" alone or just "*dairy and meat*" also indicate ellipsis. On the other hand, like Pitler et al. (2010), we have a feature for the count of "*dairy and production*." Frequent conjoining of w_1 and h is evidence that there is no ellipsis, that w_1 and h are compatible and heads of two separate and conjoined NPs.

Many of our patterns are novel in that they include commas or determiners. The presence of these often indicate that there are two separate NPs. E.g. seeing *asbestos , and polyvinyl chloride* or *the asbestos and the polyvinyl chloride* suggests no ellipsis. We also propose patterns that include left-and-right context around the NP. These aim to capture salient information about the NP's distribution as an entire unit. Finally, patterns involving prepositions look for explicit paraphrasing of the nominal relations; the presence of "h PREP w_1 and w_2" in a corpus would suggest ellipsis in the original NP.

In total, we have 48 separate count features, requiring counts for 315 distinct N-grams for each example. We use **log-counts** as the feature value, and use a separate binary feature to indicate if a particular count is zero. We efficiently acquire the counts using custom tools for managing web-scale N-gram

Real-valued count features. C(p) → count of p		
C(w_1)	C(w_2)	C(h)
C(w_1 CC w_2)	C(w_1 h)	C(w_2 h)
C(w_2 CC w_1)	C(w_1 CC h)	C(h CC w_1)
C(DT w_1 CC w_2)		C(w_1 , CC w_2)
C(DT w_2 CC w_1)		C(w_2 , CC w_1)
C(DT w_1 CC h)		C(w_1 CC w_2 ,)
C(DT h CC w_1)		C(w_2 CC w_1 ,)
C(DT w_1 and DT w_2)		C(w_1 CC DT w_2)
C(DT w_2 and DT w_1)		C(w_2 CC DT w_1)
C(DT h and DT w_1)		C(w_1 CC DT h)
C(DT h and DT w_2)		C(h CC DT w_1)
C($\langle$L-CTXT$_i\rangle$ w_1 and w_2 h)		C(w_1 CC w_2 h)
C(w_1 and w_2 h $\langle$R-CTXT$_i\rangle$)		C(h PREP w_1)
C(h PREP w_1 CC w_2)		C(h PREP w_2)
Count feature filler sets		
DT = {*the, a, an, its, his*}		CC = {*and, or, ','*}
PREP = {*of, for, in, at, on, from, with, about*}		
Binary features and feature templates → {0, 1}		
wrd$_1$=⟨wrd(w_1)⟩		tag$_1$=⟨tag(w_1)⟩
wrd$_2$=⟨wrd(w_2)⟩		tag$_2$=⟨tag(w_2)⟩
wrd$_h$=⟨wrd(h)⟩		tag$_h$=⟨tag(h)⟩
wrd$_{12}$=⟨wrd(w_1),wrd(w_2)⟩		wrd(w_1)=wrd(w_2)
tag$_{12}$=⟨tag(w_1),tag(w_2)⟩		tag(w_1)=tag(w_2)
tag$_{12h}$=⟨tag(w_1),tag(w_1),tag(h)⟩		

Table 2: Monolingual features. For counts using the filler sets CC, DT and PREP, counts are *summed* across all filler combinations. In contrast, feature templates are denoted with $\langle\cdot\rangle$, where the feature label depends on the ⟨bracketed argument⟩. E.g., we have separate count feature for each item in the L/R context sets, where {L-CTXT} = {*with, and, as, including, on, is, are, &*}, {R-CTXT} = {*and, have, of, on, said, to, were, &*}

data (§ 5). Previous approaches have used search engine page counts as substitutes for co-occurrence information (Nakov and Hearst, 2005; Rus et al., 2007). These approaches clearly cannot scale to use the wide range of information used in our system.

Binary features Table 2 gives the binary features and feature templates. These are templates in the sense that every unique word or tag fills the template and corresponds to a unique feature. We can thus learn if particular words or tags are associated with ellipsis. We also include binary features to flag the presence of any optional determiners before w_1 or w_2. We also have binary features for the context words that precede and follow the tag sequence in the source corpus. These context features are analogous to the L/R-CTXT features that were counted in the auxiliary corpus. Our classifier learns, for example, that instances preceded by the words *its* and *in* are likely to have ellipsis: these words tend to precede single NPs as opposed to conjoined NP pairs.

Monolingual: $\bar{x}_m$	Bilingual: $\bar{x}_b$
C(w_1):14.4	C(detl=h * w_1 * w_2),Dutch:1
C(w_2):15.4	C(detl=h * * w_1 * * w_2),Fr.:1
C(h):17.2	C(detl=h w_1 h * w_2),Greek:1
C(w_1 CC w_2):9.0	C(detl=h w_1 * w_2),Spanish:1
C(w_1 h):9.8	C(detl=w_1- * w_2h),Swedish:1
C(w_2 h):10.2	C(simp=h w_1 w_2),Dutch:1
C(w_2 CC w_1):10.5	C(simp=h w_1 w_2),French:1
C(w_1 CC h):3.5	C(simp=h w_1 h w_2),Greek:1
C(h CC w_1):6.8	C(simp=h w_1 w_2),Spanish:1
C(DT w_2 CC w_1:7.8	C(simp=w_1 w_2h),Swedish:1
C(w_1 and w_2 h *and*):2.4	C(span=5),Dutch:1
C(h PREP w_1 CC w_2):2.6	C(span=7),French:1
wrd$_1$=*dairy*:1	C(span=5),Greek:1
wrd$_2$=*meat*:1	C(span=4),Spanish:1
wrd$_h$=*production*:1	C(span=3),Swedish:1
tag$_1$=*NN*:1	C(ord=h w_1 w_2),Dutch:1
tag$_2$=*NN*:1	C(ord=h w_1 w_2),French:1
tag$_h$=*NN*:1	C(ord=h w_1 h w_2),Greek:1
wrd$_{12}$=*dairy,meat*:1	C(ord=h w_1 w_2),Spanish:1
tag$_{12}$=*NN,NN*:1	C(ord=w_1 w_2 h),Swedish:1
tag(w_1)=tag(w_2):1	C(ord=h w_1 w_2):4
tag$_{12h}$=*NN,NN,NN*:1	C(ord=w_1 w_2 h):1

Table 3: Example of actual instantiated feature vectors for *dairy and meat production* (in label:value format). Monolingual feature vector, $\bar{x}_m$, on the left (both count and binary features, see Table 2), Bilingual feature vector, $\bar{x}_b$, on the right (see Table 4).

Example Table 3 provides part of the actual instantiated monolingual feature vector for *dairy and meat production*. Note the count features have logarithmic values, while only the non-zero binary features are included.

A later stage of processing extracts a list of feature labels from the training data. This list is then used to map feature labels to integers, yielding the standard (sparse) format used by most machine learning software (e.g., 1:14.4 2:15.4 3:17.2 ... 7149:1 24208:1).

3.2 Bilingual Features

The above features represent the best of the information available to a coordinate NP classifier when operating on an arbitrary text. In some domains, however, we have additional information to inform our decisions. We consider the case where we seek to predict coordinate structure in parallel text: i.e., English text with a corresponding translation in one

or more target languages. A variety of mature NLP tools exists in this domain, allowing us to robustly align the parallel text first at the sentence and then at the word level. Given a word-aligned parallel corpus, we can see how the different types of coordinate NPs are translated in the target languages.

In Romance languages, examples with ellipsis, such as *dairy and meat production* (Table 1), tend to correspond to translations with the head in the first position, e.g. "producción láctea y cárnica" in Spanish (examples taken from Europarl (Koehn, 2005)). When there is no ellipsis, the head-first syntax leads to the "w_1 and h w_2" ordering, e.g. *amianto e o cloreto de polivinilo* in Portuguese. Another clue for ellipsis is the presence of a dangling hyphen, as in the Finnish *maidon- ja lihantuotantoon*. We find such hyphens especially common in Germanic languages like Dutch. In addition to language-specific clues, a translation may resolve an ambiguity by paraphrasing the example in the same way it may be paraphrased in English. E.g., we see *hard and soft drugs* translated into Spanish as *drogas blandas y drogas duras* with the head, *drogas*, repeated (akin to *soft drugs and hard drugs* in English).

One could imagine manually defining the relationship between English NP coordination and the patterns in each language, but this would need to be repeated for each language pair, and would likely miss many useful patterns. In contrast, by representing the translation patterns as features in a classifier, we can instead automatically learn the coordination-translation correspondences, in any language pair.

For each occurrence of a coordinate NP in a word-aligned bitext, we inspect the alignments and determine the mapping of w_1, w_2 and h. Recall that each of our examples represents all the occurrences of a unique coordinate NP in a corpus. We therefore aggregate translation information over all the occurrences. Since the alignments in automatically-aligned parallel text are noisy, the more occurrences we have, the more translations we have, and the more likely we are to make a correct decision. For some common instances in Europarl, like *Agriculture and Rural Development*, we have thousands of translations in several languages.

Table 4 provides the bilingual feature templates. The notation indicates that, for a given coordinate NP, we count the frequency of each translation pattern in each target language, and generate real-valued features for these counts. The feature counts are indexed to the particular pattern and language. We also have one language-independent feature, C$\langle$*ord*(w_1,w_2,h)$\rangle$, which gives the frequency of each ordering across all languages. The *span* is the number of tokens collectively spanned by the translations of w_1, w_2 and h. The "*detailed pattern*" represents the translation using wildcards for all other foreign words, but maintains punctuation. Letting '*' stand for the wildcard, the detailed patterns for the translations of *dairy and meat production* in Table 1 would be [h w_1 * w_2] (Spanish), [w_1- * $w_2 h$] (Finnish) and [h * * w_1 * * w_2] (French). Four or more consecutive wildcards are converted to '...'. For the "*simple pattern*," we remove the wildcards and punctuation. Note that our aligner allows the English word to map to multiple target words. The simple pattern differs from the *ordering* in that it denotes how many tokens each of w_1, w_2 and h span.

C$\langle$*detl*(w_1,w_2,h)$\rangle$,$\langle$LANG$\rangle$
C$\langle$*simp*(w_1,w_2,h)$\rangle$,$\langle$LANG$\rangle$
C$\langle$*span*(w_1,w_2,h)$\rangle$,$\langle$LANG$\rangle$
C$\langle$*ord*(w_1,w_2,h)$\rangle$,$\langle$LANG$\rangle$
C$\langle$*ord*(w_1,w_2,h)$\rangle$

Table 4: Real-valued bilingual feature templates. The shorthand is *detl*="detailed pattern," *simp*="simple pattern," *span*="span of pattern," *ord*="order of words." The notation C$\langle p \rangle$,$\langle$LANG$\rangle$ means the number of times we see the pattern (or span) $\langle p \rangle$ as the aligned translation of the coordinate NP in the target language $\langle$LANG$\rangle$.

Example Table 3 also provides part of the actual instantiated bilingual feature vector for *dairy and meat production*.

4 Bilingual Co-training

We exploit the orthogonality of the monolingual and bilingual features using semi-supervised learning. These features are orthogonal in the sense that they look at different sources of information for each example. If we had enough training data, a good classifier could be trained using either monolingual or bilingual features on their own. With classifiers trained on even a little labeled data, it's feasible that for a particular example, the monolingual classifier might be confident when the bilingual classifier is

Algorithm 1 The bilingual co-training algorithm: subscript m corresponds to monolingual, b to bilingual

Given: • a set L of labeled training examples in the bitext, $\{(\bar{x}^i, y^i)\}$
• a set U of unlabeled examples in the bitext, $\{\bar{x}^j\}$
• hyperparams: k (num. iterations), u_m and u_b (size smaller unlabeled pools), n_m and n_b (num. new labeled examples each iteration), C: regularization param. for classifier training

Create $L_m \leftarrow L$
Create $L_b \leftarrow L$
Create a pool U_m by choosing u_m examples randomly from U.
Create a pool U_b by choosing u_b examples randomly from U.
for $i = 0$ to k **do**
  Use L_m to train a classifier h_m using only $\bar{x}_m$, the monolingual features of $\bar{x}$
  Use L_b to train a classifier h_b using only $\bar{x}_b$, the bilingual features of $\bar{x}$
  Use h_m to label U_m, move the n_m most-confident examples to L_b
  Use h_b to label U_b, move the n_b most-confident examples to L_m
  Replenish U_m and U_b randomly from U with n_m and n_b new examples
end for

uncertain, and vice versa. This suggests using a co-training approach (Yarowsky, 1995; Blum and Mitchell, 1998). We train separate classifiers on the labeled data. We use the predictions of one classifier to label new examples for training the orthogonal classifier. We iterate this training and labeling.

We outline how this procedure can be applied to bitext data in **Algorithm 1** (above). We follow prior work in drawing predictions from smaller pools, U_m and U_b, rather than from U itself, to ensure the labeled examples "are more representative of the underlying distribution" (Blum and Mitchell, 1998). We use a logistic regression classifier for h_m and h_b. Like Blum and Mitchell (1998), we also create a *combined* classifier by making predictions according to $\text{argmax}_{y=1,0}\, Pr(y|x_m)Pr(y|x_b)$.

The hyperparameters of the algorithm are 1) k, the number of iterations, 2) u_m and u_b, the size of the smaller unlabeled pools, 3) n_m and n_b, the number of new labeled examples to include at each iteration, and 4) the regularization parameter of the logistic regression classifier. All such parameters can be tuned on a development set. Like Blum and Mitchell (1998), we ensure that we maintain roughly the true class balance in the labeled examples added at each iteration; we also estimate this balance using development data.

There are some differences between our approach and the co-training algorithm presented in Blum and Mitchell (1998, Table 1). One of our key goals is to produce an accurate classifier that uses only monolingual features, since only this classifier can be applied to arbitrary monolingual text. We thus break the symmetry in the original algorithm and allow h_b to label more examples for h_m than vice versa, so that h_m will improve faster. This is desirable because we don't have unlimited unlabeled examples to draw from, only those found in our parallel text.

5 Data

Web-scale text data is used for monolingual feature counts, parallel text is used for classifier co-training, and labeled data is used for training and evaluation.

Web-scale N-gram Data We extract our counts from *Google V2*: a new N-gram corpus (with N-grams of length one-to-five) created from the same one-trillion-word snapshot of the web as the Google 5-gram Corpus (Brants and Franz, 2006), but with enhanced filtering and processing of the source text (Lin et al., 2010, Section 5). We get counts using the suffix array tools described in (Lin et al., 2010). We add one to all counts for smoothing.

Parallel Data We use the Danish, German, Greek, Spanish, Finnish, French, Italian, Dutch, Portuguese, and Swedish portions of Europarl (Koehn, 2005). We also use the Czech, German, Spanish and French news commentary data from WMT

2010.[1] Word-aligned English-Foreign bitexts are created using the Berkeley aligner.[2] We run 5 iterations of joint IBM Model 1 training, followed by 3-to-5 iterations of joint HMM training, and align with the competitive-thresholding heuristic. The English portions of all bitexts are part-of-speech tagged with CRFTagger (Phan, 2006). 94K unique coordinate NPs and their translations are then extracted.

Labeled Data For experiments within the parallel text, we manually labeled 1320 of the 94K coordinate NP examples. We use 605 examples to set development parameters, 607 examples as held-out test data, and 2, 10 or 100 examples for training.

For experiments on the WSJ portion of the Penn Treebank, we merge the original Treebank annotations with the NP annotations provided by Vadas and Curran (2007a). We collect all coordinate NP sequences matching our pattern and collapse them into a single example. We label these instances by determining whether the annotations have w_1 and w_2 conjoined. In only one case did the same coordinate NP have different labels in different occurrences; this was clearly an error and resolved accordingly. We collected 1777 coordinate NPs in total, and divided them into 777 examples for training, 500 for development and 500 as a final held-out test set.

6 Evaluation and Settings

We evaluate using *accuracy*: the percentage of examples classified correctly in held-out test data. We compare our systems to a baseline referred to as the **Tag-Triple classifier**. This classifier has a single feature: the $\text{tag}(w_1), \text{tag}(w_2), \text{tag}(h)$ triple. Tag-Triple is therefore essentially a discriminative, *un*lexicalized parser for our coordinate NPs.

All classifiers use L2-regularized logistic regression training via LIBLINEAR (Fan et al., 2008). For co-training, we fix regularization at $C = 0.1$. For all other classifiers, we optimize the C parameter on the development data. At each iteration, i, classifier h_m annotates 50 new examples for training h_b, from a pool of 750 examples, while h_b annotates $50 * i$ new examples for h_m, from a pool of $750 * i$ examples. This ensures h_m gets the majority of automatically-labeled examples.

[1] www.statmt.org/wmt10/translation-task.html
[2] nlp.cs.berkeley.edu/pages/wordaligner.html

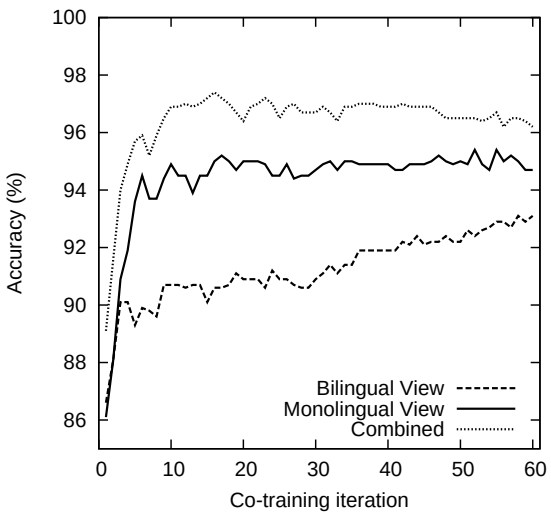


Figure 1: Accuracy on **Bitext** development data over the course of co-training (from 10 initial seed examples).

We also set k, the number of co-training iterations. The monolingual, bilingual, and combined classifiers reach their optimum levels of performance after different numbers of iterations (Figure 1). We therefore set k separately for each, stopping around 16 iterations for the combined, 51 for the monolingual, and 57 for the bilingual classifier.

7 Bitext Experiments

We evaluate our systems on our held-out bitext data. The majority class is ellipsis, in 55.8% of examples. For comparison, we ran two publicly-available broad-coverage parsers and analyzed whether they correctly predicted ellipsis. The parsers were the C&C parser (Curran et al., 2007) and Minipar (Lin, 1998). They achieved 78.6% and 77.6%.[3]

Table 5 shows that co-training results in much more accurate classifiers than supervised training alone, regardless of the features or amount of initial training data. The Tag-Triple system is the weakest system in all cases. This shows that better monolingual features are very important, but semi-supervised training can also make a big difference.

[3] We provided the parsers full sentences containing the NPs. We directly extracted the labels from the C&C bracketing, while for Minipar we checked whether w_1 was the head of w_2. Of course, the parsers performed very poorly on ellipsis involving two nouns (partly because NP structure is absent from their training corpora (see § 2 and also Vadas and Curran (2008)), but neither exceeded 88% on adjective or mixed pairs either.

System	# of Examples		
	2	10	100
Tag-Triple classifier	67.4	79.1	82.9
Monolingual classifier	69.9	90.8	91.6
Co-trained Mono. classifier	96.4	95.9	96.0
Relative error reduction via co-training	*88%*	*62%*	*52%*
Bilingual classifier	76.8	85.5	92.1
Co-trained Bili. classifier	93.2	93.2	93.9
Relative error reduction via co-training	*71%*	*53%*	*23%*
Mono.+Bili. classifier	69.9	91.4	94.9
Co-trained Combo classifier	**96.7**	**96.7**	**96.7**
Relative error reduction via co-training	*89%*	*62%*	*35%*

Table 5: Co-training improves accuracy (%) over standard supervised learning on **Bitext** test data for different feature types and number of training examples.

System	Accuracy	Δ
Monolingual alone	91.6	-
+ Bilingual	94.9	39%
+ Co-training	96.0	54%
+ Bilingual & Co-training	**96.7**	**61%**

Table 6: Net benefits of bilingual features and co-training on **Bitext** data, 100-training-example setting. Δ = relative error reduction over Monolingual alone.

Table 6 shows the net benefit of our main contributions. Bilingual features clearly help on this task, but not as much as co-training. With bilingual features and co-training together, we achieve 96.7% accuracy. This combined system could be used to very accurately resolve coordinate ambiguity in parallel data prior to training an MT system.

8 WSJ Experiments

While we can now accurately resolve coordinate NP ambiguity in parallel text, it would be even better if this accuracy carried over to new domains, where bilingual features are not available. We test the robustness of our co-trained monolingual classifier by evaluating it on our labeled WSJ data.

The Penn Treebank and the annotations added by Vadas and Curran (2007a) comprise a very special corpus; such data is clearly not available in every domain. We can take advantage of the plentiful labeled examples to also test how our co-trained system compares to supervised systems trained with in-domain labeled examples, and also other systems, like Nakov and Hearst (2005), which although unsupervised, are tuned on WSJ data.

System	Training		WSJ Acc.	
	Set	#	Nouns	All
Nakov & Hearst	-	-	79.2	84.8
Tag-Triple	WSJ	777	76.1	82.4
Pitler et al.	WSJ	777	92.3	92.8
MonoWSJ	WSJ	777	92.3	94.4
Co-trained	Bitext	2	**93.8**	**95.6**

Table 7: Coordinate resolution accuracy (%) on **WSJ**.

We reimplemented Nakov and Hearst (2005)[4] and Pitler et al. (2010)[5] and trained the latter on WSJ annotations. We compare these systems to Tag-Triple and also to a supervised system trained on the WSJ using only our monolingual features (MonoWSJ). The (out-of-domain) bitext co-trained system is the best system on the WSJ data, both on just the examples where w_1 and w_2 are nouns (Nouns), and on all examples (All) (Table 7).[6] It is statistically significantly better than the prior state-of-the-art Pitler et al. system (McNemar's test, $p<0.05$) and also exceeds the WSJ-trained system using monolingual features ($p<0.2$). This domain robustness is less surprising given its key features are derived from web-scale N-gram data; such features are known to generalize well across domains (Bergsma et al., 2010). We tried co-training without the N-gram features, and performance was worse on the WSJ (85%) than supervised training on WSJ data alone (87%).

9 Related Work

Bilingual data has been used to resolve a range of ambiguities, from PP-attachment (Schwartz et al., 2003; Fossum and Knight, 2008), to distinguishing grammatical roles (Schwarck et al., 2010), to full dependency parsing (Huang et al., 2009). Related

[4]Nakov and Hearst (2005) use an unsupervised algorithm that predicts ellipsis on the basis of a majority vote over a number of pattern counts and established heuristics.

[5]Pitler et al. (2010) uses a supervised classifier to predict bracketings; their count and binary features are a strict subset of the features used in our Monolingual classifier.

[6]For co-training, we tuned k on the WSJ dev set but left other parameters the same. We start from 2 training instances; results were the same or slightly better with 10 or 100 instances.

work has also focused on projecting syntactic annotations from one language to another (Yarowsky and Ngai, 2001; Hwa et al., 2005), and jointly parsing the two sides of a bitext by leveraging the alignments during training and testing (Smith and Smith, 2004; Burkett and Klein, 2008) or just during training (Snyder et al., 2009). None of this work has focused on coordination, nor has it combined bitexts with web-scale monolingual information.

Most prior work has focused on leveraging the alignments between a single pair of languages. Dagan et al. (1991) first articulated the need for "a multilingual corpora based system, which exploits the differences between languages to automatically acquire knowledge about word senses." Kuhn (2004) used alignments across several Europarl bitexts to devise rules for identifying parse distituents. Bannard and Callison-Burch (2005) used multiple bitexts as part of a system for extracting paraphrases.

Our co-training algorithm is well suited to using multiple bitexts because it automatically learns the value of alignment information in each language. In addition, our approach copes with noisy alignments both by aggregating information across languages (and repeated occurrences within a language), and by only selecting the most confident examples at each iteration. Burkett et al. (2010) also proposed exploiting monolingual-view and bilingual-view predictors. In their work, the bilingual view encodes the per-instance *agreement* between monolingual predictors in two languages, while our bilingual view encodes the alignment and target text together, across multiple instances and languages.

The other side of the coin is the use of syntax to perform better translation (Wu, 1997). This is a rich field of research with its own annual workshop (Syntax and Structure in Translation).

Our monolingual model is most similar to previous work using counts from web-scale text, both for resolving coordination ambiguity (Nakov and Hearst, 2005; Rus et al., 2007; Pitler et al., 2010), and for syntax and semantics in general (Lapata and Keller, 2005; Bergsma et al., 2010). We do not currently use semantic similarity (either taxonomic (Resnik, 1999) or distributional (Hogan, 2007)) which has previously been found useful for coordination. Our model can easily include such information as additional features. Adding new features without adding new training data is often problematic, but is promising in our framework, since the bitexts provide so much indirect supervision.

10 Conclusion

Resolving coordination ambiguity is hard. Parsers are reporting impressive numbers these days, but coordination remains an area with room for improvement. We focused on a specific subcase, complex NPs, and introduced a new evaluation set. We achieved a huge performance improvement from 79% for state-of-the-art parsers to 96%.[7]

Size matters. Most parsers are trained on a mere million words of the Penn Treebank. In this work, we show how to take advantage of billions of words of bitexts and trillions of words of unlabeled monolingual text. Larger corpora make it possible to use associations among lexical items (compare *dairy production* vs. *asbestos chloride*) and precise paraphrases (*production of dairy and meat*). Bitexts are helpful when the ambiguity can be resolved by some feature in another language (such as word order).

The Treebank is convenient for supervised training because it has annotations. We show that even without such annotations, high-quality supervised models can be trained using co-training and features derived from huge volumes of unlabeled data.

References

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proc. ACL*, pages 597–604.

Shane Bergsma, Emily Pitler, and Dekang Lin. 2010. Creating robust supervised classifiers via web-scale n-gram data. In *Proc. ACL*, pages 865–874.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proc. COLT*, pages 92–100.

Thorsten Brants and Alex Franz. 2006. The Google Web 1T 5-gram Corpus Version 1.1. LDC2006T13.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proc. EMNLP*, pages 877–886.

David Burkett, Slav Petrov, John Blitzer, and Dan Klein. 2010. Learning better monolingual models with unannotated bilingual text. In *Proc. CoNLL*, pages 46–53.

[7] Evaluation scripts and data are available online: `www.clsp.jhu.edu/~sbergsma/coordNP.ACL11.zip`

James Curran, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale NLP with C&C and Boxer. In *Proc. ACL Demo and Poster Sessions*, pages 33–36.

Ido Dagan and Alan Itai. 1990. Automatic processing of large corpora for the resolution of anaphora references. In *Proc. COLING*, pages 330–332.

Ido Dagan, Alon Itai, and Ulrike Schwall. 1991. Two languages are more informative than one. In *Proc. ACL*, pages 130–137.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874.

Victoria Fossum and Kevin Knight. 2008. Using bilingual Chinese-English word alignments to resolve PP-attachment ambiguity in English. In *Proc. AMTA Student Workshop*, pages 48–53.

Donald Hindle and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 19(1):103–120.

Deirdre Hogan. 2007. Coordinate noun phrase disambiguation in a generative parsing model. In *Proc. ACL*, pages 680–687.

Liang Huang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce parsing. In *Proc. EMNLP*, pages 1222–1231.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11(3):311–325.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. MT Summit X*.

Jonas Kuhn. 2004. Experiments in parallel-text based grammar induction. In *Proc. ACL*, pages 470–477.

Mirella Lapata and Frank Keller. 2005. Web-based models for natural language processing. *ACM Trans. Speech and Language Processing*, 2(1):1–31.

Mark Lauer. 1995. Corpus statistics meet the noun compound: Some empirical results. In *Proc. ACL*, pages 47–54.

Dekang Lin, Kenneth Church, Heng Ji, Satoshi Sekine, David Yarowsky, Shane Bergsma, Kailash Patil, Emily Pitler, Rachel Lathbury, Vikram Rao, Kapil Dalwani, and Sushant Narsale. 2010. New tools for web-scale N-grams. In *Proc. LREC*.

Dekang Lin. 1998. Dependency-based evaluation of MINIPAR. In *Proc. LREC Workshop on the Evaluation of Parsing Systems*.

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

Preslav Nakov and Marti Hearst. 2005. Using the web as an implicit training set: application to structural ambiguity resolution. In *Proc. HLT-EMNLP*, pages 17–24.

Xuan-Hieu Phan. 2006. CRFTagger: CRF English POS Tagger. `crftagger.sourceforge.net`.

Emily Pitler, Shane Bergsma, Dekang Lin, and Kenneth Church. 2010. Using web-scale N-grams to improve base NP parsing performance. In *In Proc. COLING*, pages 886–894.

Philip Resnik. 1999. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130.

Vasile Rus, Sireesha Ravi, Mihai C. Lintean, and Philip M. McCarthy. 2007. Unsupervised method for parsing coordinated base noun phrases. In *Proc. CICLing*, pages 229–240.

Florian Schwarck, Alexander Fraser, and Hinrich Schütze. 2010. Bitext-based resolution of German subject-object ambiguities. In *Proc. HLT-NAACL*, pages 737–740.

Lee Schwartz, Takako Aikawa, and Chris Quirk. 2003. Disambiguation of English PP attachment using multilingual aligned data. In *Proc. MT Summit IX*, pages 330–337.

David A. Smith and Noah A. Smith. 2004. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proc. EMNLP*, pages 49–56.

Benjamin Snyder, Tahira Naseem, and Regina Barzilay. 2009. Unsupervised multilingual grammar induction. In *Proc. ACL-IJCNLP*, pages 1041–1050.

David Vadas and James R. Curran. 2007a. Adding noun phrase structure to the Penn Treebank. In *Proc. ACL*, pages 240–247.

David Vadas and James R. Curran. 2007b. Large-scale supervised models for noun phrase bracketing. In *PACLING*, pages 104–112.

David Vadas and James R. Curran. 2008. Parsing noun phrase structure with CCG. In *Proc. ACL*, pages 104–112.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley & Sons.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proc. NAACL*, pages 1–8.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. ACL*, pages 189–196.

Unsupervised Decomposition of a Document into Authorial Components

Moshe Koppel **Navot Akiva**
Dept. of Computer Science
Bar-Ilan University
Ramat Gan, Israel
{moishk,navot.akiva}@gmail.com

Idan Dershowitz
Dept. of Bible
Hebrew University
Jerusalem, Israel
dershowitz@gmail.com

Nachum Dershowitz
School of Computer Science
Tel Aviv University
Ramat Aviv, Israel
nachumd@tau.ac.il

Abstract

We propose a novel unsupervised method for separating out distinct authorial components of a document. In particular, we show that, given a book artificially "munged" from two thematically similar biblical books, we can separate out the two constituent books almost perfectly. This allows us to automatically recapitulate many conclusions reached by Bible scholars over centuries of research. One of the key elements of our method is exploitation of differences in synonym choice by different authors.

1 Introduction

We propose a novel unsupervised method for separating out distinct authorial components of a document.

There are many instances in which one is faced with a multi-author document and wishes to delineate the contributions of each author. Perhaps the most salient example is that of documents of historical significance that appear to be composites of multiple earlier texts. The challenge for literary scholars is to tease apart the document's various components. More contemporary examples include analysis of collaborative online works in which one might wish to identify the contribution of a particular author for commercial or forensic purposes.

We treat two versions of the problem. In the first, easier, version, the document to be decomposed is given to us segmented into units, each of which is the work of a single author. The challenge is only to cluster the units according to author. In the second version, we are given an unsegmented document and the challenge includes segmenting the document as well as clustering the resulting units.

We assume here that no information about the authors of the document is available and that in particular we are not supplied with any identified samples of any author's writing. Thus, our methods must be entirely unsupervised.

There is surprisingly little literature on this problem, despite its importance. Some work in this direction has been done on intrinsic plagiarism detection (e.g., Meyer zu Eisen 2006) and document outlier detection (e.g., Guthrie et al. 2008), but this work makes the simplifying assumption that there is a single dominant author, so that outlier units can be identified as those that deviate from the document as a whole. We don't make this simplifying assumption. Some work on a problem that is more similar to ours was done by Graham et al. (2005). However, they assume that examples of pairs of paragraphs labeled as same-author/different-author are available for use as the basis of supervised learning. We make no such assumption.

The obvious approach to our unsupervised version of the problem would be to segment the text (if necessary), represent each of the resulting units of text as a bag-of-words, and then use clustering algorithms to find natural clusters. We will see, however, that this naïve method is quite inadequate. Instead, we exploit a method favored by the literary scholar, namely, the use of synonym choice. Synonym choice proves to be far more useful for authorial decomposition than ordinary lexical features. However, synonyms are relatively

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1356–1364,
Portland, Oregon, June 19-24, 2011.

sparse and hence, though reliable, they are not comprehensive; that is, they are useful for separating out some units but not all. Thus, we use a two-stage process: first find a reliable partial clustering based on synonym usage and then use these as the basis for supervised learning using a different feature set, such as bag-of-words.

We use biblical books as our testbed. We do this for two reasons. First, this testbed is well motivated, since scholars have been doing authorial analysis of biblical literature for centuries. Second, precisely because it is of great interest, the Bible has been manually tagged in a variety of ways that are extremely useful for our method.

Our main result is that given artificial books constructed by randomly "munging" together actual biblical books, we are able to separate out authorial components with extremely high accuracy, even when the components are thematically similar. Moreover, our automated methods recapitulate many of the results of extensive manual research in authorial analysis of biblical literature.

The structure of the paper is as follows. In the next section, we briefly review essential information regarding our biblical testbed. In Section 3, we introduce a naïve method for separating components and demonstrate its inadequacy. In Section 4, we introduce the synonym method, in Section 5 we extend it to the two-stage method, and in Section 6, we offer systematic empirical results to validate the method. In Section 7, we extend our method to handle documents that have not been pre-segmented and present more empirical results. In Section 8, we suggest conclusions, including some implications for Bible scholarship.

2 The Bible as Testbed

While the biblical canon differs across religions and denominations, the common denominator consists of twenty-odd books and several shorter works, ranging in length from tens to thousands of verses. These works vary significantly in genre, and include historical narrative, law, prophecy, and wisdom literature. Some of these books are regarded by scholars as largely the product of a single author's work, while others are thought to be composites in which multiple authors are well-represented – authors who in some cases lived in widely disparate periods. In this paper, we will focus exclusively on the Hebrew books of the Bible, and we will work with the original untranslated texts.

The first five books of the Bible, collectively known as the Pentateuch, are the subject of much controversy. According to the predominant Jewish and Christian traditions, the five books were written by a single author – Moses. Nevertheless, scholars have found in the Pentateuch what they believe are distinct narrative and stylistic threads corresponding to multiple authors.

Until now, the work of analyzing composite texts has been done in mostly impressionistic fashion, whereby each scholar attempts to detect the telltale signs of multiple authorship and compilation. Some work on biblical authorship problems within a computational framework has been attempted, but does not handle our problem. Much earlier work (for example, Radday 1970; Bee 1971; Holmes 1994) uses multivariate analysis to test whether the clusters in a given clustering of some biblical text are sufficiently distinct to be regarded as probably a composite text. By contrast, our aim is to find the optimal clustering of a document, given that it is composite. Crucially, unlike that earlier work, we empirically prove the efficacy of our methods by testing it against known ground truth. Other computational work on biblical authorship problems (Mealand 1995; Berryman et al. 2003) involves supervised learning problems where some disputed text is to be attributed to one of a set of known authors. The supervised authorship attribution problem has been well-researched (for surveys, see Juola (2008), Koppel et al. (2009) and Stamatatos (2009)), but it is quite distinct from the unsupervised problem we consider here.

Since our problem has been dealt with almost exclusively using heuristic methods, the subjective nature of such research has left much room for debate. We propose to set this work on a firm algorithmic basis by identifying an optimal stylistic subdivision of the text. We do not concern ourselves with how or why such distinct threads exist. Those for whom it is a matter of faith that the Pentateuch is not a composition of multiple writers can view the distinction investigated here as that of multiple styles.

3 A Naïve Algorithm

For expository purposes, we will use a canonical example to motivate and illustrate each of a

sequence of increasingly sophisticated algorithms for solving the decomposition problem. Jeremiah and Ezekiel are two roughly contemporaneous books belonging to the same biblical sub-genre (prophetic works), and each is widely thought to consist primarily of the work of a single distinct author. Jeremiah consists of 52 chapters and Ezekiel consists of 48 chapters. For our first challenge, we are given all 100 unlabeled chapters and our task is to separate them out into the two constituent books. (For simplicity, let's assume that it is known that there are exactly two natural clusters.) Note that this is a pre-segmented version of the problem since we know that each chapter belongs to only one of the books.

As a first try, the basics of which will serve as a foundation for more sophisticated attempts, we do the following:

1. Represent each chapter as a bag-of-words (using all words that appear at least k times in the corpus).
2. Compute the similarity of every pair of chapters in the corpus.
3. Use a clustering algorithm to cluster the chapters into two clusters.

We use k=2, cosine similarity and ncut clustering (Dhillon et al. 2004). Comparing the Jeremiah-Ezekiel split to the clusters thus obtained, we have the following matrix:

Book	Cluster I	Cluster II
Jer	29	23
Eze	28	20

As can be seen, the clusters are essentially orthogonal to the Jeremiah-Ezekiel split. Ideally, 100% of the chapters would lie on the majority diagonal, but in fact only 51% do. Formally, our measure of correspondence between the desired clustering and the actual one is computed by first normalizing rows and then computing the weight of the majority diagonal relative to the whole. This measure, which we call normalized majority diagonal (NMD), runs from 50% (when the clusters are completely orthogonal to the desired split) to 100% (where the clusters are identical with the desired split). NMD is equivalent to maximal macro-averaged recall where the maximum is taken over the (two) possible assignments of books to clusters. In this case, we obtain an NMD of 51.5%, barely above the theoretical minimum.

This negative result is not especially surprising since there are many ways for the chapters to split (e.g., according to thematic elements, sub-genre, etc.) and we can't expect an unsupervised method to read our minds. Thus, to guide the method in the direction of stylistic elements that might distinguish between Jeremiah and Ezekiel, we define a class of generic biblical words consisting of all 223 words that appear at least five times in each of ten different books of the Bible.

Repeating our experiment of above, though limiting our feature set to generic biblical words, we obtain the following matrix:

Book	Cluster I	Cluster II
Jer	32	20
Eze	28	20

As can be seen, using generic words yields NMD of 51.3%, which does not improve matters at all. Thus, we need to try a different approach.

4 Exploiting Synonym Usage

One of the key features used by Bible scholars to classify different components of biblical literature is synonym choice. The underlying hypothesis is that different authorial components are likely to differ in the proportions with which alternative words from a set of synonyms (synset) are used. This hypothesis played a part in the pioneering work of Astruc (1753) on the book of Genesis – using a single synset: divine names – and has been refined by many others using broader feature sets, such as that of Carpenter and Hartford-Battersby (1900). More recently, the synonym hypothesis has been used in computational work on authorship attribution of English texts in the work of Clark and Hannon (2007) and Koppel et al. (2006).

This approach presents several technical challenges. First, ideally – in the absence of a sufficiently comprehensive thesaurus – we would wish to identify synonyms in an automated fashion. Second, we need to adapt our similarity measure for reasons that will be made clear below.

4.1 (Almost) Automatic Synset Identification

One of the advantages of using biblical literature is the availability of a great deal of manual annotation. In particular, we are able to identify synsets by exploiting the availability of the standard King James translation of the Bible into Eng-

lish (KJV). Conveniently, and unlike most modern translations, KJV almost invariably translates synonyms identically. Thus, we can generally identify synonyms by considering the translated version of the text. There are two points we need to be precise about. First, it is not actually words that we regard as synonymous, but rather word roots. Second, to be even more precise, it is not quite roots that are synonymous, but rather senses of roots. Conveniently, Strong's (1890 [2010]) Concordance lists every occurrence of each sense of each root that appears in the Bible separately (where senses are distinguished in accordance with the KJV translation). Thus, we can exploit KJV and the concordance to automatically identify synsets as well as occurrences of the respective synonyms in a synset.[1] (The above notwithstanding, there is still a need for a bit of manual intervention: due to polysemy in English, false synsets are occasionally created when two non-synonymous Hebrew words are translated into two senses of the same English word. Although this could probably be handled automatically, we found it more convenient to do a manual pass over the raw synsets and eliminate the problems.)

The above procedure yields a set of 529 synsets including a total of 1595 individual synonyms. Most synsets consist of only two synonyms, but some include many more. For example, there are 7 Hebrew synonyms corresponding to "fear".

4.2 Adapting the Similarity Measure

Let's now represent a unit of text as a vector in the following way. Each entry represents a synonym in one of the synsets. If none of the synonyms in a synset appear in the unit, all their corresponding entries are 0. If j different synonyms in a synset appear in the unit, then each corresponding entry is 1/j and the rest are 0. Thus, in the typical case where exactly one of the synonyms in a synset appears, its corresponding entry in the vector is 1 and the rest are 0.

Now we wish to measure the similarity of two such vectors. The usual cosine measure doesn't capture what we want for the following reason. If the two units use different members of a synset, cosine is diminished; if they use the same members of a synset, cosine is increased. So far, so good. But suppose one unit uses a particular synonym and the other doesn't use any member of that synset. This should teach us nothing about the similarity of the two units, since it reflects only on the relevance of the synset to the content of that unit; it says nothing about which synonym is chosen when the synset is relevant. Nevertheless, in this case, cosine would be diminished.

The required adaptation is as follows: we first eliminate from the representation any synsets that do not appear in both units (where a synset is said to appear in a unit if any of its constituent synonyms appear in the unit). We then compute cosine of the truncated vectors. Formally, for a unit x represented in terms of synonyms, our new similarity measure is $\cos'(x,y) = \cos(x|_{S(x \cap y)}, y|_{S(x \cap y)})$, where $x|_{S(x \cap y)}$ is the projection of x onto the synsets that appear in both x and y.

4.3 Clustering Jeremiah-Ezekiel Using Synonyms

We now apply ncut clustering to the similarity matrix computed as described above. We obtain the following split:

Book	Cluster I	Cluster II
Jer	48	4
Eze	5	43

Clearly, this is quite a bit better than results obtained using simple lexical features as described above. Intuition for why this works can be purchased by considering concrete examples. There are two Hebrew synonyms – *pēʾâh* and *miqṣôaʿ* corresponding to the word "corner", two (*minḥâh* and *tĕrûmâh*) corresponding to the word "oblation", and two (*nāṭaʿ* and *šāṯal*) corresponding to the word "planted". We find that *pēʾâh*, *minḥâh* and *nāṭaʿ* tend to be located in the same units and, concomitantly, *miqṣôaʿ*, *tĕrûmâh* and *šāṯal* are located in the same units. Conveniently, the former are all Jeremiah and the latter are all Ezekiel.

While the above result is far better than those obtained using more naïve feature sets, it is, nevertheless, far from perfect. We have, however, one more trick at our disposal that will improve these results further.

5 Combining Partial Clustering and Supervised Learning

Analysis of the above clustering results leads to two observations. First, some of the units belong

[1] Thanks to Avi Shmidman for his assistance with this.

firmly to one cluster or the other. The rest have to be assigned to one cluster or the other because that's the nature of the clustering algorithm, but in fact are not part of what we might think of as the *core* of either cluster. Informally, we say that a unit is in the core of its cluster if it is sufficiently similar to the centroid of its cluster and it is sufficiently more similar to the centroid of its cluster than to any other centroid. Formally, let S be a set of synsets, let B be a set of units, and let C be a clustering of B where the units in B are represented in terms of the synsets in S. For a unit x in cluster C(x) with centroid c(x), we say that x is in the core of C(x) if $\cos'(x,c(x))>\theta_1$ and $\cos'(x,c(x))-\cos'(x,c)>\theta_2$ for every centroid $c \neq c(x)$. In our experiments below, we use $\theta_1=1/\sqrt{2}$ (corresponding to an angle of less than 45 degrees between x and the centroid of its cluster) and $\theta_2=0.1$.

Second, the clusters that we obtain are based on a subset of the full collection of synsets that does the heavy lifting. Formally, we say that a synonym n in synset s is *over-represented* in cluster C if $p(x \in C | n \in x) > p(x \in C | s \in x)$ and $p(x \in C | n \in x) > p(x \in C)$. That is, n is over-represented in C if knowing that n appears in a unit increases the likelihood that the unit is in C, relative to knowing only that some member of n's synset appears in the unit and relative to knowing nothing. We say that a synset s is a *separating* synset for a clustering {C1,C2} if some synonym in s is over-represented in C1 and a different synonym in s is over-represented in C2.

5.1 Defining the Core of a Cluster

We leverage these two observations to formally define the cores of the respective clusters using the following iterative algorithm.

1. Initially, let S be the collection of all synsets, let B be the set of all units in the corpus represented in terms of S, and let {C1,C2} be an initial clustering of the units in B.
2. Reduce B to the cores of C1 and C2.
3. Reduce S to the separating synsets for {C1,C2}.
4. Redefine C1 and C2 to be the clusters obtained from clustering the units in the reduced B represented in terms of the synsets in reduced S.
5. Repeat Steps 2-4 until convergence (no further changes to the retained units and synsets).

At the end of this process, we are left with two well-separated cluster cores and a set of separating synsets. When we compute cores of clusters in our Jeremiah-Ezekiel experiment, 26 of the initial 100 units are eliminated. Of the 154 synsets that appear in the Jeremiah-Ezekiel corpus, 118 are separating synsets for the resulting clustering. The resulting cluster cores split with Jeremiah and Ezekiel as follows:

Book	Cluster I	Cluster II
Jer	36	0
Eze	2	36

We find that all but two of the misplaced units are not part of the core. Thus, we have a better clustering but it is only a partial one.

5.2 Using Cores for Supervised Learning

Now that we have what we believe are strong representatives of each cluster, we can use them in a supervised way to classify the remaining unclustered units. The interesting question is which feature set we should use. Using synonyms would just get us back to where we began. Instead we use the set of generic Bible words introduced earlier. The point to recall is that while this feature set proved inadequate in an unsupervised setting, this does not mean that it is inadequate for separating Jeremiah and Ezekiel, given a few good training examples.

Thus, we use a bag-of-words representation restricted to generic Bible words for the 74 units in our cluster cores and label them according to the cluster to which they were assigned. We now apply SVM to learn a classifier for the two clusters. We assign each unit, including those in the training set, to the class assigned to it by the SVM classifier. The resulting split is as follows:

Book	Cluster I	Cluster II
Jer	51	1
Eze	0	48

Remarkably, even the two Ezekiel chapters that were in the Jeremiah cluster (and hence were essentially misleading training examples) end up on the Ezekiel side of the SVM boundary.

It should be noted that our two-stage approach to clustering is a generic method not specific to our particular application. The point is that there are some feature sets that are very well suited to a particular unsupervised problem but are sparse, so they give only a partial clustering. At the same time, there are other feature sets that are denser and, possibly for that reason, adequate for super-

vised separation of the intended classes but inadequate for unsupervised separation of the intended classes. This suggests an obvious two-stage method for clustering, which we use here to good advantage.

This method is somewhat reminiscent of semi-supervised methods sometimes used in text categorization where few training examples are available (Nigam et al. 2000). However, those methods typically begin with some information, either in the form of a small number of labeled documents or in the form of keywords, while we are not supplied with these. Furthermore, the semi-supervised work bootstraps iteratively, at each stage using features drawn from within the same feature set, while we use exactly two stages, the second of which uses a different type of feature set than the first.

For the reader's convenience, we summarize the entire two-stage method:

1. Represent units in terms of synonyms.
2. Compute similarities of pairs of units using cos'.
3. Use ncut to obtain an initial clustering.
4. Use the iterative method to find cluster cores.
5. Represent units in cluster cores in terms of generic words.
6. Use units in cluster cores as training for learning an SVM classifier.
7. Classify all units according to the learned SVM classifier.

6 Empirical Results

We now test our method on other pairs of biblical books to see if we obtain comparable results to those seen above. We need, therefore, to identify a set of biblical books such that (i) each book is sufficiently long (say, at least 20 chapters), (ii) each is written by one primary author, and (iii) the authors are distinct. Since we wish to use these books as a gold standard, it is important that there be a broad consensus regarding the latter two, potentially controversial, criteria. Our choice is thus limited to the following five books that belong to two biblical sub-genres: Isaiah, Jeremiah, Ezekiel (prophetic literature), Job and Proverbs (wisdom literature). (Due to controversies regarding authorship (Pope 1952, 1965), we include only Chapters 1-33 of Isaiah and only Chapters 3-41 of Job.)

Recall that our experiment is as follows: For each pair of books, we are given all the chapters in the union of the two books and are given no information regarding labels. The object is to sort out the chapters belonging to the respective two books. (The fact that there are precisely two constituent books is given.)

We will use the three algorithms seen above:

1. generic biblical words representation and ncut clustering;
2. synonym representation and ncut clustering;
3. our two-stage algorithm.

We display the results in two separate figures. In Figure 1, we see results for the six pairs of books that belong to different sub-genres. In Figure 2, we see results for the four pairs of books that are in the same genre. (For completeness, we include Jeremiah-Ezekiel, although it served above as a development corpus.) All results are normalized majority diagonal.

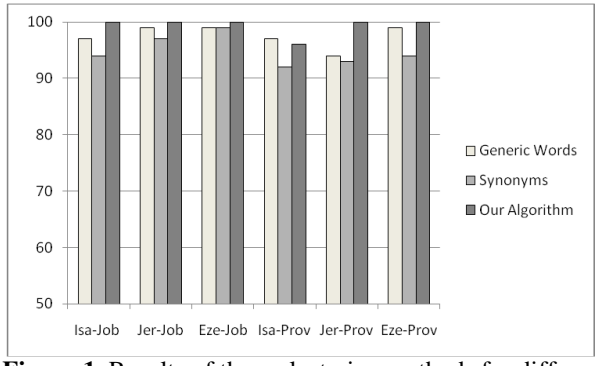


Figure 1. Results of three clustering methods for different-genre pairs

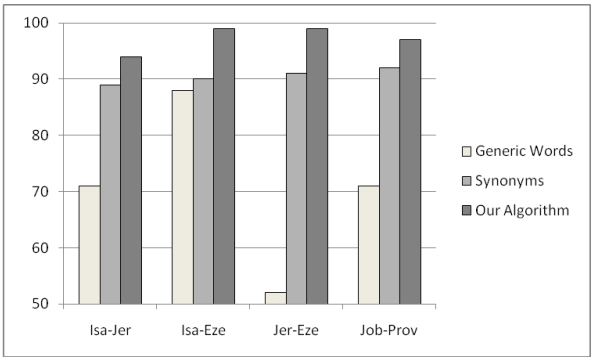


Figure 2. Results of three clustering methods for same-genre pairs

As is evident, for different-genre pairs, even the simplest method works quite well, though not as well as the two-stage method, which is perfect for five of six such pairs. The real advantage of the two-stage method is for same-genre pairs. For

these the simple method is quite erratic, while the two-stage method is near perfect. We note that the synonym method without the second stage is slightly worse than generic words for different-genre pairs (probably because these pairs share relatively few synsets) but is much more consistent for same-genre pairs, giving results in the area of 90% for each such pair. The second stage reduces the errors considerably over the synonym method for both same-genre and different-genre pairs.

7 Decomposing Unsegmented Documents

Up to now, we have considered the case where we are given text that has been pre-segmented into pure authorial units. This does not capture the kind of decomposition problems we face in real life. For example, in the Pentateuch problem, the text is divided up according to chapter, but there is no indication that the chapter breaks are correlated with crossovers between authorial units. Thus, we wish now to generalize our two-stage method to handle unsegmented text.

7.1 Generating Composite Documents

To make the problem precise, let's consider how we might create the kind of document that we wish to decompose. For concreteness, let's think about Jeremiah and Ezekiel. We create a composite document, called *Jer-iel*, as follows:

1. Choose the first k_1 available verses of Jeremiah, where k_1 is a random integer drawn from the uniform distribution over the integers 1 to m.
2. Choose the first k_2 available verses of Ezekiel, where k_2 is a new random integer drawn from the above distribution.
3. Repeat until one of the books is exhausted; then choose the remaining verses of the other book.

For the experiments discussed below, we use m=100 (though further experiments, omitted for lack of space, show that results shown are essentially unchanged for any $m \geq 60$). Furthermore, to simulate the Pentateuch problem, we break Jer-iel into initial units by beginning a new unit whenever we reach the first verse of one of the original chapters of Jeremiah or Ezekiel. (This does not leak any information since there is no inherent connection between these verses and actual crossover points.)

7.2 Applying the Two-Stage Method

Our method works as follows. First, we refine the initial units (each of which might be a mix of verses from Jeremiah and Ezekiel) by splitting them into smaller units that we hope will be pure (wholly from Jeremiah or from Ezekiel). We say that a synset is *doubly-represented* in a unit if the unit includes two different synonyms of that synset. Doubly-represented synsets are an indication that the unit might include verses from two different books. Our object is thus to split the unit in a way that minimizes doubly-represented synonyms. Formally, let M(x) represent the number of synsets for which more than one synonym appear in x. Call $\langle x_1, x_2 \rangle$ a *split* of x if $x = x_1 x_2$. A split $\langle x_1', x_2' \rangle$ is *optimal* if $\langle x_1', x_2' \rangle = \text{argmax } M(x) - \max(M(x_1), M(x_2))$ where the maximum is taken over all splits of x. If for an initial unit, there is some split for which $M(x) - \max(M(x_1), M(x_2))$ is greater than 0, we split the unit optimally; if there is more than one optimal split, we choose the one closest to the middle verse of the unit. (In principle, we could apply this procedure iteratively; in the experiments reported here, we split only the initial units but not split units.)

Next, we run the first six steps of the two-stage method on the units of Jer-iel obtained from the splitting process, as described above, until the point where the SVM classifier has been learned. Now, instead of classifying chapters as in Step 7 of the algorithm, we classify individual verses.

The problem with classifying individual verses is that verses are short and may contain few or no relevant features. In order to remedy this, and also to take advantage of the stickiness of classes across consecutive verses (if a given verse is from a certain book, there is a good chance that the next verse is from the same book), we use two smoothing tactics.

Initially, each verse is assigned a raw score by the SVM classifier, representing its signed distance from the SVM boundary. We smooth these scores by computing for each verse a refined score that is a weighted average of the verse's raw score and the raw scores of the two verses preceding and succeeding it. (In our scheme, the verse itself is given 1.5 times as much weight as its immediate neighbors and three times as much weight as secondary neighbors.)

Moreover, if the refined score is less than 1.0 (the width of the SVM margin), we do not initially

assign the verse to either class. Rather, we check the class of the last assigned verse before it and the first assigned verse after it. If these are the same, the verse is assigned to that class (an operation we call "filling the gaps"). If they are not, the verse remains unassigned.

To illustrate on the case of Jer-iel, our original "munged" book has 96 units. After pre-splitting, we have 143 units. Of these, 105 are pure units. Our two cluster cores, include 33 and 39 units, respectively; 27 of the former are pure Jeremiah and 30 of the latter are pure Ezekiel; no pure units are in the "wrong" cluster core. Applying the SVM classifier learned on the cluster cores to individual verses, 992 of the 2637 verses in Jer-iel lie outside the SVM margin and are assigned to some class. All but four of these are assigned correctly. Filling the gaps assigns a class to 1186 more verses, all but ten of them correctly. Of the remaining 459 unassigned verses, most lie along transition points (where smoothing tends to flatten scores and where preceding and succeeding assigned verses tend to belong to opposite classes).

7.3 Empirical Results

We randomly generated composite books for each of the book pairs considered above. In Figures 3 and 4, we show for each book pair the percentage of all verses in the munged document that are "correctly" classed (that is, in the majority diagonal), the percentage incorrectly classed (minority diagonal) and the percentage not assigned to either class. As is evident, in each case the vast majority of verses are correctly assigned and only a small fraction are incorrectly assigned. That is, we can tease apart the components almost perfectly.

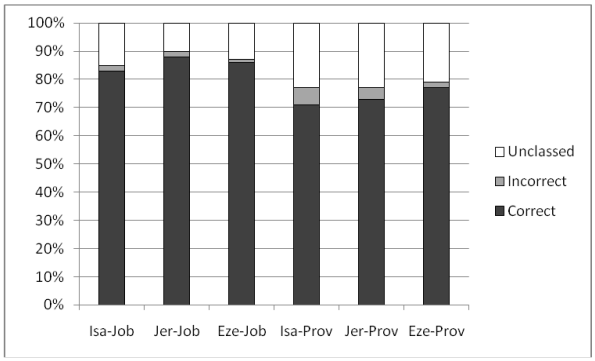


Figure 3. Percentage of verses in each munged different-genre pair of books that are correctly and incorrectly assigned or remain unassigned.

Figure 4. Percentage of verses in each munged same-genre pair of books that are correctly and incorrectly assigned or remain unassigned.

8 Conclusions and Future Work

We have shown that documents can be decomposed into authorial components with very high accuracy by using a two-stage process. First, we establish a reliable partial clustering of units by using synonym choice and then we use these partial clusters as training texts for supervised learning using generic words as features.

We have considered only decompositions into two components, although our method generalizes trivially to more than two components, for example by applying it iteratively. The real challenge is to determine the correct number of components, where this information is not given. We leave this for future work.

Despite this limitation, our success on munged biblical books suggests that our method can be fruitfully applied to the Pentateuch, since the broad consensus in the field is that the Pentateuch can be divided into two main authorial categories: Priestly (P) and non-Priestly (Driver 1909). (Both categories are often divided further, but these subdivisions are more controversial.) We find that our split corresponds to the expert consensus regarding P and non-P for over 90% of the verses in the Pentateuch for which such consensus exists. We have thus been able to largely recapitulate several centuries of painstaking manual labor with our automated method. We offer those instances in which we disagree with the consensus for the consideration of scholars in the field.

In this work, we have exploited the availability of tools for identifying synonyms in biblical literature. In future work, we intend to extend our methods to texts for which such tools are unavailable.

References

J. Astruc. 1753. *Conjectures sur les mémoires originaux dont il paroit que Moyse s'est servi pour composer le livre de la Genèse*. Brussels.

R. E. Bee. 1971. Statistical methods in the study of the Masoretic text of the Old Testament. *J. of the Royal Statistical Society,* 134(1):611-622.

M. J. Berryman, A. Allison, and D. Abbott. 2003. Statistical techniques for text classification based on word recurrence intervals. *Fluctuation and Noise Letters,* 3(1):L1-L10.

J. E. Carpenter, G. Hartford-Battersby. 1900. *The Hexateuch: According to the Revised Version*. London.

J. Clark and C. Hannon. 2007. A classifier system for author recognition using synonym-based features. *Proc. Sixth Mexican International Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence*, vol. 4827, pp. 839-849.

I. S. Dhillon, Y. Guan, and B. Kulis. 2004. Kernel k-means: spectral clustering and normalized cuts. *Proc. ACM International Conference on Knowledge Discovery and Data Mining (KDD),* pp. 551-556.

S. R. Driver. 1909. *An Introduction to the Literature of the Old Testament* (8th ed.). Clark, Edinburgh.

N. Graham, G. Hirst, and B. Marthi. 2005. Segmenting documents by stylistic character. *Natural Language Engineering*, 11(4):397-415.

D. Guthrie, L. Guthrie, and Y. Wilks. 2008. An unsupervised probabilistic approach for the detection of outliers in corpora. *Proc. Sixth International Language Resources and Evaluation (LREC'08),* pp. 28-30.

D. Holmes. 1994. Authorship attribution, *Computers and the Humanities*, 28(2):87-106.

P. Juola. 2008. *Author Attribution*. Series title: *Foundations and Trends in Information Retrieval*. Now Publishing, Delft.

M. Koppel, N. Akiva, and I. Dagan. 2006. Feature instability as a criterion for selecting potential style markers. *J. of the American Society for Information Science and Technology,* 57(11):1519-1525.

M. Koppel, J. Schler, and S. Argamon. 2009. Computational methods in authorship attribution. *J. of the American Society for Information Science and Technology*, 60(1):9-26.

D. L. Mealand. 1995. Correspondence analysis of Luke. *Lit. Linguist Computing*, 10(3):171-182.

S. Meyer zu Eisen and B. Stein. 2006. Intrinsic plagiarism detection. *Proc. European Conference on Information Retrieval (ECIR 2006), Lecture Notes in Computer Science,* vol. 3936, pp. 565–569.

K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. 2000. Text classification from labeled and unlabeled documents using EM, *Machine Learning*, 39(2/3):103-134.

M. H. Pope. 1965. *Job (The Anchor Bible*, Vol. XV). Doubleday, New York, NY.

M. H. Pope. 1952. Isaiah 34 in relation to Isaiah 35, 40-66. *Journal of Biblical Literature,* 71(4):235-243.

Y. Radday. 1970. Isaiah and the computer: A preliminary report, *Computers and the Humanities*, 5(2):65-73.

E. Stamatatos. 2009. A survey of modern authorship attribution methods. *J. of the American Society for Information Science and Technology*, 60(3):538-556.

J. Strong. 1890. *The Exhaustive Concordance of the Bible*. Nashville, TN. (Online edition: http://www.htmlbible.com/sacrednamebiblecom/kjvstrongs/STRINDEX.htm; accessed 14 November 2010.)

Discovering Sociolinguistic Associations with Structured Sparsity

Jacob Eisenstein Noah A. Smith Eric P. Xing
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{jacobeis,nasmith,epxing}@cs.cmu.edu

Abstract

We present a method to discover robust and interpretable sociolinguistic associations from raw geotagged text data. Using aggregate demographic statistics about the authors' geographic communities, we solve a multi-output regression problem between demographics and lexical frequencies. By imposing a composite $\ell_{1,\infty}$ regularizer, we obtain structured sparsity, driving entire rows of coefficients to zero. We perform two regression studies. First, we use term frequencies to predict demographic attributes; our method identifies a compact set of words that are strongly associated with author demographics. Next, we conjoin demographic attributes into *features*, which we use to predict term frequencies. The composite regularizer identifies a small number of features, which correspond to communities of authors united by shared demographic and linguistic properties.

1 Introduction

How is language influenced by the speaker's sociocultural identity? Quantitative sociolinguistics usually addresses this question through carefully crafted studies that correlate individual demographic attributes and linguistic variables—for example, the interaction between income and the "dropped r" feature of the New York accent (Labov, 1966). But such studies require the knowledge to select the "dropped r" and the speaker's income, from thousands of other possibilities. In this paper, we present a method to acquire such patterns from raw data. Using multi-output regression with structured sparsity, our method identifies a small subset of lexical items that are most influenced by demographics, and discovers conjunctions of demographic attributes that are especially salient for lexical variation.

Sociolinguistic associations are difficult to model, because the space of potentially relevant interactions is large and complex. On the linguistic side there are thousands of possible variables, even if we limit ourselves to unigram lexical features. On the demographic side, the interaction between demographic attributes is often non-linear: for example, gender may negate or amplify class-based language differences (Zhang, 2005). Thus, additive models which assume that each demographic attribute makes a linear contribution are inadequate.

In this paper, we explore the large space of potential sociolinguistic associations using structured sparsity. We treat the relationship between language and demographics as a set of multi-input, multi-output regression problems. The regression coefficients are arranged in a matrix, with rows indicating predictors and columns indicating outputs. We apply a composite regularizer that drives entire rows of the coefficient matrix to zero, yielding compact, interpretable models that reuse features across different outputs. If we treat the lexical frequencies as inputs and the author's demographics as outputs, the induced sparsity pattern reveals the set of lexical items that is most closely tied to demographics. If we treat the demographic attributes as inputs and build a model to predict the text, we can incrementally construct a conjunctive feature space of demographic attributes, capturing key non-linear interactions.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1365–1374,
Portland, Oregon, June 19-24, 2011.

The primary purpose of this research is exploratory data analysis to identify both the most linguistic-salient demographic features, and the most demographically-salient words. However, this model also enables predictions about demographic features by analyzing raw text, potentially supporting applications in targeted information extraction or advertising. On the task of predicting demographics from text, we find that our sparse model yields performance that is statistically indistinguishable from the full vocabulary, even with a reduction in the model complexity an order of magnitude. On the task of predicting text from author demographics, we find that our incrementally constructed feature set obtains significantly better perplexity than a linear model of demographic attributes.

	mean	std. dev.
race & ethnicity		
% white	52.1	29.0
% African American	32.2	29.1
% Hispanic	15.7	18.3
language		
% English speakers	73.7	18.4
% Spanish speakers	14.6	15.6
% other language speakers	11.7	9.2
socioeconomic		
% urban	95.1	14.3
% with family	64.1	14.4
% renters	48.9	23.4
median income ($)	42,500	18,100

Table 1: The demographic attributes used in this research.

2 Data

Our dataset is derived from prior work in which we gathered the text and geographical locations of 9,250 microbloggers on the website `twitter.com` (Eisenstein et al., 2010). Bloggers were selected from a pool of frequent posters whose messages include metadata indicating a geographical location within a bounding box around the continental United States. We limit the vocabulary to the 5,418 terms which are used by at least 40 authors; no stoplists are applied, as the use of standard or non-standard orthography for stopwords (e.g., *to* vs. *2*) may convey important information about the author. The dataset includes messages during the first week of March 2010.

O'Connor et al. (2010) obtained aggregate demographic statistics for these data by mapping geolocations to publicly-available data from the U. S. Census ZIP Code Tabulation Areas (ZCTA).[1] There are 33,178 such areas in the USA (the 9,250 microbloggers in our dataset occupy 3,458 unique ZCTAs), and they are designed to contain roughly equal numbers of inhabitants and demographically-homogeneous populations. The demographic attributes that we consider in this paper are shown in Table 1. All attributes are based on self-reports. The race and ethnicity attributes are not mutually exclusive—individuals can indicate any number of races or ethnicities. The "other language" attribute aggregates all languages besides English and Spanish. "Urban areas" refer to sets of census tracts or census blocks which contain at least 2,500 residents; our "% urban" attribute is the percentage of individuals in each ZCTA who are listed as living in an urban area. We also consider the percentage of individuals who live with their families, the percentage who live in rented housing, and the median reported income in each ZCTA.

While geographical aggregate statistics are frequently used to proxy for individual socioeconomic status in research areas such as public health (e.g., Rushton, 2008), it is clear that interpretation must proceed with caution. Consider an author from a ZIP code in which 60% of the residents are Hispanic:[2] we do not know the likelihood that the author is Hispanic, because the set of Twitter users is not a representative sample of the overall population. Polling research suggests that users of both Twitter (Smith and Rainie, 2010) and geolocation services (Zickuhr and Smith, 2010) are much more diverse with respect to age, gender, race and ethnicity than the general population of Internet users. Nonetheless, at present we can only use aggregate statistics to make inferences about the geographic communities in which our authors live, and not the authors themselves.

[1] `http://www.census.gov/support/cen2000.html`

[2] In the U.S. Census, the official ethnonym is *Hispanic or Latino*; for brevity we will use *Hispanic* in the rest of this paper.

3 Models

The selection of both words and demographic features can be framed in terms of multi-output regression with structured sparsity. To select the lexical indicators that best predict demographics, we construct a regression problem in which term frequencies are the predictors and demographic attributes are the outputs; to select the demographic features that predict word use, this arrangement is reversed. Through structured sparsity, we learn models in which entire sets of coefficients are driven to zero; this tells us which words and demographic features can safely be ignored.

This section describes the model and implementation for output-regression with structured sparsity; in Section 4 and 5 we give the details of its application to select terms and demographic features. Formally, we consider the linear equation $\mathbf{Y} = \mathbf{XB} + \epsilon$, where,

- $\mathbf{Y}$ is the dependent variable matrix, with dimensions $N \times T$, where N is the number of samples and T is the number of output dimensions (or *tasks*);
- $\mathbf{X}$ is the independent variable matrix, with dimensions $N \times P$, where P is the number of input dimensions (or *predictors*);
- $\mathbf{B}$ is the matrix of regression coefficients, with dimensions $P \times T$;
- ϵ is a $N \times T$ matrix in which each element is noise from a zero-mean Gaussian distribution.

We would like to solve the unconstrained optimization problem,

$$\text{minimize}_{\mathbf{B}} \qquad ||\mathbf{Y} - \mathbf{XB}||_F^2 + \lambda R(\mathbf{B}), \qquad (1)$$

where $||\mathbf{A}||_F^2$ indicates the squared Frobenius norm $\sum_i \sum_j a_{ij}^2$, and the function $R(\mathbf{B})$ defines a norm on the regression coefficients $\mathbf{B}$. Ridge regression applies the ℓ_2 norm $R(\mathbf{B}) = \sum_{t=1}^{T} \sqrt{\sum_p^P b_{pt}^2}$, and lasso regression applies the ℓ_1 norm $R(\mathbf{B}) = \sum_{t=1}^{T} \sum_p^P |b_{pt}|$; in both cases, it is possible to decompose the multi-output regression problem, treating each output dimension separately. However, our working hypothesis is that there will be substantial correlations across both the vocabulary and the demographic features—for example, a demographic feature such as the percentage of Spanish speakers will predict a large set of words. Our goal is to select a small set of predictors yielding good performance across all output dimensions. Thus, we desire *structured* sparsity, in which entire rows of the coefficient matrix $\mathbf{B}$ are driven to zero.

Structured sparsity is not achieved by the lasso's ℓ_1 norm. The lasso gives element-wise sparsity, in which many entries of $\mathbf{B}$ are driven to zero, but each predictor may have a non-zero value for some output dimension. To drive entire rows of $\mathbf{B}$ to zero, we require a composite regularizer. We consider the $\ell_{1,\infty}$ norm, which is the sum of ℓ_∞ norms across output dimensions: $R(\mathbf{B}) = \sum_t^T \max_p b_{pt}$ (Turlach et al., 2005). This norm, which corresponds to a *multi-output lasso* regression, has the desired property of driving entire rows of $\mathbf{B}$ to zero.

3.1 Optimization

There are several techniques for solving the $\ell_{1,\infty}$ normalized regression, including interior point methods (Turlach et al., 2005) and projected gradient (Duchi et al., 2008; Quattoni et al., 2009). We choose the blockwise coordinate descent approach of Liu et al. (2009) because it is easy to implement and efficient: the time complexity of each iteration is independent of the number of samples.[3]

Due to space limitations, we defer to Liu et al. (2009) for a complete description of the algorithm. However, we note two aspects of our implementation which are important for natural language processing applications. The algorithm's efficiency is accomplished by precomputing the matrices $\mathbf{C} = \tilde{\mathbf{X}}^\mathsf{T}\tilde{\mathbf{Y}}$ and $\mathbf{D} = \tilde{\mathbf{X}}^\mathsf{T}\tilde{\mathbf{X}}$, where $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ are the standardized versions of $\mathbf{X}$ and $\mathbf{Y}$, obtained by subtracting the mean and scaling by the variance. Explicit mean correction would destroy the sparse term frequency data representation and render us unable to store the data in memory; however, we can achieve the same effect by computing $\mathbf{C} = \mathbf{X}^\mathsf{T}\mathbf{Y} - N\bar{\boldsymbol{x}}^\mathsf{T}\bar{\boldsymbol{y}}$, where $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{y}}$ are row vectors indicating the means

[3] Our implementation is available at `http://sailing.cs.cmu.edu/sociolinguistic.html`.

of $\mathbf{X}$ and $\mathbf{Y}$ respectively.[4] We can similarly compute $\mathbf{D} = \mathbf{X}^\mathsf{T}\mathbf{X} - N\bar{\boldsymbol{x}}^\mathsf{T}\bar{\boldsymbol{x}}$.

If the number of predictors is too large, it may not be possible to store the dense matrix $\mathbf{D}$ in memory. We have found that approximation based on the truncated singular value decomposition provides an effective trade-off of time for space. Specifically, we compute $\mathbf{X}^\mathsf{T}\mathbf{X} \approx$

$$\mathbf{U}\mathbf{S}\mathbf{V}^\mathsf{T}\left(\mathbf{U}\mathbf{S}\mathbf{V}^\mathsf{T}\right)^\mathsf{T} = \mathbf{U}\left(\mathbf{S}\mathbf{V}^\mathsf{T}\mathbf{V}\mathbf{S}^\mathsf{T}\mathbf{U}^\mathsf{T}\right) = \mathbf{U}\mathbf{M}.$$

Lower truncation levels are less accurate, but are faster and require less space: for K singular values, the storage cost is $\mathcal{O}(KP)$, instead of $\mathcal{O}(P^2)$; the time cost increases by a factor of K. This approximation was not necessary in the experiments presented here, although we have found that it performs well as long as the regularizer is not too close to zero.

3.2 Regularization

The regularization constant λ can be computed using cross-validation. As λ increases, we reuse the previous solution of $\mathbf{B}$ for initialization; this "warm start" trick can greatly accelerate the computation of the overall regularization path (Friedman et al., 2010). At each λ_i, we solve the sparse multi-output regression; the solution $\mathbf{B}_i$ defines a sparse set of predictors for all tasks.

We then use this limited set of predictors to construct a new input matrix $\hat{\mathbf{X}}_i$, which serves as the input in a standard ridge regression, thus refitting the model. The tuning set performance of this regression is the score for λ_i. Such post hoc refitting is often used in tandem with the lasso and related sparse methods; the effectiveness of this procedure has been demonstrated in both theory (Wasserman and Roeder, 2009) and practice (Wu et al., 2010). The regularization parameter of the ridge regression is determined by internal cross-validation.

4 Predicting Demographics from Text

Sparse multi-output regression can be used to select a subset of vocabulary items that are especially indicative of demographic and geographic differences. Starting from the regression problem (1), the predictors $\mathbf{X}$ are set to the term frequencies, with one column for each word type and one row for each author in the dataset. The outputs $\mathbf{Y}$ are set to the ten demographic attributes described in Table 1 (we consider much larger demographic feature spaces in the next section) The $\ell_{1,\infty}$ regularizer will drive entire rows of the coefficient matrix $\mathbf{B}$ to zero, eliminating all demographic effects for many words.

4.1 Quantitative Evaluation

We evaluate the ability of lexical features to predict the demographic attributes of their authors (as proxied by the census data from the author's geographical area). The purpose of this evaluation is to assess the predictive ability of the compact subset of lexical items identified by the multi-output lasso, as compared with the full vocabulary. In addition, this evaluation establishes a baseline for performance on the demographic prediction task.

We perform five-fold cross-validation, using the multi-output lasso to identify a sparse feature set in the training data. We compare against several other dimensionality reduction techniques, matching the number of features obtained by the multi-output lasso at each fold. First, we compare against a truncated singular value decomposition, with the truncation level set to the number of terms selected by the multi-output lasso; this is similar in spirit to vector-based lexical semantic techniques (Schütze and Pedersen, 1993). We also compare against simply selecting the N most frequent terms, and the N terms with the greatest variance in frequency across authors. Finally, we compare against the complete set of all 5,418 terms. As before, we perform post hoc refitting on the training data using a standard ridge regression. The regularization constant for the ridge regression is identified using nested five-fold cross validation within the training set.

We evaluate on the refit models on the heldout test folds. The scoring metric is Pearson's correlation coefficient between the predicted and true demographics: $\rho(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{\text{cov}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\sigma_{\boldsymbol{y}}\sigma_{\hat{\boldsymbol{y}}}}$, with $\text{cov}(\boldsymbol{y}, \hat{\boldsymbol{y}})$ indicating the covariance and $\sigma_{\boldsymbol{y}}$ indicating the standard deviation. On this metric, a perfect predictor will score 1 and a random predictor will score 0. We report the average correlation across all ten demo-

[4] Assume without loss of generality that $\mathbf{X}$ and $\mathbf{Y}$ are scaled to have variance 1, because this scaling does not affect the sparsity pattern.

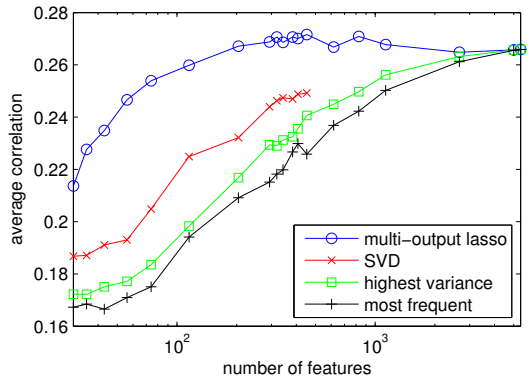


Figure 1: Average correlation plotted against the number of active features (on a logarithmic scale).

graphic attributes, as well as the individual correlations.

Results Table 2 shows the correlations obtained by regressions performed on a range of different vocabularies, averaged across all five folds. Linguistic features are best at predicting race, ethnicity, language, and the proportion of renters; the other demographic attributes are more difficult to predict. Among feature sets, the highest average correlation is obtained by the full vocabulary, but the multi-output lasso obtains nearly identical performance using a feature set that is an order of magnitude smaller. Applying the Fischer transformation, we find that all correlations are statistically significant at $p < .001$.

The Fischer transformation can also be used to estimate 95% confidence intervals around the correlations. The extent of the confidence intervals varies slightly across attributes, but all are tighter than ± 0.02. We find that the multi-output lasso and the full vocabulary regression are not significantly different on any of the attributes. Thus, the multi-output lasso achieves a 93% compression of the feature set without a significant decrease in predictive performance. The multi-output lasso yields higher correlations than the other dimensionality reduction techniques on all of the attributes; these differences are statistically significant in many—but not all—cases. The correlations for each attribute are clearly not independent, so we do not compare the average across attributes.

Recall that the regularization coefficient was chosen by nested cross-validation within the training set; the average number of features selected is 394.6. Figure 1 shows the performance of each dimensionality-reduction technique across the regularization path for the first of five cross-validation folds. Computing the truncated SVD of a sparse matrix at very large truncation levels is computationally expensive, so we cannot draw the complete performance curve for this method. The multi-output lasso dominates the alternatives, obtaining a particularly strong advantage with very small feature sets. This demonstrates its utility for identifying interpretable models which permit qualitative analysis.

4.2 Qualitative Analysis

For a qualitative analysis, we retrain the model on the full dataset, and tune the regularization to identify a compact set of 69 features. For each identified term, we apply a significance test on the relationship between the presence of each term and the demographic indicators shown in the columns of the table. Specifically, we apply the Wald test for comparing the means of independent samples, while making the Bonferroni correction for multiple comparisons (Wasserman, 2003). The use of sparse multi-output regression for variable selection increases the power of post hoc significance testing, because the Bonferroni correction bases the threshold for statistical significance on the total number of comparisons. We find 275 associations at the $p < .05$ level; at the higher threshold required by a Bonferroni correction for comparisons among all terms in the vocabulary, 69 of these associations would have been missed.

Table 3 shows the terms identified by our model which have a significant correlation with at least one of the demographic indicators. We divide words in the list into categories, which order alphabetically by the first word in each category: emoticons; standard English, defined as words with Wordnet entries; proper names; abbreviations; non-English words; non-standard words used with English. The categorization was based on the most frequent sense in an informal analysis of our data. A glossary of non-standard terms is given in Table 4.

Some patterns emerge from Table 3. Standard English words tend to appear in areas with more

vocabulary	# features	average	white	Afr. Am.	Hisp.	Eng. lang.	Span. lang.	other lang.	urban	family	renter	med. inc.
full	5418	0.260	0.337	0.318	0.296	0.384	0.296	0.256	0.155	0.113	0.295	0.152
multi-output lasso	394.6	0.260	0.326	0.308	0.304	0.383	0.303	0.249	0.153	0.113	0.302	0.156
SVD		0.237	0.321	0.299	0.269	0.352	0.272	0.226	0.138	0.081	0.278	0.136
highest variance		0.220	0.309	0.287	0.245	0.315	0.248	0.199	0.132	0.085	0.250	0.135
most frequent		0.204	0.294	0.264	0.222	0.293	0.229	0.178	0.129	0.073	0.228	0.126

Table 2: Correlations between predicted and observed demographic attributes, averaged across cross validation folds.

English speakers; predictably, Spanish words tend to appear in areas with Spanish speakers and Hispanics. Emoticons tend to be used in areas with many Hispanics and few African Americans. Abbreviations (e.g., *lmaoo*) have a nearly uniform demographic profile, displaying negative correlations with whites and English speakers, and positive correlations with African Americans, Hispanics, renters, Spanish speakers, and areas classified as urban.

Many non-standard English words (e.g., *dats*) appear in areas with high proportions of renters, African Americans, and non-English speakers, though a subset (*haha*, *hahaha*, and *yep*) display the opposite demographic pattern. Many of these non-standard words are phonetic transcriptions of standard words or phrases: *that's→dats*, *what's up→wassup*, *I'm going to→ima*. The relationship between these transcriptions and the phonological characteristics of dialects such as African-American Vernacular English is a topic for future work.

5 Conjunctive Demographic Features

Next, we demonstrate how to select conjunctions of demographic features that predict text. Again, we apply multi-output regression, but now we reverse the direction of inference: the predictors are demographic features, and the outputs are term frequencies. The sparsity-inducing $\ell_{1,\infty}$ norm will select a subset of demographic features that explain the term frequencies.

We create an initial feature set $f^{(0)}(\mathbf{X})$ by binning each demographic attribute, using five equal-frequency bins. We then constructive conjunctive features by applying a procedure inspired by related work in computational biology, called "Screen and Clean" (Wu et al., 2010). On iteration i:

- Solve the sparse multi-output regression problem $\mathbf{Y} = f^{(i)}(\mathbf{X})\mathbf{B}^{(i)} + \epsilon$.
- Select a subset of features $S^{(i)}$ such that $m \in S^{(i)}$ iff $\max_j |\boldsymbol{b}^{(i)}_{m,j}| > 0$. These are the row indices of the predictors with non-zero coefficients.
- Create a new feature set $f^{(i+1)}(\mathbf{X})$, including the conjunction of each feature (and its negation) in $S^{(i)}$ with each feature in the initial set $f^{(0)}(\mathbf{X})$.

We iterate this process to create features that conjoin as many as three attributes. In addition to the binned versions of the demographic attributes described in Table 1, we include geographical information. We built Gaussian mixture models over the locations, with 3, 5, 8, 12, 17, and 23 components. For each author we include the most likely cluster assignment in each of the six mixture models. For efficiency, the outputs $\mathbf{Y}$ are not set to the raw term frequencies; instead we compute a truncated singular value decomposition of the term frequencies $\mathbf{W} \approx \mathbf{U}\mathbf{V}\mathbf{D}^\mathsf{T}$, and use the basis $\mathbf{U}$. We set the truncation level to 100.

5.1 Quantitative Evaluation

The ability of the induced demographic features to predict text is evaluated using a traditional perplexity metric. The same test and training split is used from the vocabulary experiments. We construct a language model from the induced demographic features by training a multi-output ridge regression, which gives a matrix $\hat{\mathbf{B}}$ that maps from demographic features to term frequencies across the entire vocabulary. For each document in the test set, the "raw" predicted language model is $\hat{\boldsymbol{y}}_d = f(\boldsymbol{x}_d)\mathbf{B}$, which is then normalized. The probability mass assigned

	white	Afr. Am.	Hisp.	Eng. lang.	Span. lang.	other lang.	urban	family	renter	med. inc.
-_-	-		+	-	+	+	+			
;)		-	+	-	+					
:(		-								
:)		-								
:d	+	-	+	-	+					
as			-	+	-					
awesome	+	-					-		-	+
break			-	+	-	-				
campus			-	+	-	-				
dead	-	+		-	+		+		+	
hell			-	+	-	-				
shit	-								+	
train				-	+				+	
will			-	+	-					
would				+					-	
atlanta			-	+	-	-				
famu		+	-	+	-	-				-
harlem				-					+	
bbm	-	+		-		+	+		+	
lls		+	-	+	-	-				
lmaoo	-	+	+	-	+	+	+		+	
lmaooo	-	+	+	-	+	+	+		+	
lmaoooo	-	+	+	-	+	+			+	
lmfaoo	-		+	-	+	+			+	
lmfaooo	-		+	-	+	+			+	
lml	-	+	+	-	+	+	+		+	-
odee	-		+	-	+		+		+	
omw	-	+	+	-	+	+	+		+	
smfh	-	+	+	-	+	+	+		+	
smh	-	+					+		+	
w|	-		+	-	+	+	+		+	
con			+	-	+				+	
la		-	+	-	+					
si		-	+	-	+					
dats	-	+		-					+	-
deadass	-	+	+	-	+	+	+		+	
haha	+	-							-	
hahah	+	-								
hahaha	+	-							-	+
ima	-		+	-	+				+	
madd	-			-		+		+		
nah	-		+	-	+	+			+	
ova	-	+		-					+	
sis	-	+							+	
skool	-	+		-		+	+		+	-
wassup	-	+	+	-	+	+	+		+	-
wat	-	+	+	-	+	+	+		+	-
ya	-	+							+	
yall	-	+								
yep			-	+	-	-	-		-	
yoo	-	+	+	-	+	+	+		+	
yooo	-	+		-	+				+	

Table 3: Demographically-indicative terms discovered by multi-output sparse regression. Statistically significant ($p < .05$) associations are marked with a + or -.

term	definition	term	definition
bbm	Blackberry Messenger	omw	on my way
dats	that's	ova	over
dead(ass)	very	sis	sister
famu	Florida Agricultural and Mechanical Univ.	skool	school
		sm(f)h	shake my (fucking) head
ima	I'm going to		
lls	laughing like shit	w|	with
lm(f)ao+	laughing my (fucking) ass off	wassup	what's up
		wat	what
lml	love my life	ya	your, you
madd	very, lots	yall	you plural
nah	no	yep	yes
odee	very	yoo+	you

Table 4: A glossary of non-standard terms from Table 3. Definitions are obtained by manually inspecting the context in which the terms appear, and by consulting `www.urbandictionary.com`.

model	perplexity
induced demographic features	333.9
raw demographic attributes	335.4
baseline (no demographics)	337.1

Table 5: Word perplexity on test documents, using language models estimated from induced demographic features, raw demographic attributes, and a relative-frequency baseline. Lower scores are better.

to unseen words is determined through nested cross-validation. We compare against a baseline language model obtained from the training set, again using nested cross-validation to set the probability of unseen terms.

Results are shown in Table 5. The language models induced from demographic data yield small but statistically significant improvements over the baseline (Wilcoxon signed-rank test, $p < .001$). Moreover, the model based on conjunctive features significantly outperforms the model constructed from raw attributes ($p < .001$).

5.2 Features Discovered

Our approach discovers 37 conjunctive features, yielding the results shown in Table 5. We sort all features by frequency, and manually select a subset to display in Table 6. Alongside each feature, we show the words with the highest and lowest log-odds ratios with respect to the feature. Many of these terms are non-standard; while space does not permit a complete glossary, some are defined in Table 4 or in our earlier work (Eisenstein et al., 2010).

	feature			positive terms	negative terms
1	geo: Northeast			m2 brib mangoville soho odeee	fasho #ilovefamu foo coo fina
2	geo: NYC			mangoville lolss m2 brib wordd	bahaha fasho goofy #ilovefamu tacos
4	geo: South+Midwest	renter $\leq$ 0.615	white $\leq$ 0.823	hme muthafucka bae charlotte tx	odeee m2 lolss diner mangoville
7	Afr. Am. $>$ 0.101	renter $>$ 0.615	Span. lang. $>$ 0.063	dhat brib odeee lolss wassupp	bahaha charlotte california ikr enter
8	Afr. Am. $\leq$ 0.207	Hispanic $>$ 0.119	Span. lang. $>$ 0.063	les ahah para san donde	bmore ohio #lowkey #twitterjail nahhh
9	geo: NYC	Span. lang. $\leq$ 0.213		mangoville thatt odeee lolss buzzin	landed rodney jawn wiz golf
12	Afr. Am. $>$ 0.442	geo: South+Midwest	white $\leq$ 0.823	#ilovefamu panama midterms willies #lowkey	knoe esta pero odeee hii
15	geo: West Coast	other lang. $>$ 0.110		ahah fasho san koo diego	granted pride adore phat pressure
17	Afr. Am. $>$ 0.442	geo: NYC	other lang. $\leq$ 0.110	lolss iim buzzin qonna qood	foo tender celebs pages pandora
20	Afr. Am. $\leq$ 0.207	Span. lang. $>$ 0.063	white $>$ 0.823	del bby cuando estoy muscle	knicks becoming uncomfortable large granted
23	Afr. Am. $\leq$ 0.050	geo: West	Span. lang. $\leq$ 0.106	leno it'd 15th hacked government	knicks liquor uu hunn homee
33	Afr. Am. $>$ 0.101	geo: SF Bay	Span. lang. $>$ 0.063	hella aha california bay o.o	aj everywhere phones shift regardless
36	Afr. Am. $\leq$ 0.050	geo: DC/Philadelphia	Span. lang. $\leq$ 0.106	deh opens stuffed yaa bmore	hmmmmm dyin tea cousin hella

Table 6: Conjunctive features discovered by our method with a strong sparsity-inducing prior, ordered by frequency. We also show the words with high log-odds for each feature (postive terms) and its negation (negative terms).

In general, geography was a strong predictor, appearing in 25 of the 37 conjunctions. Features 1 and 2 (F1 and F2) are purely geographical, capturing the northeastern United States and the New York City area. The geographical area of F2 is completely contained by F1; the associated terms are thus very similar, but by having both features, the model can distinguish terms which are used in northeastern areas outside New York City, as well as terms which are especially likely in New York.[5]

Several features conjoin geography with demographic attributes. For example, F9 further refines the New York City area by focusing on communities that have relatively low numbers of Spanish speakers; F17 emphasizes New York neighborhoods that have very high numbers of African Americans and few speakers of languages other than English and Spanish. The regression model can use these features in combination to make fine-grained distinctions about the differences between such neighborhoods. Outside New York, we see that F4 combines a broad geographic area with attributes that select at least moderate levels of minorities and fewer renters (a proxy for areas that are less urban), while F15 identifies West Coast communities with large numbers of speakers of languages other than English and Spanish.

Race and ethnicity appear in 28 of the 37 conjunctions. The attribute indicating the proportion of African Americans appeared in 22 of these features, strongly suggesting that African American Vernacular English (Rickford, 1999) plays an important role in social media text. Many of these features conjoined the proportion of African Americans with geographical features, identifying local linguistic styles used predominantly in either African American or white communities. Among features which focus on minority communities, F17 emphasizes the New York area, F33 focuses on the San Francisco Bay area, and F12 selects a broad area in the Midwest and South. Conversely, F23 selects areas with very few African Americans and Spanish-speakers in the western part of the United States, and F36 selects for similar demographics in the area of Washington and Philadelphia.

Other features conjoined the proportion of African Americans with the proportion of Hispanics and/or Spanish speakers. In some cases, features selected for high proportions of both African Americans and Hispanics; for example, F7 seems to identify a general "urban minority" group, emphasizing renters, African Americans, and Spanish speakers. Other features differentiate between African Ameri-

[5] *Mangoville* and *M2* are clubs in New York; *fasho* and *coo* were previously found to be strongly associated with the West Coast (Eisenstein et al., 2010).

cans and Hispanics: F8 identifies regions with many Spanish speakers and Hispanics, but few African Americans; F20 identifies regions with both Spanish speakers and whites, but few African Americans. F8 and F20 tend to emphasize more Spanish words than features which select for both African Americans and Hispanics.

While race, geography, and language predominate, the socioeconomic attributes appear in far fewer features. The most prevalent attribute is the proportion of renters, which appears in F4 and F7, and in three other features not shown here. This attribute may be a better indicator of the urban/rural divide than the "% urban" attribute, which has a very low threshold for what counts as urban (see Table 1). It may also be a better proxy for wealth than median income, which appears in only one of the thirty-seven selected features. Overall, the selected features tend to include attributes that are easy to predict from text (compare with Table 2).

6 Related Work

Sociolinguistics has a long tradition of quantitative and computational research. Logistic regression has been used to identify relationships between demographic features and linguistic variables since the 1970s (Cedergren and Sankoff, 1974). More recent developments include the use of mixed factor models to account for idiosyncrasies of individual speakers (Johnson, 2009), as well as clustering and multidimensional scaling (Nerbonne, 2009) to enable aggregate inference across multiple linguistic variables. However, all of these approaches assume that both the linguistic indicators and demographic attributes have already been identified by the researcher. In contrast, our approach focuses on identifying these indicators automatically from data. We view our approach as an exploratory complement to more traditional analysis.

There is relatively little computational work on identifying speaker demographics. Chang et al. (2010) use U.S. Census statistics about the ethnic distribution of last names as an anchor in a latent-variable model that infers the ethnicity of Facebook users; however, their paper analyzes social behavior rather than language use. In unpublished work, David Bamman uses geotagged Twitter text and U.S. Census statistics to estimate the age, gender, and racial distributions of various lexical items.[6] Eisenstein et al. (2010) infer geographic clusters that are coherent with respect to both location and lexical distributions; follow-up work by O'Connor et al. (2010) applies a similar generative model to demographic data. The model presented here differs in two key ways: first, we use sparsity-inducing regularization to perform variable selection; second, we eschew high-dimensional mixture models in favor of a bottom-up approach of building conjunctions of demographic and geographic attributes. In a mixture model, each component must define a distribution over all demographic variables, which may be difficult to estimate in a high-dimensional setting.

Early examples of the use of sparsity in natural language processing include maximum entropy classification (Kazama and Tsujii, 2003), language modeling (Goodman, 2004), and incremental parsing (Riezler and Vasserman, 2004). These papers all apply the standard lasso, obtaining sparsity for a single output dimension. Structured sparsity has rarely been applied to language tasks, but Duh et al. (2010) reformulated the problem of reranking N-best lists as multi-task learning with structured sparsity.

7 Conclusion

This paper demonstrates how regression with structured sparsity can be applied to select words and conjunctive demographic features that reveal sociolinguistic associations. The resulting models are compact and interpretable, with little cost in accuracy. In the future we hope to consider richer linguistic models capable of identifying multi-word expressions and syntactic variation.

Acknowledgments We received helpful feedback from Moira Burke, Scott Kiesling, Seyoung Kim, André Martins, Kriti Puniyani, and the anonymous reviewers. Brendan O'Connor provided the data for this research, and Seunghak Lee shared a Matlab implementation of the multi-output lasso, which was the basis for our C implementation. This research was enabled by AFOSR FA9550010247, ONR N0001140910758, NSF CAREER DBI-0546594, NSF CAREER IIS-1054319, NSF IIS-0713379, an Alfred P. Sloan Fellowship, and Google's support of the Worldly Knowledge project at CMU.

[6] http://www.lexicalist.com

References

Henrietta J. Cedergren and David Sankoff. 1974. Variable rules: Performance as a statistical reflection of competence. *Language*, 50(2):333–355.

Jonathan Chang, Itamar Rosenn, Lars Backstrom, and Cameron Marlow. 2010. ePluribus: Ethnicity on social networks. In *Proceedings of ICWSM*.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. 2008. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proceedings of ICML*.

Kevin Duh, Katsuhito Sudoh, Hajime Tsukada, Hideki Isozaki, and Masaaki Nagata. 2010. n-best reranking by multitask learning. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics*.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A latent variable model of geographic lexical variation. In *Proceedings of EMNLP*.

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. 2010. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22.

Joshua Goodman. 2004. Exponential priors for maximum entropy models. In *Proceedings of NAACL-HLT*.

Daniel E. Johnson. 2009. Getting off the GoldVarb standard: Introducing Rbrul for mixed-effects variable rule analysis. *Language and Linguistics Compass*, 3(1):359–383.

Jun'ichi Kazama and Jun'ichi Tsujii. 2003. Evaluation and extension of maximum entropy models with inequality constraints. In *Proceedings of EMNLP*.

William Labov. 1966. *The Social Stratification of English in New York City*. Center for Applied Linguistics.

Han Liu, Mark Palatucci, and Jian Zhang. 2009. Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In *Proceedings of ICML*.

John Nerbonne. 2009. Data-driven dialectology. *Language and Linguistics Compass*, 3(1):175–198.

Brendan O'Connor, Jacob Eisenstein, Eric P. Xing, and Noah A. Smith. 2010. A mixture model of demographic lexical variation. In *Proceedings of NIPS Workshop on Machine Learning in Computational Social Science*.

Ariadna Quattoni, Xavier Carreras, Michael Collins, and Trevor Darrell. 2009. An efficient projection for $\ell_{1,\infty}$ regularization. In *Proceedings of ICML*.

John R. Rickford. 1999. *African American Vernacular English*. Blackwell.

Stefan Riezler and Alexander Vasserman. 2004. Incremental feature selection and ℓ_1 regularization for relaxed maximum-entropy modeling. In *Proceedings of EMNLP*.

Gerard Rushton, Marc P. Armstrong, Josephine Gittler, Barry R. Greene, Claire E. Pavlik, Michele M. West, and Dale L. Zimmerman, editors. 2008. *Geocoding Health Data: The Use of Geographic Codes in Cancer Prevention and Control, Research, and Practice*. CRC Press.

Hinrich Schütze and Jan Pedersen. 1993. A vector model for syntagmatic and paradigmatic relatedness. In *Proceedings of the 9th Annual Conference of the UW Centre for the New OED and Text Research*.

Aaron Smith and Lee Rainie. 2010. Who tweets? Technical report, Pew Research Center, December.

Berwin A. Turlach, William N. Venables, and Stephen J. Wright. 2005. Simultaneous variable selection. *Technometrics*, 47(3):349–363.

Larry Wasserman and Kathryn Roeder. 2009. High-dimensional variable selection. *Annals of Statistics*, 37(5A):2178–2201.

Larry Wasserman. 2003. *All of Statistics: A Concise Course in Statistical Inference*. Springer.

Jing Wu, Bernie Devlin, Steven Ringquist, Massimo Trucco, and Kathryn Roeder. 2010. Screen and clean: A tool for identifying interactions in genome-wide association studies. *Genetic Epidemiology*, 34(3):275–285.

Qing Zhang. 2005. A Chinese yuppie in Beijing: Phonological variation and the construction of a new professional identity. *Language in Society*, 34:431–466.

Kathryn Zickuhr and Aaron Smith. 2010. 4% of online Americans use location-based services. Technical report, Pew Research Center, November.

Local and Global Algorithms for Disambiguation to Wikipedia

Lev Ratinov [1] **Dan Roth**[1] **Doug Downey**[2] **Mike Anderson**[3]

[1]University of Illinois at Urbana-Champaign
{ratinov2|danr}@uiuc.edu

[2]Northwestern University
ddowney@eecs.northwestern.edu

[3]Rexonomy
mrander@gmail.com

Abstract

Disambiguating concepts and entities in a context sensitive way is a fundamental problem in natural language processing. The comprehensiveness of Wikipedia has made the online encyclopedia an increasingly popular target for disambiguation. Disambiguation to Wikipedia is similar to a traditional Word Sense Disambiguation task, but distinct in that the Wikipedia link structure provides additional information about which disambiguations are compatible. In this work we analyze approaches that utilize this information to arrive at coherent sets of disambiguations for a given document (which we call "global" approaches), and compare them to more traditional (local) approaches. We show that previous approaches for global disambiguation can be improved, but even then the local disambiguation provides a baseline which is very hard to beat.

1 Introduction

Wikification is the task of identifying and linking expressions in text to their referent Wikipedia pages. Recently, Wikification has been shown to form a valuable component for numerous natural language processing tasks including text classification (Gabrilovich and Markovitch, 2007b; Chang et al., 2008), measuring semantic similarity between texts (Gabrilovich and Markovitch, 2007a), cross-document co-reference resolution (Finin et al., 2009; Mayfield et al., 2009), and other tasks (Kulkarni et al., 2009).

Previous studies on Wikification differ with respect to the corpora they address and the subset of expressions they attempt to link. For example, some studies focus on linking only named entities, whereas others attempt to link all "interesting" expressions, mimicking the link structure found in Wikipedia. Regardless, all Wikification systems are faced with a key *Disambiguation to Wikipedia* (D2W) task. In the D2W task, we're given a text along with explicitly identified substrings (called *mentions*) to disambiguate, and the goal is to output the corresponding Wikipedia page, if any, for each mention. For example, given the input sentence "I am visiting friends in <Chicago>," we output *http://en.wikipedia.org/wiki/Chicago* – the Wikipedia page for the city of Chicago, Illinois, and not (for example) the page for the 2002 film of the same name.

Local D2W approaches disambiguate each mention in a document separately, utilizing clues such as the textual similarity between the document and each candidate disambiguation's Wikipedia page. Recent work on D2W has tended to focus on more sophisticated *global* approaches to the problem, in which all mentions in a document are disambiguated simultaneously to arrive at a *coherent* set of disambiguations (Cucerzan, 2007; Milne and Witten, 2008b; Han and Zhao, 2009). For example, if a mention of "Michael Jordan" refers to the computer scientist rather than the basketball player, then we would expect a mention of "Monte Carlo" in the same document to refer to the statistical technique rather than the location. Global approaches utilize the Wikipedia link graph to estimate coherence.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1375–1384,
Portland, Oregon, June 19-24, 2011.

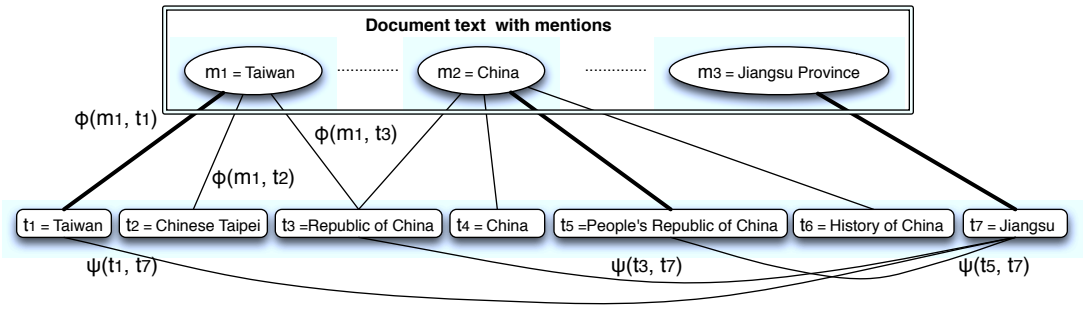


Figure 1: Sample Disambiguation to Wikipedia problem with three mentions. The mention "Jiangsu" is unambiguous. The correct mapping from mentions to titles is marked by heavy edges

In this paper, we analyze global and local approaches to the D2W task. Our contributions are as follows: (1) We present a formulation of the D2W task as an optimization problem with local and global variants, and identify the strengths and the weaknesses of each, (2) Using this formulation, we present a new global D2W system, called GLOW. In experiments on existing and novel D2W data sets,[1] GLOW is shown to outperform the previous state-of-the-art system of (Milne and Witten, 2008b), (3) We present an error analysis and identify the key remaining challenge: determining when mentions refer to concepts *not* captured in Wikipedia.

2 Problem Definition and Approach

We formalize our *Disambiguation to Wikipedia* (D2W) task as follows. We are given a document d with a set of mentions $M = \{m_1, \ldots, m_N\}$, and our goal is to produce a mapping from the set of mentions to the set of Wikipedia titles $W = \{t_1, \ldots, t_{|W|}\}$. Often, mentions correspond to a concept *without* a Wikipedia page; we treat this case by adding a special *null* title to the set W.

The D2W task can be visualized as finding a many-to-one matching on a bipartite graph, with mentions forming one partition and Wikipedia titles the other (see Figure 1). We denote the output matching as an N-tuple $\Gamma = (t_1, \ldots, t_N)$ where t_i is the output disambiguation for mention m_i.

[1]The data sets are available for download at http://cogcomp.cs.illinois.edu/Data

2.1 Local and Global Disambiguation

A *local* D2W approach disambiguates each mention m_i separately. Specifically, let $\phi(m_i, t_j)$ be a score function reflecting the likelihood that the candidate title $t_j \in W$ is the correct disambiguation for $m_i \in M$. A local approach solves the following optimization problem:

$$\Gamma^*_{\text{local}} = \arg\max_{\Gamma} \sum_{i=1}^{N} \phi(m_i, t_i) \quad (1)$$

Local D2W approaches, exemplified by (Bunescu and Pasca, 2006) and (Mihalcea and Csomai, 2007), utilize ϕ functions that assign higher scores to titles with content similar to that of the input document.

We expect, all else being equal, that the correct disambiguations will form a "coherent" set of related concepts. Global approaches define a coherence function ψ, and attempt to solve the following disambiguation problem:

$$\Gamma^* = \arg\max_{\Gamma} [\sum_{i=1}^{N} \phi(m_i, t_i) + \psi(\Gamma)] \quad (2)$$

The global optimization problem in Eq. 2 is NP-hard, and approximations are required (Cucerzan, 2007). The common approach is to utilize the Wikipedia link graph to obtain an estimate pairwise relatedness between titles $\psi(t_i, t_j)$ and to efficiently generate a *disambiguation context* Γ', a rough approximation to the optimal Γ^*. We then solve the easier problem:

$$\Gamma^* \approx \arg\max_{\Gamma} \sum_{i=1}^{N} [\phi(m_i, t_i) + \sum_{t_j \in \Gamma'} \psi(t_i, t_j)] \quad (3)$$

Eq. 3 can be solved by finding each t_i and then mapping m_i independently as in a local approach, but still enforces some degree of coherence among the disambiguations.

3 Related Work

Wikipedia was first explored as an information source for named entity disambiguation and information retrieval by Bunescu and Pasca (2006). There, disambiguation is performed using an SVM kernel that compares the lexical context around the ambiguous named entity to the content of the candidate disambiguation's Wikipedia page. However, since each ambiguous mention required a separate SVM model, the experiment was on a very limited scale. Mihalcea and Csomai applied Word Sense Disambiguation methods to the Disambiguation to Wikipedia task (2007). They experimented with two methods: (a) the lexical overlap between the Wikipedia page of the candidate disambiguations and the context of the ambiguous mention, and (b) training a Naive Bayes classiffier for each ambiguous mention, using the hyperlink information found in Wikipedia as ground truth. Both (Bunescu and Pasca, 2006) and (Mihalcea and Csomai, 2007) fall into the local framework.

Subsequent work on Wikification has stressed that assigned disambiguations for the same document should be related, introducing the global approach (Cucerzan, 2007; Milne and Witten, 2008b; Han and Zhao, 2009; Ferragina and Scaiella, 2010). The two critical components of a global approach are the semantic relatedness function ψ between two titles, and the disambiguation context Γ'. In (Milne and Witten, 2008b), the semantic context is defined to be a set of "unambiguous surface forms" in the text, and the title relatedness ψ is computed as Normalized Google Distance (NGD) (Cilibrasi and Vitanyi, 2007).[2] On the other hand, in (Cucerzan, 2007) the disambiguation context is taken to be all plausible disambiguations of the named entities in the text, and title relatedness is based on the overlap in categories and incoming links. Both approaches have limitations. The first approach relies on the presence of unambiguous mentions in the input document, and the second approach inevitably adds irrelevant titles to the disambiguation context. As we demonstrate in our experiments, by utilizing a more accurate disambiguation context, GLOW is able to achieve better performance.

[2](Milne and Witten, 2008b) also weight each mention in Γ' by its estimated disambiguation utility, which can be modeled by augmenting ψ on per-problem basis.

4 System Architecture

In this section, we present our global D2W system, which solves the optimization problem in Eq. 3. We refer to the system as GLOW, for Global Wikification. We use GLOW as a test bed for evaluating local and global approaches for D2W. GLOW combines a powerful local model ϕ with an novel method for choosing an accurate disambiguation context Γ', which as we show in our experiments allows it to outperform the previous state of the art.

We represent the functions ϕ and ψ as weighted sums of features. Specifically, we set:

$$\phi(m,t) = \sum_i w_i \phi_i(m,t) \tag{4}$$

where each feature $\phi_i(m,t)$ captures some aspect of the relatedness between the mention m and the Wikipedia title t. Feature functions $\psi_i(t,t')$ are defined analogously. We detail the specific feature functions utilized in GLOW in following sections. The coefficients w_i are learned using a Support Vector Machine over bootstrapped training data from Wikipedia, as described in Section 4.5.

At a high level, the GLOW system optimizes the objective function in Eq. 3 in a two-stage process. We first execute a *ranker* to obtain the best non-null disambiguation for each mention in the document, and then execute a *linker* that decides whether the mention should be linked to Wikipedia, or whether instead switching the top-ranked disambiguation to *null* improves the objective function. As our experiments illustrate, the linking task is the more challenging of the two by a significant margin.

Figure 2 provides detailed pseudocode for GLOW. Given a document d and a set of mentions M, we start by augmenting the set of mentions with all phrases in the document that *could* be linked to Wikipedia, but were not included in M. Introducing these additional mentions provides context that may be informative for the global coherence computation (it has no effect on local approaches). In the second

Algorithm: Disambiguate to Wikipedia
Input: document d, Mentions $M = \{m_1, \ldots, m_N\}$
Output: a disambiguation $\Gamma = (t_1, \ldots, t_N)$.
1) Let M' = $M \cup$ { Other potential mentions in d}
2) For each mention $m'_i \in M'$, construct a set of disambiguation candidates $T_i = \{t^i_1, \ldots, t^i_{k_i}\}, t^i_j \neq$ null
3) **Ranker***: Find a solution $\Gamma = (t'_1, \ldots, t'_{|M'|})$, where $t'_i \in T_i$ is the best non-null disambiguation of m'_i.*
4) **Linker***: For each m'_i, map t'_i to* null *in* Γ iff *doing so improves the objective function*
5) Return Γ entries for the original mentions M.

Figure 2: High-level pseudocode for GLOW.

Baseline Feature: $P(t\|m)$, $P(t)$
Local Features: $\phi_i(t, m)$
cosine-sim(Text(t),Text(m)) : Naive/Reweighted
cosine-sim(Text(t),Context(m)): Naive/Reweighted
cosine-sim(Context(t),Text(m)): Naive/Reweighted
cosine-sim(Context(t),Context(m)): Naive/Reweighted
Global Features: $\psi_i(t_i, t_j)$
$I_{[t_i - t_j]}$**PMI(InLinks(t_i),InLinks(t_j)) : avg/max*
$I_{[t_i - t_j]}$**NGD(InLinks(t_i),InLinks(t_j)) : avg/max*
$I_{[t_i - t_j]}$**PMI(OutLinks(t_i),OutLinks(t_j)) : avg/max*
$I_{[t_i - t_j]}$**NGD(OutLinks(t_i),OutLinks(t_j)) : avg/max*
$I_{[t_i \leftrightarrow t_j]}$ *: avg/max*
$I_{[t_i \leftrightarrow t_j]}$**PMI(InLinks($t_i$),InLinks($t_j$)) : avg/max*
$I_{[t_i \leftrightarrow t_j]}$**NGD(InLinks($t_i$),InLinks($t_j$)) : avg/max*
$I_{[t_i \leftrightarrow t_j]}$**PMI(OutLinks($t_i$),OutLinks($t_j$)) : avg/max*
$I_{[t_i \leftrightarrow t_j]}$**NGD(OutLinks($t_i$),OutLinks($t_j$)) : avg/max*

Table 1: Ranker features. $I_{[t_i - t_j]}$ is an indicator variable which is 1 iff t_i links to t_j or vise-versa. $I_{[t_i \leftrightarrow t_j]}$ is 1 iff the titles point to each other.

step, we construct for each mention m_i a limited set of candidate Wikipedia titles T_i that m_i may refer to. Considering only a small subset of Wikipedia titles as potential disambiguations is crucial for tractability (we detail which titles are selected below). In the third step, the ranker outputs the most appropriate non-null disambiguation t_i for each mention m_i.

In the final step, the linker decides whether the top-ranked disambiguation is correct. The disambiguation (m_i, t_i) may be incorrect for several reasons: (1) mention m_i does not have a corresponding Wikipedia page, (2) m_i does have a corresponding Wikipedia page, but it was not included in T_i, or (3) the ranker erroneously chose an incorrect disambiguation over the correct one.

In the below sections, we describe each step of the GLOW algorithm, and the local and global features utilized, in detail. Because we desire a system that can process documents at scale, each step requires trade-offs between accuracy and efficiency.

4.1 Disambiguation Candidates Generation

The first step in GLOW is to extract all mentions that can refer to Wikipedia titles, and to construct a set of disambiguation candidates for each mention. Following previous work, we use Wikipedia hyperlinks to perform these steps. GLOW utilizes an *anchor-title* index, computed by crawling Wikipedia, that maps each distinct hyperlink anchor text to its target Wikipedia titles. For example, the anchor text "Chicago" is used in Wikipedia to refer both to the city in Illinois and to the movie. Anchor texts in the index that appear in document d are used to supplement the mention set M in Step 1 of the GLOW algorithm in Figure 2. Because checking *all* substrings in the input text against the index is computationally inefficient, we instead prune the search space by applying a publicly available shallow parser and named entity recognition system.[3] We consider only the expressions marked as named entities by the NER tagger, the noun-phrase chunks extracted by the shallow parser, and all sub-expressions of up to 5 tokens of the noun-phrase chunks.

To retrieve the disambiguation candidates T_i for a given mention m_i in Step 2 of the algorithm, we query the anchor-title index. T_i is taken to be the set of titles most frequently linked to with anchor text m_i in Wikipedia. For computational efficiency, we utilize only the top 20 most frequent target pages for the anchor text; the accuracy impact of this optimization is analyzed in Section 6.

From the anchor-title index, we compute two local features $\phi_i(m, t)$. The first, $P(t|m)$, is the fraction of times the title t is the target page for an anchor text m. This single feature is a very reliable indicator of the correct disambiguation (Fader et al., 2009), and we use it as a baseline in our experiments. The second, $P(t)$, gives the fraction of all Wikipedia articles that link to t.

4.2 Local Features ϕ

In addition to the two baseline features mentioned in the previous section, we compute a set of text-based

[3] Available at http://cogcomp.cs.illinois.edu/page/software.

local features $\phi(t, m)$. These features capture the intuition that a given Wikipedia title t is more likely to be referred to by mention m appearing in document d if the Wikipedia page for t has high textual similarity to d, or if the context surrounding hyperlinks to t are similar to m's context in d.

For each Wikipedia title t, we construct a top-200 token TF-IDF summary of the Wikipedia page t, which we denote as $Text(t)$ and a top-200 token TF-IDF summary of the context within which t was hyperlinked to in Wikipedia, which we denote as $Context(t)$. We keep the IDF vector for all tokens in Wikipedia, and given an input mention m in a document d, we extract the TF-IDF representation of d, which we denote $Text(d)$, and a TF-IDF representation of a 100-token window around m, which we denote $Context(m)$. This allows us to define four local features described in Table 1.

We additionally compute *weighted* versions of the features described above. Error analysis has shown that in many cases the summaries of the different disambiguation candidates for the same surface form s were very similar. For example, consider the disambiguation candidates of "China' and their TF-IDF summaries in Figure 1. The majority of the terms selected in *all* summaries refer to the general issues related to China, such as *"legalism, reform, military, control, etc."*, while a minority of the terms actually allow disambiguation between the candidates. The problem stems from the fact that the TF-IDF summaries are constructed against the entire Wikipedia, and not against the confusion set of disambiguation candidates of m. Therefore, we *re-weigh* the TF-IDF vectors using the TF-IDF scheme on the disambiguation candidates as a ad-hoc document collection, similarly to an approach in (Joachims, 1997) for classifying documents. In our scenario, the TF of the a token is the original TF-IDF summary score (a real number), and the IDF term is the sum of all the TF-IDF scores for the token within the set of disambiguation candidates for m. This adds 4 more "reweighted local" features in Table 1.

4.3 Global Features ψ

Global approaches require a disambiguation context Γ' and a relatedness measure ψ in Eq. 3. In this section, we describe our method for generating a disambiguation context, and the set of global features $\psi_i(t, t')$ forming our relatedness measure.

In previous work, Cucerzan defined the disambiguation context as the union of disambiguation candidates for all the named entity mentions in the input document (2007). The disadvantage of this approach is that irrelevant titles are inevitably added to the disambiguation context, creating noise. Milne and Witten, on the other hand, use a set of unambiguous mentions (2008b). This approach utilizes only a fraction of the available mentions for context, and relies on the presence of unambiguous mentions with high disambiguation utility. In GLOW, we utilize a simple and efficient alternative approach: we first train a local disambiguation system, and then use the predictions of that system as the disambiguation context. The advantage of this approach is that unlike (Milne and Witten, 2008b) we use all the available mentions in the document, and unlike (Cucerzan, 2007) we reduce the amount of irrelevant titles in the disambiguation context by taking only the top-ranked disambiguation per mention.

Our global features are refinements of previously proposed semantic relatedness measures between Wikipedia titles. We are aware of two previous methods for estimating the relatedness between two Wikipedia concepts: (Strube and Ponzetto, 2006), which uses category overlap, and (Milne and Witten, 2008a), which uses the incoming link structure. Previous work experimented with two relatedness measures: NGD, and Specificity-weighted Cosine Similarity. Consistent with previous work, we found NGD to be the better-performing of the two. Thus we use only NGD along with a well-known Pontwise Mutual Information (PMI) relatedness measure. Given a Wikipedia title collection W, titles t_1 and t_2 with a set of incoming links L_1, and L_2 respectively, PMI and NGD are defined as follows:

$$NGD(L_1, L_2) = \frac{Log(Max(|L_1|, |L_2|)) - Log(|L_1 \cap L_2|)}{Log(|W|) - Log(Min(|L_1|, |L_2|))}$$

$$PMI(L_1, L_2) = \frac{|L_1 \cap L_2|/|W|}{|L_1|/|W||L_2|/|W|}$$

The NGD and the PMI measures can also be computed over the set of *outgoing* links, and we include these as features as well. We also included a feature indicating whether the articles each link to one

another. Lastly, rather than taking the sum of the relatedness scores as suggested by Eq. 3, we use two features: the average and the maximum relatedness to Γ'. We expect the average to be informative for many documents. The intuition for also including the maximum relatedness is that for longer documents that may cover many different subtopics, the maximum may be more informative than the average.

We have experimented with other semantic features, such as category overlap or cosine similarity between the TF-IDF summaries of the titles, but these did not improve performance in our experiments. The complete set of global features used in GLOW is given in Table 1.

4.4 Linker Features

Given the mention m and the top-ranked disambiguation t, the linker attempts to decide whether t is indeed the correct disambiguation of m. The linker includes the same features as the ranker, plus additional features we expect to be particularly relevant to the task. We include the confidence of the ranker in t with respect to second-best disambiguation t', intended to estimate whether the ranker may have made a mistake. We also include several properties of the mention m: the entropy of the distribution $P(t|m)$, the percent of Wikipedia titles in which m appears hyperlinked versus the percent of times m appears as plain text, whether m was detected by NER as a named entity, and a Good-Turing estimate of how likely m is to be out-of-Wikipedia concept based on the counts in $P(t|m)$.

4.5 Linker and Ranker Training

We train the coefficients for the ranker features using a linear Ranking Support Vector Machine, using training data gathered from Wikipedia. Wikipedia links are considered gold-standard links for the training process. The methods for compiling the Wikipedia training corpus are given in Section 5.

We train the linker as a separate linear Support Vector Machine. Training data for the linker is obtained by applying the ranker on the training set. The mentions for which the top-ranked disambiguation did not match the gold disambiguation are treated as negative examples, while the mentions the ranker got correct serve as positive examples.

Mentions/Distinct titles			
data set	Gold	Identified	Solvable
ACE	257/255	213/212	185/184
MSNBC	747/372	530/287	470/273
AQUAINT	727/727	601/601	588/588
Wikipedia	928/813	855/751	843/742

Table 2: Number of mentions and corresponding distinct titles by data set. Listed are (number of mentions)/(number of distinct titles) for each data set, for each of three mention types. *Gold* mentions include all disambiguated mentions in the data set. *Identified* mentions are gold mentions whose correct disambiguations exist in GLOW's author-title index. *Solvable* mentions are identified mentions whose correct disambiguations are among the candidates selected by GLOW (see Table 3).

5 Data sets and Evaluation Methodology

We evaluate GLOW on four data sets, of which two are from previous work. The first data set, from (Milne and Witten, 2008b), is a subset of the *AQUAINT* corpus of newswire text that is annotated to mimic the hyperlink structure in Wikipedia. That is, only the first mentions of "important" titles were hyperlinked. Titles deemed uninteresting and redundant mentions of the same title are not linked. The second data set, from (Cucerzan, 2007), is taken from *MSNBC* news and focuses on disambiguating named entities after running NER and co-reference resolution systems on newsire text. In this case, *all* mentions of all the detected named entities are linked.

We also constructed two additional data sets. The first is a subset of the *ACE* co-reference data set, which has the advantage that mentions and their types are given, and the co-reference is resolved. We asked annotators on Amazon's Mechanical Turk to link the first nominal mention of each co-reference chain to Wikipedia, if possible. Finding the accuracy of a majority vote of these annotations to be approximately 85%, we manually corrected the annotations to obtain ground truth for our experiments. The second data set we constructed, *Wiki*, is a sample of paragraphs from Wikipedia pages. Mentions in this data set correspond to existing hyperlinks in the Wikipedia text. Because Wikipedia editors explicitly link mentions to Wikipedia pages, their anchor text tends to match the title of the linked-to-page—as a result, in the overwhelming majority of

cases, the disambiguation decision is as trivial as string matching. In an attempt to generate more challenging data, we extracted 10,000 random paragraphs for which choosing the top disambiguation according to $P(t|m)$ results in at least a 10% ranker error rate. 40 paragraphs of this data was utilized for testing, while the remainder was used for training.

The data sets are summarized in Table 2. The table shows the number of annotated mentions which were hyperlinked to *non-null* Wikipedia pages, and the number of titles in the documents (without counting repetitions). For example, the AQUAINT data set contains 727 mentions,[4] all of which refer to distinct titles. The MSNBC data set contains 747 mentions mapped to non-null Wikipedia pages, but some mentions within the same document refer to the same titles. There are 372 titles in the data set, when multiple instances of the same title within one document are not counted.

To isolate the performance of the individual components of GLOW, we use multiple distinct metrics for evaluation. *Ranker accuracy*, which measures the performance of the ranker alone, is computed only over those mentions with a non-null gold disambiguation that appears in the candidate set. It is equal to the fraction of these mentions for which the ranker returns the correct disambiguation. Thus, a perfect ranker should achieve a ranker accuracy of 1.0, irrespective of limitations of the candidate generator. *Linker accuracy* is defined as the fraction of *all* mentions for which the linker outputs the correct disambiguation (note that, when the title produced by the ranker is incorrect, this penalizes linker accuracy). Lastly, we evaluate our whole system against other baselines using a previously-employed "bag of titles" (BOT) evaluation (Milne and Witten, 2008b). In BOT, we compare the set of titles output for a document with the gold set of titles for that document (ignoring duplicates), and utilize standard precision, recall, and F1 measures.

In BOT, the set of titles is collected from the mentions hyperlinked in the gold annotation. That is, if the gold annotation is { *(China, People's Republic of China), (Taiwan, Taiwan), (Jiangsu, Jiangsu)*} and the predicted anotation is: { *(China, People's Republic of China), (China, History of China), (Taiwan, null), (Jiangsu, Jiangsu), (republic, Government)*} , then the BOT for the gold annotation is: {*People's Republic of China, Taiwan, Jiangsu*} , and the BOT for the predicted annotation is: {*People's Republic of China, History of China, Jiangsu*} . The title *Government* is not included in the BOT for predicted annotation, because its associate mention *republic* did not appear as a mention in the gold annotation. Both the precision and the recall of the above prediction is 0.66. We note that in the BOT evaluation, following (Milne and Witten, 2008b) we consider all the titles within a document, even if some the titles were due to mentions we failed to identify.[5]

[4]The data set contains votes on how important the mentions are. We believe that the results in (Milne and Witten, 2008b) were reported on mentions which the majority of annotators considered important. In contrast, we used all the mentions.

Generated Candidates k	data sets			
	ACE	MSNBC	AQUAINT	Wiki
1	81.69	72.26	91.01	84.79
3	85.44	86.22	96.83	94.73
5	86.38	87.35	97.17	96.37
20	86.85	88.67	97.83	98.59

Table 3: Percent of "solvable" mentions as a function of the number of generated disambiguation candidates. Listed is the fraction of identified mentions m whose target disambiguation t is among the top k candidates ranked in descending order of $P(t|m)$.

6 Experiments and Results

In this section, we evaluate and analyze GLOW's performance on the D2W task. We begin by evaluating the mention detection component (Step 1 of the algorithm). The second column of Table 2 shows how many of the "non-null" mentions and corresponding titles we could successfully identify (e.g. out of 747 mentions in the MSNBC data set, only 530 appeared in our anchor-title index). Missing entities were primarily due to especially rare surface forms, or sometimes due to idiosyncratic capitalization in the corpus. Improving the number of identified mentions substantially is non-trivial; (Zhou et al., 2010) managed to successfully identify only 59 more entities than we do in the MSNBC data set, using a much more powerful detection method based on search engine query logs.

We generate disambiguation candidates for a

[5]We evaluate the mention identification stage in Section 6.

Features	Data sets ACE	MSNBC	AQUAINT	Wiki
$P(t\|m)$	94.05	81.91	93.19	85.88
$P(t\|m)$+Local				
Naive	95.67	84.04	94.38	92.76
Reweighted	96.21	85.10	**95.57**	**93.59**
All above	95.67	84.68	95.40	93.59
$P(t\|m)$+Global				
NER	96.21	84.04	94.04	89.56
Unambiguous	94.59	84.46	95.40	89.67
Predictions	**96.75**	**88.51**	**95.91**	89.79
$P(t\|m)$+Local+Global				
All features	**97.83**	**87.02**	94.38	**94.18**

Table 4: Ranker Accuracy. Bold values indicate the best performance in each feature group. The global approaches marginally outperform the local approaches on *ranker accuracy* , while combing the approaches leads to further marginal performance improvement.

Data set	Local	Global	Local+Global
ACE	$80.1 \rightarrow 82.8$	$80.6 \rightarrow 80.6$	$81.5 \rightarrow 85.1$
MSNBC	$74.9 \rightarrow 76.0$	$77.9 \rightarrow 77.9$	$76.5 \rightarrow 76.9$
AQUAINT	$93.5 \rightarrow 91.5$	$93.8 \rightarrow 92.1$	$92.3 \rightarrow 91.3$
Wiki	$92.2 \rightarrow 92.0$	$88.5 \rightarrow 87.2$	$92.8 \rightarrow 92.6$

Table 5: Linker performance. The notation $X \rightarrow Y$ means that when linking all mentions, the linking accuracy is X, while when applying the trained linker, the performance is Y. The local approaches are better suited for linking than the global approaches. The linking accuracy is very sensitive to domain changes.

System	ACE	MSNBC	AQUAINT	Wiki
Baseline: $P(t\|m)$	69.52	72.83	82.67	81.77
GLOW Local	75.60	74.39	84.52	90.20
GLOW Global	74.73	74.58	84.37	86.62
GLOW	77.25	74.88	83.94	90.54
M&W	72.76	68.49	83.61	80.32

Table 6: End systems performance - BOT F1. The performance of the full system (GLOW) is similar to that of the local version. GLOW outperforms (Milne and Witten, 2008b) on all data sets.

mention m using an anchor-title index, choosing the 20 titles with maximal $P(t|m)$. Table 3 evaluates the accuracy of this generation policy. We report the percent of mentions for which the correct disambiguation is generated in the top k candidates (called "solvable" mentions). We see that the baseline prediction of choosing the disambiguation t which maximizes $P(t|m)$ is very strong (80% of the correct mentions have maximal $P(t|m)$ in all data sets except MSNBC). The fraction of solvable mentions increases until about five candidates per mention are generated, after which the increase is rather slow. Thus, we believe choosing a limit of 20 candidates per mention offers an attractive trade-off of accuracy and efficiency. The last column of Table 2 reports the number of solvable mentions and the corresponding number of titles with a cutoff of 20 disambiguation candidates, which we use in our experiments.

Next, we evaluate the accuracy of the ranker. Table 4 compares the ranker performance with baseline, local and global features. The reweighted local features outperform the unweighted ("Naive") version, and the global approach outperforms the local approach on all data sets except Wikipedia. As the table shows, our approach of defining the disambiguation context to be the predicted disambiguations of a simpler local model ("Predictions") performs better than using NER entities as in (Cucerzan, 2007), or only the unambiguous entities as in (Milne and Witten, 2008b).[6] Combining the local and the global approaches typically results in minor improvements.

While the global approaches are most effective for ranking, the linking problem has different characteristics as shown in Table 5. We can see that the global features are not helpful in general for predicting whether the top-ranked disambiguation is indeed the correct one.

Further, although the trained linker improves accuracy in some cases, the gains are marginal—and the linker decreases performance on some data sets. One explanation for the decrease is that the linker is trained on Wikipedia, but is being tested on non-Wikipedia text which has different characteristics. However, in separate experiments we found that training a linker on out-of-Wikipedia text only increased test set performance by approximately 3 percentage points. Clearly, while ranking accuracy is high overall, different strategies are needed to achieve consistently high linking performance.

A few examples from the ACE data set help il-

[6] In NER we used only the top prediction, because using all candidates as in (Cucerzan, 2007) proved prohibitively inefficient.

lustrate the tradeoffs between local and global features in GLOW. The global system mistakenly links *"<Dorothy Byrne>, a state coordinator for the Florida Green Party, said ..."* to the British journalist, because the journalist sense has high coherence with other mentions in the newswire text. However, the local approach correctly maps the mention to $null$ because of a lack of local contextual clues. On the other hand, in the sentence *"Instead of Los Angeles International, for example, consider flying into <Burbank> or John Wayne Airport in Orange County, Calif."*, the local ranker links the mention *Burbank* to *Burbank,_California*, while the global system correctly maps the entity to *Bob_Hope_Airport*, because the three airports mentioned in the sentence are highly related to one another.

Lastly, in Table 6 we compare the end system BOT F1 performance. The local approach proves a very competitive baseline which is hard to beat. Combining the global and the local approach leads to marginal improvements. The full GLOW system outperforms the existing state-of-the-art system from (Milne and Witten, 2008b), denoted as M&W, on all data sets. We also compared our system with the recent TAGME Wikification system (Ferragina and Scaiella, 2010). However, TAGME is designed for a different setting than ours: extremely short texts, like Twitter posts. The TAGME RESTful API was unable to process some of our documents at once. We attempted to input test documents one sentence at a time, disambiguating each sentence independently, which resulted in poor performance (0.07 points in F1 lower than the $P(t|m)$ baseline). This happened mainly because the same mentions were linked to different titles in different sentences, leading to low precision.

An important question is why M&W underperforms the baseline on the MSNBC and Wikipedia data sets. In an error analysis, M&W performed poorly on the MSNBC data not due to poor disambiguations, but instead because the data set contains only named entities, which were often delimited incorrectly by M&W. Wikipedia was challenging for a different reason: M&W performs less well on the short (one paragraph) texts in that set, because they contain relatively few of the unambiguous entities the system relies on for disambiguation.

7 Conclusions

We have formalized the *Disambiguation to Wikipedia* (D2W) task as an optimization problem with local and global variants, and analyzed the strengths and weaknesses of each. Our experiments revealed that previous approaches for global disambiguation can be improved, but even then the local disambiguation provides a baseline which is very hard to beat.

As our error analysis illustrates, the primary remaining challenge is determining when a mention does *not* have a corresponding Wikipedia page. Wikipedia's hyperlinks offer a wealth of disambiguated mentions that can be leveraged to train a D2W system. However, when compared with mentions from general text, Wikipedia mentions are disproportionately likely to have corresponding Wikipedia pages. Our initial experiments suggest that accounting for this bias requires more than simply training a D2W system on a moderate number of examples from non-Wikipedia text. Applying distinct semi-supervised and active learning approaches to the task is a primary area of future work.

Acknowledgments

This research supported by the Army Research Laboratory (ARL) under agreement W911NF-09-2-0053 and by the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. The third author was supported by a Microsoft New Faculty Fellowship. Any opinions, findings, conclusions or recommendations are those of the authors and do not necessarily reflect the view of the ARL, DARPA, AFRL, or the US government.

References

R. Bunescu and M. Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06), Trento, Italy*, pages 9–16, April.

Ming-Wei Chang, Lev Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of semantic representation: dataless classification. In *Proceedings of the*

23rd national conference on Artificial intelligence - Volume 2, pages 830–835. AAAI Press.

Rudi L. Cilibrasi and Paul M. B. Vitanyi. 2007. The google similarity distance. *IEEE Trans. on Knowl. and Data Eng.*, 19(3):370–383.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic, June. Association for Computational Linguistics.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2009. Scaling wikipedia-based named entity disambiguation to arbitrary web text. In *Proceedings of the WikiAI 09 - IJCAI Workshop: User Contributed Knowledge and Artificial Intelligence: An Evolving Synergy*, Pasadena, CA, USA, July.

Paolo Ferragina and Ugo Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In Jimmy Huang, Nick Koudas, Gareth J. F. Jones, Xindong Wu, Kevyn Collins-Thompson, and Aijun An, editors, *Proceedings of the 19th ACM conference on Information and knowledge management*, pages 1625–1628. ACM.

Tim Finin, Zareen Syed, James Mayfield, Paul McNamee, and Christine Piatko. 2009. Using Wikitology for Cross-Document Entity Coreference Resolution. In *Proceedings of the AAAI Spring Symposium on Learning by Reading and Learning to Read*. AAAI Press, March.

Evgeniy Gabrilovich and Shaul Markovitch. 2007a. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 1606–1611, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Evgeniy Gabrilovich and Shaul Markovitch. 2007b. Harnessing the expertise of 70,000 human editors: Knowledge-based feature generation for text categorization. *J. Mach. Learn. Res.*, 8:2297–2345, December.

Xianpei Han and Jun Zhao. 2009. Named entity disambiguation by leveraging wikipedia semantic knowledge. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 215–224, New York, NY, USA. ACM.

Thorsten Joachims. 1997. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 143–151, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 457–466, New York, NY, USA. ACM.

James Mayfield, David Alexander, Bonnie Dorr, Jason Eisner, Tamer Elsayed, Tim Finin, Clay Fink, Marjorie Freedman, Nikesh Garera, James Mayfield, Paul McNamee, Saif Mohammad, Douglas Oard, Christine Piatko, Asad Sayeed, Zareen Syed, and Ralph Weischede. 2009. Cross-Document Coreference Resolution: A Key Technology for Learning by Reading. In *Proceedings of the AAAI 2009 Spring Symposium on Learning by Reading and Learning to Read*. AAAI Press, March.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 233–242, New York, NY, USA. ACM.

David Milne and Ian H. Witten. 2008a. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *In the Wikipedia and AI Workshop of AAAI*.

David Milne and Ian H. Witten. 2008b. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 509–518, New York, NY, USA. ACM.

Michael Strube and Simone Paolo Ponzetto. 2006. Wikirelate! computing semantic relatedness using wikipedia. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, pages 1419–1424. AAAI Press.

Yiping Zhou, Lan Nie, Omid Rouhani-Kalleh, Flavian Vasile, and Scott Gaffney. 2010. Resolving surface forms to wikipedia topics. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1335–1343, Beijing, China, August. Coling 2010 Organizing Committee.

A Stacked Sub-Word Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging

Weiwei Sun
Department of Computational Linguistics, Saarland University
German Research Center for Artificial Intelligence (DFKI)
D-66123, Saarbrücken, Germany
wsun@coli.uni-saarland.de

Abstract

The large combined search space of joint word segmentation and Part-of-Speech (POS) tagging makes efficient decoding very hard. As a result, effective high order features representing rich contexts are inconvenient to use. In this work, we propose a novel stacked sub-word model for this task, concerning both efficiency and effectiveness. Our solution is a two step process. First, one word-based segmenter, one character-based segmenter and one local character classifier are trained to produce coarse segmentation and POS information. Second, the outputs of the three predictors are merged into sub-word sequences, which are further bracketed and labeled with POS tags by a fine-grained sub-word tagger. The coarse-to-fine search scheme is efficient, while in the sub-word tagging step rich contextual features can be approximately derived. Evaluation on the Penn Chinese Treebank shows that our model yields improvements over the best system reported in the literature.

1 Introduction

Word segmentation and part-of-speech (POS) tagging are necessary initial steps for more advanced Chinese language processing tasks, such as parsing and semantic role labeling. Joint approaches that resolve the two tasks simultaneously have received much attention in recent research. Previous work has shown that joint solutions led to accuracy improvements over pipelined systems by avoiding segmentation error propagation and exploiting POS information to help segmentation. A challenge for joint approaches is the large combined search space, which makes efficient decoding and structured learning of parameters very hard. Moreover, the representation ability of models is limited since using rich contextual word features makes the search intractable. To overcome such efficiency and effectiveness limitations, the approximate inference and reranking techniques have been explored in previous work (Zhang and Clark, 2010; Jiang et al., 2008b).

In this paper, we present an effective and efficient solution for joint Chinese word segmentation and POS tagging. Our work is motivated by several characteristics of this problem. First of all, a majority of words are easy to identify in the segmentation problem. For example, a simple maximum matching segmenter can achieve an f-score of about 90. We will show that it is possible to improve the efficiency and accuracy by using different strategies for different words. Second, segmenters designed with different views have complementary strength. We argue that the agreements and disagreements of different solvers can be used to construct an intermediate sub-word structure for joint segmentation and tagging. Since the sub-words are large enough in practice, the decoding for POS tagging over sub-words is efficient. Finally, the Chinese language is characterized by the lack of morphology that often provides important clues for POS tagging, and the POS tags contain much syntactic information, which need context information within a large window for disambiguation. For example, Huang et al. (2007) showed the effectiveness of utilizing syntactic information to rerank POS tagging results. As a result, the capability to represent rich contextual features is crucial to a POS tagger. In this work, we use a representation-efficiency tradeoff through stacked learning, a way of approximating rich *non-local* fea-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1385–1394,
Portland, Oregon, June 19-24, 2011.

tures.

This paper describes a novel stacked sub-word model. Given multiple word segmentations of one sentence, we formally define a sub-word structure that maximizes the agreement of non-word-break positions. Based on the sub-word structure, joint word segmentation and POS tagging is addressed as a two step process. In the first step, one word-based segmenter, one character-based segmenter and one local character classifier are used to produce coarse segmentation and POS information. The results of the three predictors are then merged into sub-word sequences, which are further bracketed and labeled with POS tags by a fine-grained sub-word tagger. If a string is consistently segmented as a word by the three segmenters, it will be a correct word prediction with a very high probability. In the sub-word tagging phase, the fine-grained tagger mainly considers its POS tag prediction problem. For the words that are not consistently predicted, the fine-grained tagger will also consider their bracketing problem. The coarse-to-fine scheme significantly improves the efficiency of decoding. Furthermore, in the sub-word tagging step, word features in a large window can be approximately derived from the coarse segmentation and tagging results. To train a good sub-word tagger, we use the stacked learning technique, which can effectively correct the training/test mismatch problem.

We conduct our experiments on the Penn Chinese Treebank and compare our system with the state-of-the-art systems. We present encouraging results. Our system achieves an f-score of 98.17 for the word segmentation task and an f-score of 94.02 for the whole task, resulting in relative error reductions of 14.1% and 5.5% respectively over the best system reported in the literature.

The remaining part of the paper is organized as follows. Section 2 gives a brief introduction to the problem and reviews the relevant previous research. Section 3 describes the details of our method. Section 4 presents experimental results and empirical analyses. Section 5 concludes the paper.

2 Background

2.1 Problem Definition

Given a sequence of characters $\mathbf{c} = (c_1, ..., c_{\#\mathbf{c}})$, the task of word segmentation and POS tagging is to predict a sequence of word and POS tag pairs $\mathbf{y} = (\langle w_1, p_1 \rangle, \langle w_{\#\mathbf{y}}, p_{\#\mathbf{y}} \rangle)$, where w_i is a word, p_i is its POS tag, and a "#" symbol denotes the number of elements in each variable. In order to avoid error propagation and make use of POS information for word segmentation, the two tasks should resolved jointly. Previous research has shown that the integrated methods outperformed pipelined systems (Ng and Low, 2004; Jiang et al., 2008a; Zhang and Clark, 2008).

2.2 Character-Based and Word-Based Methods

Two kinds of approaches are popular for joint word segmentation and POS tagging. The first is the "character-based" approach, where basic processing units are characters which compose words. In this kind of approach, the task is formulated as the classification of characters into POS tags with boundary information. Both the *IOB2* representation (Ramshaw and Marcus, 1995) and the *Start/End* representation (Kudo and Matsumoto, 2001) are popular. For example, the label *B-NN* indicates that a character is located at the begging of a noun. Using this method, POS information is allowed to interact with segmentation. Note that word segmentation can also be formulated as a sequential classification problem to predict whether a character is located at the beginning of, inside or at the end of a word. This character-by-character method for segmentation was first proposed in (Xue, 2003), and was then further used in POS tagging in (Ng and Low, 2004). One main disadvantage of this model is the difficulty in incorporating the whole word information.

The second kind of solution is the "word-based" method, where the basic predicting units are words themselves. This kind of solver sequentially decides whether the local sequence of characters makes up a word as well as its possible POS tag. In particular, a word-based solver reads the input sentence from left to right, predicts whether the current piece of continuous characters is a word token and which class it belongs to. Solvers may use previously predicted words and their POS information as clues to find a new word. After one word is found and classified, solvers move on and search for the next possible word. This word-by-word method for segmentation was first proposed in (Zhang and Clark, 2007),

and was then further used in POS tagging in (Zhang and Clark, 2008).

In our previous work(Sun, 2010), we presented a theoretical and empirical comparative analysis of character-based and word-based methods for Chinese word segmentation. We showed that the two methods produced different distributions of segmentation errors in a way that could be explained by theoretical properties of the two models. A system combination method that leverages the complementary strength of word-based and character-based segmentation models was also successfully explored in their work. Different from our previous focus, the diversity of different models designed with different views is utilized to construct sub-word structures in this work. We will discuss the details in the next section.

2.3 Stacked Learning

Stacked generalization is a meta-learning algorithm that was first proposed in (Wolpert, 1992) and (Breiman, 1996). The idea is to include two "levels" of predictors. The first level includes one or more predictors $g_1, ...g_K : \mathbb{R}^d \rightarrow \mathbb{R}$; each receives input $\mathbf{x} \in \mathbb{R}^d$ and outputs a prediction $g_k(\mathbf{x})$. The second level consists of a single function $h : \mathbb{R}^{d+K} \rightarrow \mathbb{R}$ that takes as input $\langle \mathbf{x}, g_1(\mathbf{x}), ..., g_K(\mathbf{x}) \rangle$ and outputs a final prediction $\hat{y} = h(\mathbf{x}, g_1(\mathbf{x}), ..., g_K(\mathbf{x}))$.

Training is done as follows. The training data $S = \{(\mathbf{x}_t, \mathbf{y}_t) : t \in [1, T]\}$ is split into L equal-sized disjoint subsets $S_1, ..., S_L$. Then functions $\mathbf{g}_1, ..., \mathbf{g}_L$ (where $\mathbf{g}_l = \langle g_1^l, ..., g_K^l \rangle$) are seperately trained on $S - S_l$, and are used to construct the augmented dataset $\hat{S} = \{(\langle \mathbf{x}_t, \hat{\mathbf{y}}_t^1, ..., \hat{\mathbf{y}}_t^K \rangle, \mathbf{y}_t) : \hat{\mathbf{y}}_t^k = g_k^l(\mathbf{x}_t)$ and $\mathbf{x}_t \in S_l\}$. Finally, each g_k is trained on the original dataset and the second level predictor h is trained on $\hat{S}$. The intent of the *cross-validation* scheme is that $\mathbf{y}_t^k$ is similar to the prediction produced by a predictor which is learned on a sample that does not include $\mathbf{x}_t$.

Stacked learning has been applied as a system ensemble method in several NLP tasks, such as named entity recognition (Wu et al., 2003) and dependency parsing (Nivre and McDonald, 2008). This framework is also explored as a solution for learning *non-local* features in (Torres Martins et al., 2008). In the machine learning research, stacked learning has been applied to structured prediction (Cohen and Carvalho, 2005). In this work, stacked learning is used to acquire extended training data for sub-word tagging.

3 Method

3.1 Architecture

In our stacked sub-word model, joint word segmentation and POS tagging is decomposed into two steps: (1) coarse-grained word segmentation and tagging, and (2) fine-grained sub-word tagging. The workflow is shown in Figure 1. In the first phase, one word-based segmenter ($\mathrm{Seg_W}$) and one character-based segmenter ($\mathrm{Seg_C}$) are trained to produce word boundaries. Additionally, a *local* character-based joint segmentation and tagging solver ($\mathrm{SegTag_L}$) is used to provide word boundaries as well as inaccurate POS information. Here, the word *local* means the labels of nearby characters are not used as features. In other words, the local character classifier assumes that the tags of characters are independent of each other. In the second phase, our system first combines the three segmentation and tagging results to get sub-words which maximize the agreement about word boundaries. Finally, a fine-grained sub-word tagger (SubTag) is applied to bracket sub-words into words and also to obtain their POS tags.

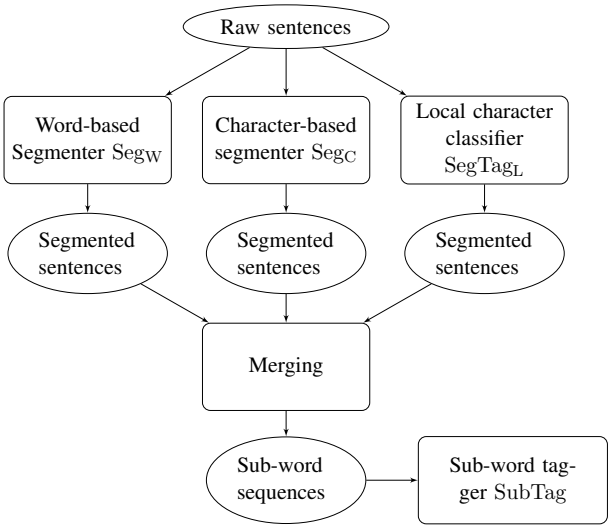


Figure 1: Workflow of the stacked sub-word model.

In our model, segmentation and POS tagging interact with each other in two processes. First, although $SegTag_L$ is locally trained, it resolves the

two sub-tasks simultaneously. Therefore, in the sub-word generating stage, segmentation and POS tagging help each other. Second, in the sub-word tagging stage, the bracketing and the classification of sub-words are jointly resolved as one sequence labeling problem.

Our experiments on the Penn Chinese Treebank will show that the word-based and character-based segmenters and the local tagger on their own produce high quality word boundaries. As a result, the oracle performance to recover words from a sub-word sequence is very high. The quality of the final tagger relies on the quality of the sub-word tagger. If a high performance sub-word tagger can be constructed, the whole task can be well resolved. The statistics will also empirically show that sub-words are significantly larger than characters and only slightly smaller than words. As a result, the search space of the sub-word tagging is significantly shrunken, and exact Viterbi decoding without approximately pruning can be efficiently processed. This property makes nearly all popular sequence labeling algorithms applicable.

Zhang et al. (2006) described a sub-word based tagging model to resolve word segmentation. To get the pieces which are larger than characters but smaller than words, they combine a character-based segmenter and a dictionary matching segmenter. Our contributions include (1) providing a formal definition of our sub-word structure that is based on multiple segmentations and (2) proposing a stacking method to acquire sub-words.

3.2 The Coarse-grained Solvers

We systematically described the implementation of two state-of-the-art Chinese word segmenters in word-based and character-based architectures, respectively (Sun, 2010). Our word-based segmenter is based on a discriminative joint model with a first order semi-Markov structure, and the other segmenter is based on a first order Markov model. Exact Viterbi-style search algorithms are used for decoding. Limited to the document length, we do not give the description of the features. We refer readers to read the above paper for details. For parameter estimation, our work adopt the *Passive-Aggressive* (PA) framework (Crammer et al., 2006), a family of margin based online learning algorithms. In this work, we introduce two simple but important refinements: (1) to shuffle the sample orders in each iteration and (2) to average the parameters in each iteration as the final parameters.

Idiom In linguistics, idioms are usually presumed to be figures of speech contradicting the principle of compositionality. As a result, it is very hard to recognize out-of-vocabulary idioms for word segmentation. However, the lexicon of idioms can be taken as a close set, which helps resolve the problem well. We collect 12992 idioms[1] from several online Chinese dictionaries. For both word-based and character-based segmentation, we first match every string of a given sentence with idioms. Every sentence is then splitted into smaller pieces which are seperated by idioms. Statistical segmentation models are then performed on these smaller character sequences.

We use a local classifier to predict the POS tag with positional information for each character. Each character can be assigned one of two possible boundary tags: "B" for a character that begins a word and "I" for a character that occurs in the middle of a word. We denote a candidate character token c_i with a fixed window $c_{i-2}c_{i-1}c_ic_{i+1}c_{i+2}$. The following features are used:

- character uni-grams: c_k $(i-2 \leq k \leq i+2)$
- character bi-grams: c_kc_{k+1} $(i-2 \leq k \leq i+1)$

To resolve the classification problem, we use the linear SVM classifier LIBLINEAR[2].

3.3 Merging Multiple Segmentation Results into Sub-Word Sequences

A majority of words are easy to identify in the segmentation problem. We favor the idea treating different words using different strategies. In this work we try to identify *simple* and *difficult* words first and to integrate them into a sub-word level. Inspired by previous work, we constructed this sub-word structure by using multiple solvers designed from different views. If a piece of continuous characters is consistently segmented by multiple segmenters, it will

[1]This resource is publicly available at http://www.coli.uni-saarland.de/~wsun/idioms.txt.

[2]Available at http://www.csie.ntu.edu.tw/~cjlin/liblinear/.

	以	总	成	绩	3	5	5	.	3	5	分	居	领	先	地	位
Answer:	[P]	[JJ]	[NN]		[			CD		]	[M]	[VV]	[JJ]		[NN]	
$\mathrm{Seg_W}$:	[]	[]	[	]	[		]	[		]	[	]	[	]	[	]
$\mathrm{Seg_C}$:	[]	[]	[	]	[						]	[]	[	]	[	]
$\mathrm{SegTag_L}$:	[P]	[JJ]	[NN]		[	CD		]	[NT]	[CD]	[NT]	[VV]	[VV]		[NN]	
Sub-words:	[P]	[JJ]	[NN]		[	B-CD	]	[I-CD]	[NT]	[CD]	[NT]	[VV]	[VV]		[NN]	

Figure 2: An example phrase: 以总成绩 3 5 5 . 3 5 分居领先地位 (Being in front with a total score of 355.35 points).

not be separated in the sub-word tagging step. The intuition is that strings which are consistently segmented by the different segmenters tend to be correct predictions. In our experiment on the Penn Chinese Treebank (Xue et al., 2005), the accuracy is 98.59% on the development data which is defined in the next section. The key point for the intermediate sub-word structures is to maximize the agreement of the three coarse-grained systems. In other words, the goal is to make merged sub-words as large as possible but not overlap with any predicted word produced by the three coarse-grained solvers. In particular, if the position between two continuous characters is predicted as a word boundary by any segmenter, this position is taken as a separation position of the sub-word sequence. This strategy makes sure that it is still possible to *re*-segment the strings of which the boundaries are disagreed with by the coarse-grained segmenters in the fine-grained tagging stage.

The formal definition is as follows. Given a sequence of characters $\mathbf{c} = (c_1, ..., c_{\#\mathbf{c}})$, let $c[i:j]$ denote a string that is made up of characters between c_i and c_j (including c_i and c_j), then a partition of the sentence can be written as $c[0:e_1], c[e_1+1:e_2], ..., c[e_m:\#\mathbf{c}]$. Let $s_k = \{c[i:j]\}$ denote the set of all segments of a partition. Given multiple partitions of a character sequence $\mathcal{S} = \{s_k\}$, there is one and only one merged partition $s_{\mathcal{S}} = \{c[i:j]\}$ s.t.

1. $\forall c[i:j] \in s_{\mathcal{S}}, \forall s_k \in \mathcal{S}, \exists c[s:e] \in s_k, s \leq i \leq j \leq e$.

2. $\forall \mathcal{C}'$ satisfies the above condition, $|\mathcal{C}'| > |\mathcal{C}|$.

The first condition makes sure that all segments in the merged partition can be only embedded in but do not overlap with any segment of any partition from $\mathcal{S}$. The second condition promises that segments of the merged partition achieve maximum length.

Figure 2 is an example to illustrate the procedure of our method. The lines $\mathrm{Seg_W}$, $\mathrm{Seg_C}$ and $\mathrm{SegTag_L}$ are the predictions of the three coarse-grained solvers. For the three words at the beginning and the two words at the end, the three predictors agree with each other. And these five words are kept as sub-words. For the character sequence " 3 5 5 . 3 5 分居", the predictions are very different. Because there are no word break predictions among the first three characters " 3 5 5 ", it is as a whole taken as one sub-word. For the other five characters, either the left position or the right position is segmented as a word break by some predictor, so the merging processor seperates them and takes each one as a single sub-word. The last line shows the merged sub-word sequence. The coarse-grained POS tags with positional information are derived from the labels provided by $\mathrm{SegTag_L}$.

3.4 The Fine-grained Sub-Word Tagger

Bracketing sub-words into words is formulated as a IOB-style sequential classification problem. Each sub-word may be assigned with one POS tag as well as two possible boundary tags: "B" for the beginning position and "I" for the middle position. A tagger is trained to classify sub-word by using the features derived from its contexts.

The sub-word level allows our system to utilize features in a large context, which is very important for POS tagging of the morphologically poor language. Features are formed making use of sub-word contents, their IOB-style inaccurate POS tags. In the following description, "C" refers to the content of the sub-word, while "T" refers to the IOB-style POS tags. For convenience, we denote a sub-word with its context $...s_{i-2}s_{i-1}s_is_{i+1}s_{i+2}...$, where s_i is

C(s_{i-1})="成绩"; T(s_{i-1})="NN" C(s_i)=" 3 5 5 "; T(s_i)="B-CD" C(s_{i+1})=". "; T(s_{i+1})="I-CD"
C(s_{i-1})C(s_i)="成绩_3 5 5 " T(s_{i-1})T(s_i)="NN_B-CD" C(s_i)C(s_{i+1})=" 3 5 5 _. " T(s_i)T(s_{i+1})="B-CD_I-CD"
C(s_{i-1})C(s_{i+1})="成绩_. " T(s_{i-1})T(s_{i+1})="B-NN_I-CD"
Prefix(1)=" 3 "; Prefix(2)=" 3 5 "; Prefix(3)=" 3 5 5 " Suffix(1)=" 5 "; Suffix(2)=" 5 5 "; Suffix(3)=" 3 5 5 "

Table 1: An example of features used in the sub-word tagging.

the current token. We denote l_C, l_T as the sizes of the window.

- Uni-gram features: C(s_k) ($-l_C \leq k \leq l_C$), T(s_k) ($-l_T \leq k \leq l_T$)
- Bi-gram features: C(s_k)C(s_{k+1}) ($-l_C \leq k \leq l_C - 1$), T(s_k)T(s_{k+1}) ($-l_T \leq k \leq l_T - 1$)
- C(s_{i-1})C(s_{i+1}) (if $l_C \geq 1$), T(s_{i-1})T(s_{i+1}) (if $l_T \geq 1$)
- T(s_{i-2})T(s_{i+1}) (if $l_T \geq 2$)
- In order to better handle unknown words, we also extract morphological features: character n-gram prefixes and suffixes for n up to 3. These features have been shown useful in previous research (Huang et al., 2007).

Take the sub-word " 3 5 5 " in Figure 2 for example, when l_C and l_T are both set to 1, all features used are listed in Table 1.

In the following experiments, we will vary window sizes l_C and l_T to find out the contribution of context information for the disambiguation. A first order Max-Margin Markov Networks model is used to resolve the sequence tagging problem. We use the SVM-HMM[3] implementation for the experiments in this work. We use the basic linear model without applying any kernel function.

[3]Available at http://www.cs.cornell.edu/People/tj/svm_light/svm_hmm.html.

Algorithm 1: The stacked learning procedure for the sub-word tagger.

input : Data $S = \{(\mathbf{c}_t, \mathbf{y}_t), t = 1, 2, ..., n\}$
Split S into L partitions $\{S_1, ...S_L\}$
for $l = 1, ..., L$ **do**
  Train $\mathrm{Seg_W}^l$, $\mathrm{Seg_C}^l$ and $\mathrm{SegTag_L}^l$ using $S - S_l$.
  Predict S_l using $\mathrm{Seg_W}^l$, $\mathrm{Seg_C}^l$ and $\mathrm{SegTag_L}^l$.
  Merge the predictions to get sub-words training sample S'_l.
end
Train the sub-word tagger SubTag using S'.

3.5 Stacked Learning for the Sub-Word Tagger

The three coarse-grained solvers $\mathrm{Seg_W}$, $\mathrm{Seg_C}$ and $\mathrm{SegTag_L}$ are directly trained on the original training data. When these three predictors are used to produce the training data, the performance is perfect. However, this does not hold when these models are applied to the test data. If we directly apply $\mathrm{Seg_W}$, $\mathrm{Seg_C}$ and $\mathrm{SegTag_L}$ to extend the training data to generate sub-word samples, the extended training data for the sub-word tagger will be very different from the data in the run time, resulting in poor performance.

One way to correct the training/test mismatch is to use the stacking method, where a K-fold *cross-validation* on the original data is performed to construct the training data for sub-word tagging. Algorithm 1 illustrates the learning procedure. First, the training data $S = \{(\mathbf{c}_t, \mathbf{y}_t)\}$ is split into L equal-sized disjoint subsets $S_1, ..., S_L$. For each subset S_l, the complementary set $S - S_l$ is used to train three coarse solvers $\mathrm{Seg_W}^l$, $\mathrm{Seg_C}^l$ and $\mathrm{SegTag_L}^l$, which process the S_l and provide inaccurate predictions. Then the inaccurate predictions are merged into sub-word sequences and S_l is extended to S'_l. Finally, the sub-word tagger is trained on the whole extended data set S'.

4 Experiments

4.1 Setting

Previous studies on joint Chinese word segmentation and POS tagging have used the Penn Chinese Treebank (CTB) in experiments. We follow this set-

ting in this paper. We use CTB 5.0 as our main corpus and define the training, development and test sets according to (Jiang et al., 2008a; Jiang et al., 2008b; Kruengkrai et al., 2009; Zhang and Clark, 2010). Table 2 shows the statistics of our experimental settings.

Data set	CTB files	# of sent.	# of words
Training	1-270	18,089	493,939
	400-931		
	1001-1151		
Devel.	301-325	350	6821
Test	271-300	348	8008

Table 2: Training, development and test data on CTB 5.0

Three metrics are used for evaluation: precision (P), recall (R) and balanced f-score (F) defined by 2PR/(P+R). Precision is the relative amount of correct words in the system output. Recall is the relative amount of correct words compared to the gold standard annotations. For segmentation, a token is considered to be correct if its boundaries match the boundaries of a word in the gold standard. For the whole task, both the boundaries and the POS tag have to be correctly identified.

4.2 Performance of the Coarse-grained Solvers

Table 3 shows the performance on the development data set of the three coarse-grained solvers. In this paper, we use 20 iterations to train $\mathrm{Seg_W}$ and $\mathrm{Seg_C}$ for all experiments. Even only locally trained, the character classifier $\mathrm{SegTag_L}$ still significantly outperforms the two state-of-the-art segmenters $\mathrm{Seg_W}$ and $\mathrm{Seg_C}$. This good performance indicates that the POS information is very important for word segmentation.

Devel.	Task	P(%)	R(%)	F
$\mathrm{Seg_W}$	Seg	94.55	94.84	94.69
$\mathrm{Seg_C}$	Seg	95.10	94.38	94.73
$\mathrm{SegTag_L}$	Seg	95.67	95.98	95.83
	Seg&Tag	87.54	91.29	89.38

Table 3: Performance of the coarse-grained solvers on the development data.

4.3 Statistics of Sub-Words

Since the base predictors to generate coarse information are two word segmenters and a local character classifier, the coarse decoding is efficient. If the length of sub-words is too short, i.e. the decoding path for sub-word sequences are too long, the decoding of the fine-grained stage is still hard. Although we cannot give a theoretical average length of sub-words, we can still show the empirical one. The average length of sub-words on the development set is 1.64, while the average length of words is 1.69. The number of all IOB-style POS tags is 59 (when using 5-fold cross-validation to generate stacked training samples). The number of all POS tags is 35. Empirically, the decoding over sub-words is $\frac{1.69}{1.64} \times (\frac{59}{35})^{n+1}$ times as slow as the decoding over words, where n is the order of the markov model. When a first order markov model is used, this number is 2.93. These statistics empirically suggest that the decoding over sub-word sequence can be efficient.

On the other hand, the sub-word sequences are not perfect in the sense that they do not promise to recover all words because of the errors made in the first step. Similarly, we can only show the empirical upper bound of the sub-word tagging. The oracle performance of the final POS tagging on the development data set is shown in Table 4. The upper bound indicates that the coarse search procedure does not lose too much.

Task	P(%)	R(%)	F
Seg&Tag	99.50%	99.09%	99.29

Table 4: Upper bound of the sub-word tagging on the development data.

One main disadvantage of character-based approach is the difficulty to incorporate word features. Since the sub-words are on average close to words, sub-word features are good approximations of word features.

4.4 Rich Contextual Features Are Useful

Table 5 shows the effect that features within different window size has on the sub-word tagging task. In this table, the symbol "C" means sub-word content features while the symbol "T" means IOB-style POS tag features. The number indicates the length

Devel.		P(%)	R(%)	F
C:±0	T:±0	92.52	92.83	92.67
C:±1	T:±0	92.63	93.27	92.95
C:±1	T:±1	92.62	93.05	92.83
C:±2	T:±0	93.17	93.86	93.51
C:±2	T:±1	93.27	93.64	93.45
C:±2	T:±2	93.08	93.61	93.34
C:±3	T:±0	93.12	93.86	93.49
C:±3	T:±1	93.34	93.96	93.65
C:±3	T:±2	93.34	93.96	93.65

Table 5: Performance of the stacked sub-word model ($K = 5$) with features in different window sizes.

of the window. For example, "C:±1" means that the tagger uses one preceding sub-word and one succeeding sub-word as features. From this table, we can clearly see the impact of features derived from neighboring sub-words. There is a significant increase between "C:±2" and "C:±1" models. This confirms our motivation that longer history and future features are crucial to the Chinese POS tagging problem. It is the main advantage of our model that making rich contextual features applicable. In all previous solutions, only features within a short history can be used due to the efficiency limitation.

The performance is further slightly improved when the window size is increased to 3. Using the labeled bracketing f-score, the evaluation shows that the "C:±3 T:±1" model performs the same as the "C:±3 T:±2" model. However, the sub-word classification accuracy of the "C:±3 T:±1" model is higher, so in the following experiments and the final results reported on the test data set, we choose this setting.

This table also suggests that the IOB-style POS information of sub-words does not contribute. We think there are two main reasons: (1) The POS information provided by the local classifier is inaccurate; (2) The structured learning of the sub-word tagger can use *real predicted* sub-word labels during its decoding time, since this learning algorithm does inference during the training time. It is still an open question whether more accurate POS information in rich contexts can help this task. If the answer is *YES*, how can we efficiently incorporate these features?

4.5 Stacked Learning Is Useful

Table 6 compares the performance of "C:±3 T:±1" models trained with no stacking as well as different folds of cross-validation. We can see that although it is still possible to improve the segmentation and POS tagging performance compared to the local character classifier, the whole task just benefits only a little from the sub-word tagging procedure if the stacking technique is not applied. The stacking technique can significantly improve the system performance, both for segmentation and POS tagging. This experiment confirms the theoretical motivation of using stacked learning: simulating the test-time setting when a sub-word tagger is applied to a new instance. There is not much difference between the 5-fold and the 10-fold cross-validation.

Devel.	Task	P(%)	R(%)	F
No stacking	Seg	95.75	96.48	96.12
	Seg&Tag	91.42	92.13	91.77
$K = 5$	Seg	96.42	97.04	96.73
	Seg&Tag	93.34	93.96	93.65
$K = 10$	Seg	96.67	97.11	96.89
	Seg&Tag	93.50	94.06	93.78

Table 6: Performance on the development data. No stacking and different folds of cross-validation are separately applied.

4.6 Final Results

Table 7 summarizes the performance of our final system on the test data and other systems reported in a majority of previous work. The final results of our system are achieved by using 10-fold cross-validation "C:±3 T:±1" models. The left most column indicates the reference of previous systems that represent state-of-the-art results. The comparison of the accuracy between our stacked sub-word system and the state-of-the-art systems in the literature indicates that our method is competitive with the best systems. Our system obtains the highest f-score performance on both segmentation and the whole task, resulting in error reductions of 14.1% and 5.5% respectively.

Test	Seg	Seg&Tag
(Jiang et al., 2008a)	97.85	93.41
(Jiang et al., 2008b)	97.74	93.37
(Kruengkrai et al., 2009)	97.87	93.67
(Zhang and Clark, 2010)	97.78	93.67
Our system	**98.17**	**94.02**

Table 7: F-score performance on the test data.

5 Conclusion and Future Work

This paper has described a stacked sub-word model for joint Chinese word segmentation and POS tagging. We defined a sub-word structure which maximizes the agreement of multiple segmentations provided by different segmenters. We showed that this sub-word structure could explore the complementary strength of different systems designed with different views. Moreover, the POS tagging could be efficiently and effectively resolved over sub-word sequences. To train a good sub-word tagger, we introduced a stacked learning procedure. Experiments showed that our approach was superior to the existing approaches reported in the literature.

Machine learning and statistical approaches encounter difficulties when the input/output data have a structured and relational form. Research in empirical Natural Language Processing has been tackling these complexities since the early work in the field. Recent work in machine learning has provided several paradigms to globally represent and process such data: linear models for structured prediction, graphical models, constrained conditional models, and reranking, among others. A general expressivity-efficiency trade off is observed. Although the stacked sub-word model is an ad hoc solution for a particular problem, namely joint word segmentation and POS tagging, the idea to employ system ensemble and stacked learning in general provides an alternative for structured problems. Multiple "cheap" coarse systems are used to provide diverse outputs, which may be inaccurate. These outputs are further merged into an intermediate representation, which allows an extractive system to use rich contexts to predict the final results. A natural avenue for future work is the extension of our method to other NLP tasks.

Acknowledgments

The work is supported by the project TAKE (Technologies for Advanced Knowledge Extraction), funded under contract 01IW08003 by the German Federal Ministry of Education and Research. The author is also funded by German Academic Exchange Service (DAAD).

The author would would like to thank Dr. Jia Xu for her helpful discussion, and Regine Bader for proofreading this paper.

References

Leo Breiman. 1996. Stacked regressions. *Mach. Learn.*, 24:49–64, July.

William W. Cohen and Vitor R. Carvalho. 2005. Stacked sequential learning. In *Proceedings of the 19th international joint conference on Artificial intelligence*, pages 671–676, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *JOURNAL OF MACHINE LEARNING RESEARCH*, 7:551–585.

Zhongqiang Huang, Mary Harper, and Wen Wang. 2007. Mandarin part-of-speech tagging and discriminative reranking. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1093–1102, Prague, Czech Republic, June. Association for Computational Linguistics.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008a. A cascaded linear model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL-08: HLT*, pages 897–904, Columbus, Ohio, June. Association for Computational Linguistics.

Wenbin Jiang, Haitao Mi, and Qun Liu. 2008b. Word lattice reranking for Chinese word segmentation and part-of-speech tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 385–392, Manchester, UK, August. Coling 2008 Organizing Committee.

Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa, and Hitoshi Isahara. 2009. An error-driven word-character hybrid model for joint Chinese word segmentation and pos tagging. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Process-*

ing of the AFNLP, pages 513–521, Suntec, Singapore, August. Association for Computational Linguistics.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *NAACL '01: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001*, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 277–284, Barcelona, Spain, July. Association for Computational Linguistics.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio, June. Association for Computational Linguistics.

L. A. Ramshaw and M. P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the 3rd ACL/SIGDAT Workshop on Very Large Corpora, Cambridge, Massachusetts, USA*, pages 82–94.

Weiwei Sun. 2010. Word-based and character-based word segmentation models: Comparison and combination. In *Coling 2010: Posters*, pages 1211–1219, Beijing, China, August. Coling 2010 Organizing Committee.

André Filipe Torres Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 157–166, Honolulu, Hawaii, October. Association for Computational Linguistics.

David H. Wolpert. 1992. Original contribution: Stacked generalization. *Neural Netw.*, 5:241–259, February.

Dekai Wu, Grace Ngai, and Marine Carpuat. 2003. A stacked, voted, stacked model for named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 200–203.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. In *International Journal of Computational Linguistics and Chinese Language Processing*.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 840–847, Prague, Czech Republic, June. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL-08: HLT*, pages 888–896, Columbus, Ohio, June. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2010. A fast decoder for joint word segmentation and POS-tagging using a single discriminative model. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 843–852, Cambridge, MA, October. Association for Computational Linguistics.

Ruiqiang Zhang, Genichiro Kikui, and Eiichiro Sumita. 2006. Subword-based tagging by conditional random fields for Chinese word segmentation. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 193–196, New York City, USA, June. Association for Computational Linguistics.

Language-independent Compound Splitting with Morphological Operations

Klaus Macherey[1] **Andrew M. Dai**[2] **David Talbot**[1] **Ashok C. Popat**[1] **Franz Och**[1]

[1]Google Inc.
1600 Amphitheatre Pkwy.
Mountain View, CA 94043, USA
{kmach,talbot,popat,och}@google.com

[2]University of Edinburgh
10 Crichton Street
Edinburgh, UK EH8 9AB
a.dai@ed.ac.uk

Abstract

Translating compounds is an important problem in machine translation. Since many compounds have not been observed during training, they pose a challenge for translation systems. Previous decompounding methods have often been restricted to a small set of languages as they cannot deal with more complex compound forming processes. We present a novel and unsupervised method to learn the compound parts and morphological operations needed to split compounds into their compound parts. The method uses a bilingual corpus to learn the morphological operations required to split a compound into its parts. Furthermore, monolingual corpora are used to learn and filter the set of compound part candidates. We evaluate our method within a machine translation task and show significant improvements for various languages to show the versatility of the approach.

1 Introduction

A compound is a lexeme that consists of more than one stem. Informally, a compound is a combination of two or more words that function as a single unit of meaning. Some compounds are written as space-separated words, which are called open compounds (e.g. *hard drive*), while others are written as single words, which are called closed compounds (e.g. *wallpaper*). In this paper, we shall focus only on closed compounds because open compounds do not require further splitting.

The objective of compound splitting is to split a compound into its corresponding sequence of constituents. If we look at how compounds are created from lexemes in the first place, we find that for some languages, compounds are formed by concatenating existing words, while in other languages compounding additionally involves certain morphological operations. These morphological operations can become very complex as we illustrate in the following case studies.

1.1 Case Studies

Below, we look at splitting compounds from 3 different languages. The examples introduce in part the notation used for the decision rule outlined in Section 3.1.

1.1.1 English Compound Splitting

The word *flowerpot* can appear as a closed or open compound in English texts. To automatically split the closed form we have to try out every split point and choose the split with minimal costs according to a cost function. Let's assume that we already know that *flowerpot* must be split into two parts. Then we have to position two split points that mark the end of each part (one is always reserved for the last character position). The *number* of split points is denoted by K (i.e. $K = 2$), while the *position* of split points is denoted by n_1 and n_2. Since *flowerpot* consists of 9 characters, we have 8 possibilities to position split point n_1 within the characters $c_1, \ldots, c_8$. The final split point corresponds with the last character, that is, $n_2 = 9$. Trying out all possible single splits results in the following candidates:

$$\begin{aligned}
&\text{flowerpot} \rightarrow \text{f} + \text{lowerpot} \\
&\text{flowerpot} \rightarrow \text{fl} + \text{owerpot} \\
&\qquad\vdots \\
&\text{flowerpot} \rightarrow \textbf{flower} + \textbf{pot} \\
&\qquad\vdots \\
&\text{flowerpot} \rightarrow \text{flowerpo} + \text{t}
\end{aligned}$$

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1395–1404,
Portland, Oregon, June 19-24, 2011.

If we associate each compound part candidate with a cost that reflects how frequent this part occurs in a large collection of English texts, we expect that the correct split *flower* + *pot* will have the lowest cost.

1.1.2 German Compound Splitting

The previous example covered a case where the compound is constructed by directly concatenating the compound parts. While this works well for English, other languages require additional morphological operations. To demonstrate, we look at the German compound *Verkehrszeichen* (traffic sign) which consists of the two nouns *Verkehr* (traffic) and *Zeichen* (sign). Let's assume that we want to split this word into 3 parts, that is, $K = 3$. Then, we get the following candidates.

$$\begin{aligned}
&\text{Verkehrszeichen} \rightarrow \text{V} + \text{e} + \text{rkehrszeichen}\\
&\text{Verkehrszeichen} \rightarrow \text{V} + \text{er} + \text{kehrszeichen}\\
&\qquad\vdots\\
&\text{Verkehrszeichen} \rightarrow \textbf{Verkehr} + \textbf{s} + \textbf{zeichen}\\
&\qquad\vdots\\
&\text{Verkehrszeichen} \rightarrow \text{Verkehrszeich} + \text{e} + \text{n}
\end{aligned}$$

Using the same procedure as described before, we can lookup the compound parts in a dictionary or determine their frequency from large text collections. This yields the optimal split points $n_1 = 7, n_2 = 8, n_3 = 15$. The interesting part here is the additional s morpheme, which is called a linking morpheme, because it combines the two compound parts to form the compound *Verkehrszeichen*. If we have a list of all possible linking morphemes, we can hypothesize them between two ordinary compound parts.

1.1.3 Greek Compound Splitting

The previous example required the insertion of a linking morpheme between two compound parts. We shall now look at a more complicated morphological operation. The Greek compound χαρτόκουτο (*cardboard box*) consists of the two parts χαρτί (*paper*) and κουτί (*box*). Here, the problem is that the parts χαρτό and κουτο are not valid words in Greek. To lookup the correct words, we must substitute the suffix of the compound part candidates with some other morphemes. If we allow the compound part candidates to be transformed by some morphological operation, we can lookup the transformed compound parts in a dictionary or determine their frequencies in some large collection of Greek texts. Let's assume that we need only one split point. Then this yields the following compound part candidates:

$$\begin{aligned}
&\text{χαρτόκουτο} \rightarrow \text{χ} + \text{αρτόκουτο}\\
&\text{χαρτόκουτο} \rightarrow \text{χ} + \text{αρτίκουτο} \quad g_2 : \text{ό} \,/\, \text{ί}\\
&\text{χαρτόκουτο} \rightarrow \text{χ} + \text{αρτόκουτί} \quad g_2 : \text{ο} \,/\, \text{ί}\\
&\qquad\vdots\\
&\text{χαρτόκουτο} \rightarrow \textbf{χαρτί} + \textbf{κουτί} \quad g_1 : \text{ό} \,/\, \text{ί}\,, \quad g_2 : \text{ο} \,/\, \text{ί}\\
&\qquad\vdots\\
&\text{χαρτόκουτο} \rightarrow \text{χαρτίκουτ} + \text{ο} \quad g_1 : \text{ό} \,/\, \text{ί}\\
&\text{χαρτόκουτο} \rightarrow \text{χαρτίκουτ} + \text{ί} \quad g_2 : \text{ο} \,/\, \text{ί}
\end{aligned}$$

Here, $g_k : s/t$ denotes the kth compound part which is obtained by replacing string s with string t in the original string, resulting in the transformed part g_k.

1.2 Problems and Objectives

Our goal is to design a language-independent compound splitter that is useful for machine translation. The previous examples addressed the importance of a cost function that favors valid compound parts versus invalid ones. In addition, the examples have shown that, depending on the language, the morphological operations can become very complex. For most Germanic languages like Danish, German, or Swedish, the list of possible linking morphemes is rather small and can be provided manually. However, in general, these lists can become very large, and language experts who could provide such lists might not be at our disposal. Because it seems infeasible to list the morphological operations explicitly, we want to find and extract those operations automatically in an unsupervised way and provide them as an additional knowledge source to the decompounding algorithm.

Another problem is how to evaluate the quality of the compound splitter. One way is to compile for every language a large collection of compounds together with their valid splits and to measure the proportion of correctly split compounds. Unfortunately, such lists do not exist for many languages.

While the training algorithm for our compound splitter shall be unsupervised, the evaluation data needs to be verified by human experts. Since we are interested in improving machine translation and to circumvent the problem of explicitly annotating compounds, we evaluate the compound splitter within a machine translation task. By decompounding training and test data of a machine translation system, we expect an increase in the number of matching phrase table entries, resulting in better translation quality measured in BLEU score (Papineni et al., 2002). If BLEU score is sensitive enough to measure the quality improvements obtained from decompounding, there is no need to generate a separate gold standard for compounds.

Finally, we do not want to split non-compounds and named entities because we expect them to be translated non-compositionally. For example, the German word *Deutschland* (Germany) could be split into two parts *Deutsch* (German) + *Land* (country). Although this is a valid split, named entities should be kept as single units. An example for a non-compound is the German participle *vereinbart* (agreed) which could be wrongly split into the parts *Verein* (club) + *Bart* (beard). To avoid overly eager splitting, we will compile a list of non-compounds in an unsupervised way that serves as an exception list for the compound splitter. To summarize, we aim to solve the following problems:

- Define a cost function that favors valid compound parts and rejects invalid ones.
- Learn morphological operations, which is important for languages that have complex compound forming processes.
- Apply compound splitting to machine translation to aid in translation of compounds that have not been seen in the bilingual training data.
- Avoid splitting non-compounds and named entities as this may result in wrong translations.

2 Related work

Previous work concerning decompounding can be divided into two categories: monolingual and bilingual approaches.

Brown (2002) describes a corpus-driven approach for splitting compounds in a German-English translation task derived from a medical domain. A large proportion of the tokens in both texts are cognates with a Latin or Greek etymological origin. While the English text keeps the cognates as separate tokens, they are combined into compounds in the German text. To split these compounds, the author compares both the German and the English cognates on a character level to find reasonable split points. The algorithm described by the author consists of a sequence of *if-then-else* conditions that are applied on the two cognates to find the split points. Furthermore, since the method relies on finding similar character sequences between both the source and the target tokens, the approach is restricted to cognates and cannot be applied to split more complex compounds.

Koehn and Knight (2003) present a frequency-based approach to compound splitting for German. The compound parts and their frequencies are estimated from a monolingual corpus. As an extension to the frequency approach, the authors describe a bilingual approach where they use a dictionary extracted from parallel data to find better split options. The authors allow only two linking morphemes between compound parts and a few letters that can be dropped. In contrast to our approach, those operations are not learned automatically, but must be provided explicitly.

Garera and Yarowsky (2008) propose an approach to translate compounds without the need for bilingual training texts. The compound splitting procedure mainly follows the approach from (Brown, 2002) and (Koehn and Knight, 2003), so the emphasis is put on finding correct translations for compounds. To accomplish this, the authors use cross-language compound evidence obtained from bilingual dictionaries. In addition, the authors describe a simple way to learn glue characters by allowing the deletion of up to two middle and two end characters.[1] More complex morphological operations are not taken into account.

Alfonseca et al. (2008b) describe a state-of-the-art German compound splitter that is particularly robust with respect to noise and spelling errors. The compound splitter is trained on monolingual data. Besides applying frequency and probability-based methods, the authors also take the mutual information of compound parts into account. In addition, the

[1] However, the glue characters found by this procedure seem to be biased for at least German and Albanian. A very frequent glue morpheme like *-es-* is not listed, while glue morphemes like *-k-* and *-h-* rank very high, although they are invalid glue morphemes for German. Albanian shows similar problems.

authors look for compound parts that occur in different anchor texts pointing to the same document. All these signals are combined and the weights are trained using a support vector machine classifier. Alfonseca et al. (2008a) apply this compound splitter on various other Germanic languages.

Dyer (2009) applies a maximum entropy model of compound splitting to generate segmentation lattices that serve as input to a translation system. To train the model, reference segmentations are required. Here, we produce only single best segmentations, but otherwise do not rely on reference segmentations.

3 Compound Splitting Algorithm

In this section, we describe the underlying optimization problem and the algorithm used to split a token into its compound parts. Starting from Bayes' decision rule, we develop the Bellman equation and formulate a dynamic programming-based algorithm that takes a word as input and outputs the constituent compound parts. We discuss the procedure used to extract compound parts from monolingual texts and to learn the morphological operations using bilingual corpora.

3.1 Decision Rule for Compound Splitting

Given a token $w = c_1, \ldots, c_N = c_1^N$ consisting of a sequence of N characters c_i, the objective function is to find the optimal number $\hat{K}$ and sequence of split points $\hat{n}_0^{\hat{K}}$ such that the subwords are the constituents of the token, where[2] $n_0 := 0$ and $n_K := N$:

$$
\begin{aligned}
w = c_1^N \rightarrow (\hat{K}, \hat{n}_0^{\hat{K}}) &= \\
&= \arg\max_{K, n_0^K} \left\{ \Pr(c_1^N, K, n_0^K) \right\} \qquad (1) \\
&= \arg\max_{K, n_0^K} \left\{ \Pr(K) \cdot \Pr(c_1^N, n_0^K | K) \right\} \\
&\approx \arg\max_{K, n_0^K} \Big\{ p(K) \cdot \prod_{k=1}^{K} p(c_{n_{k-1}+1}^{n_k}, n_{k-1} | K) \cdot \\
&\qquad \cdot p(n_k | n_{k-1}, K) \Big\} \qquad (2)
\end{aligned}
$$

with $p(n_0) = p(n_K|\cdot) \equiv 1$. Equation 2 requires that token w can be fully decomposed into a sequence of lexemes, the compound parts. Thus, determining the optimal segmentation is sufficient for finding the constituents. While this may work for some languages, the subwords are not valid words in general as discussed in Section 1.1.3. Therefore, we allow the lexemes to be the result of a transformation process, where the transformed lexemes are denoted by g_1^K. This leads to the following refined decision rule:

$$
\begin{aligned}
w = c_1^N \rightarrow (\hat{K}, \hat{n}_0^{\hat{K}}, \hat{g}_1^{\hat{K}}) &= \\
&= \arg\max_{K, n_0^K, g_1^K} \left\{ \Pr(c_1^N, K, n_0^K, g_1^K) \right\} \qquad (3) \\
&= \arg\max_{K, n_0^K, g_1^K} \left\{ \Pr(K) \cdot \Pr(c_1^N, n_0^K, g_1^K | K) \right\} \qquad (4) \\
&\approx \arg\max_{K, n_0^K, g_1^K} \Big\{ p(K) \cdot \prod_{k=1}^{K} \underbrace{p(c_{n_{k-1}+1}^{n_k}, n_{k-1}, g_k | K)}_{\text{compound part probability}} \cdot \\
&\qquad \cdot p(n_k | n_{k-1}, K) \Big\} \qquad (5)
\end{aligned}
$$

The compound part probability is a zero-order model. If we penalize each split with a constant split penalty ξ, and make the probability independent of the number of splits K, we arrive at the following decision rule:

$$
\begin{aligned}
w = c_1^N \rightarrow (\hat{K}, \hat{n}_1^{\hat{K}}, \hat{g}_1^{\hat{K}}) &\\
= \arg\max_{K, n_0^K, g_1^K} \Big\{ \xi^K \cdot \prod_{k=1}^{K} p(c_{n_{k-1}+1}^{n_k}, n_{k-1}, g_k) \Big\} \qquad (6)
\end{aligned}
$$

3.2 Dynamic Programming

We use dynamic programming to find the optimal split sequence. Each split infers certain costs that are determined by a cost function. The total costs of a decomposed word can be computed from the individual costs of the component parts. For the dynamic programming approach, we define the following auxiliary function $\mathcal{Q}$ with $n_k = j$:

$$
\mathcal{Q}(c_1^j) = \max_{n_0^k, g_1^k} \Big\{ \xi^k \cdot \prod_{\kappa=1}^{k} p(c_{n_{\kappa-1}+1}^{n_\kappa}, n_{\kappa-1}, g_\kappa) \Big\}
$$

that is, $\mathcal{Q}(c_1^j)$ is equal to the minimal costs (maximum probability) that we assign to the prefix string c_1^j where we have used k split points at positions n_1^k. This yields the following recursive equation:

$$
\begin{aligned}
\mathcal{Q}(c_1^j) = \max_{n_k, g_k} \Big\{ & \xi \cdot \mathcal{Q}(c_1^{n_{k-1}}) \cdot \\
& \cdot p(c_{n_{k-1}+1}^{n_k}, n_{k-1}, g_k) \Big\} \qquad (7)
\end{aligned}
$$

[2] For algorithmic reasons, we use the start position 0 to represent a fictitious start symbol before the first character of the word.

Algorithm 1 Compound splitting

Input: input word $w = c_1^N$
Output: compound parts
  $\mathcal{Q}(0) = 0$
  $\mathcal{Q}(1) = \cdots = \mathcal{Q}(N) = \infty$
  for $i = 0, \ldots, N-1$ **do**
    for $j = i+1, \ldots, N$ **do**
      split-costs $= \mathcal{Q}(i) + \text{cost}(c_{i+1}^j, i, g_j) +$ split-penalty
      if split-costs $< \mathcal{Q}(j)$ **then**
        $\mathcal{Q}(j) =$ split-costs
        $\mathcal{B}(j) = (i, g_j)$
      end if
    end for
  end for

with backpointer

$$\mathcal{B}(j) = \arg\max_{n_k, g_k} \left\{ \xi \cdot \mathcal{Q}(c_1^{n_{k-1}}) \cdot \cdot p(c_{n_{k-1}+1}^{n_k}, n_{k-1}, g_k) \right\} \quad (8)$$

Using logarithms in Equations 7 and 8, we can interpret the quantities as additive costs rather than probabilities. This yields Algorithm 1, which is quadratic in the length of the input string. By enforcing that each compound part does not exceed a predefined constant length ℓ, we can change the second **for** loop as follows:

for $j = i+1, \ldots, \min(i+\ell, N)$ **do**

With this change, Algorithm 1 becomes linear in the length of the input word, $O(|w|)$.

4 Cost Function and Knowledge Sources

The performance of Algorithm 1 depends on the cost function cost$(\cdot)$, that is, the probability $p(c_{n_{k-1}+1}^{n_k}, n_{k-1}, g_k)$. This cost function incorporates knowledge about morpheme transformations, morpheme positions within a compound part, and the compound parts themselves.

4.1 Learning Morphological Operations using Phrase Tables

Let s and t be strings of the (source) language alphabet $\mathcal{A}$. A morphological operation s/t is a pair of strings $s, t \in \mathcal{A}^*$, where s is replaced by t. With the usual definition of the Kleene operator $*$, s and t can be empty, denoted by ε. An example for such a pair is ε/es, which models the linking morpheme *es* in the German compound *Bundesagentur* (federal agency):

$$\text{Bundesagentur} \rightarrow \text{Bund} + \text{es} + \text{Agentur} .$$

Note that by replacing either s or t with ε, we can model insertions or deletions of morphemes. The explicit dependence on position n_{k-1} in Equation 6 allows us to determine if we are at the beginning, in the middle, or at the end of a token. Thus, we can distinguish between start, middle, or end morphemes and hypothesize them during search.[3] Although not explicitly listed in Algorithm 1, we disallow sequences of linking morphemes. This can be achieved by setting the costs to infinity for those morpheme hypotheses, which directly succeed another morpheme hypothesis.

To learn the morphological operations involved in compounding, we determine the differences between a compound and its compound parts. This can be done by computing the Levenshtein distance between the compound and its compound parts, with the allowable edit operations being insertion, deletion, or substitution of one or more characters. If we store the current and previous characters, edit operation and the location (prefix, infix or suffix) at each position during calculation of the Levenshtein distance then we can obtain the morphological operations required for compounding. Applying the inverse operations, that is, replacing t with s yields the operation required for decompounding.

4.1.1 Finding Compounds and their Parts

To learn the morphological operations, we need compounds together with their compound parts. The basic idea of finding compound candidates and their compound parts in a bilingual setting are related to the ideas presented in (Garera and Yarowsky, 2008). Here, we use phrase tables rather than dictionaries. Although phrase tables might contain more noise, we believe that overall phrase tables cover more phenomena of translations than what can be found in dictionaries. The procedure is as follows. We are given a phrase table that provides translations for phrases from a source language l into English and from English into l. Under the assumption that English does not contain many closed compounds, we can search

[3] We jointly optimize over K and the split points n_k, so we know that $c_{n_{K-1}}^{n_K}$ is a suffix of w.

the phrase table for those single-token source words f in language l, which translate into multi-token English phrases $e_1, \ldots, e_n$ for $n > 1$. This results in a list of $(f; e_1, \ldots, e_n)$ pairs, which are potential compound candidates together with their English translations. If for each pair, we take each token e_i from the English (multi-token) phrase and lookup the corresponding translation for language l to get g_i, we should find entries that have at least some partial match with the original source word f, if f is a true compound. Because the translation phrase table was generated automatically during the training of a multi-language translation system, there is no guarantee that the original translations are correct. Thus, the bilingual extraction procedure is subject to introduce a certain amount of noise. To mitigate this, thresholds such as minimum edit distance between the potential compound and its parts, minimum co-occurrence frequencies for the selected bilingual phrase pairs and minimum source and target word lengths are used to reduce the noise at the expense of finding fewer compounds. Those entries that obey these constraints are output as triples of form:

$$(f; e_1, \ldots, e_n; g_1, \ldots, g_n) \tag{9}$$

where

- f is likely to be a compound,
- $e_1, \ldots, e_n$ is the English translation, and
- $g_1, \ldots, g_n$ are the compound parts of f.

The following example for German illustrates the process. Suppose that the most probable translation for *Überweisungsbetrag* is *transfer amount* using the phrase table. We then look up the translation back to German for each translated token: *transfer* translates to *Überweisung* and *amount* translates to *Betrag*. We then calculate the distance between all permutations of the parts and the original compound and choose the one with the lowest distance and highest translation probability: *Überweisung Betrag*.

4.2 Monolingual Extraction of Compound Parts

The most important knowledge source required for Algorithm 1 is a word-frequency list of compound parts that is used to compute the split costs. The procedure described in Section 4.1.1 is useful for learning morphological operations, but it is not sufficient to extract an exhaustive list of compound parts. Such lists can be extracted from monolingual data for which we use language model (LM) word frequency lists in combination with some filter steps. The extraction process is subdivided into 2 passes, one over a high-quality news LM to extract the parts and the other over a web LM to filter the parts.

4.2.1 Phase 1: Bootstrapping pass

In the first pass, we generate word frequency lists derived from news articles for multiple languages. The motivation for using news articles rather than arbitrary web texts is that news articles are in general less noisy and contain fewer spelling mistakes. The language-dependent word frequency lists are filtered according to a sequence of filter steps. These filter steps include discarding all words that contain digits or punctuations other than hyphen, minimum occurrence frequency, and a minimum length which we set to 4. The output is a table that contains preliminary compound parts together with their respective counts for each language.

4.2.2 Phase 2: Filtering pass

In the second pass, the compound part vocabulary is further reduced and filtered. We generate a LM vocabulary based on arbitrary web texts for each language and build a compound splitter based on the vocabulary list that was generated in phase 1. We now try to split every word of the web LM vocabulary based on the compound splitter model from phase 1. For the compound parts that occur in the compound splitter output, we determine how often each compound part was used and output only those compound parts whose frequency exceed a predefined threshold n.

4.3 Example

Suppose we have the following word frequencies output from pass 1:

floor	10k	poll	4k
flow	9k	pot	5k
flower	15k	potter	20k

In pass 2, we observe the word *flowerpot*. With the above list, the only compound parts used are *flower* and *pot*. If we did not split any other words and threshold at $n = 1$, our final list would consist of *flower* and *pot*. This filtering pass has the advantage of outputting only those compound part candidates

which were actually used to split words from web texts. The thresholding also further reduces the risk of introducing noise. Another advantage is that since the set of parts output in the first pass may contain a high number of compounds, the filter is able to remove a large number of these compounds by examining relative frequencies. In our experiments, we have assumed that compound part frequencies are higher than the compound frequency and so remove words from the part list that can themselves be split and have a relatively high frequency. Finally, after removing the low frequency compound parts, we obtain the final compound splitter vocabulary.

4.4 Generating Exception Lists

To avoid eager splitting of non-compounds and named entities, we use a variant of the procedure described in Section 4.1.1. By emitting all those source words that translate with high probability into single-token English words, we obtain a list of words that should not be split.[4]

4.5 Final Cost Function

The final cost function is defined by the following components which are combined log-linearly.

- The split penalty ξ penalizes each compound part to avoid eager splitting.
- The cost for each compound part g_k is computed as $-\log C(g_k)$, where $C(g_k)$ is the unigram count for g_k obtained from the news LM word frequency list. Since we use a zero-order model, we can ignore the normalization and work with unigram counts rather than unigram probabilities.
- Because Algorithm 1 iterates over the characters of the input token w, we can infer from the boundaries (i, j) if we are at the start, in the middle, or at the end of the token. Applying a morphological operation adds costs 1 to the overall costs.

Although the cost function is language dependent, we use the same split penalty weight $\xi = 20$ for all languages except for German, where the split penalty weight is set to 13.5.

5 Results

To show the language independence of the approach within a machine translation task, we translate from languages belonging to different language families into English. The publicly available Europarl corpus is not suitable for demonstrating the utility of compound splitting because there are few unseen compounds in the test section of the Europarl corpus. The WMT shared translation task has a broader domain compared to Europarl but covers only a few languages. Hence, we present results for German-English using the WMT-07 data and cover other languages using non-public corpora which contain news as well as open-domain web texts. Table 1 lists the various corpus statistics. The source languages are grouped according to their language family.

For learning the morphological operations, we allowed the substitution of at most 2 consecutive characters. Furthermore, we only allowed at most one morphological substitution to avoid introducing too much noise. The found morphological operations were sorted according to their frequencies. Those which occurred less than 100 times were discarded. Examples of extracted morphological operations are given in Table 2. Because the extraction procedure described in Section 4.1 is not purely restricted to the case of decompounding, we found that many morphological operations emitted by this procedure reflect morphological variations that are not directly linked to compounding, but caused by inflections.

To generate the language-dependent lists of compound parts, we used language model vocabulary lists[5] generated from news texts for different languages as seeds for the first pass. These lists were filtered by discarding all entries that either contained digits, punctuations other than hyphens, or sequences of the same characters. In addition, the infrequent entries were discarded as well to further reduce noise. For the second pass, we used the lists generated in the first pass together with the learned morphological operations to construct a preliminary compound splitter. We then generated vocabulary lists for monolingual web texts and applied the preliminary compound splitter onto this list. The used

[4] Because we will translate only into English, this is not an issue for the introductory example *flowerpot*.

[5] The vocabulary lists also contain the word frequencies. We use the term vocabulary list synonymously for a word frequency list.

Family	Src Language	#Tokens Train src/trg		#Tokens Dev src/trg		#Tokens Tst src/trg	
Germanic	Danish	196M	201M	43,475	44,479	72,275	74,504
	German	43M	45M	23,151	22,646	45,077	43,777
	Norwegian	251M	255M	42,096	43,824	70,257	73,556
	Swedish	201M	213M	42,365	44,559	70,666	74,547
Hellenic	Greek	153M	148M	47,576	44,658	79,501	74,776
Uralic	Estonian	199M	244M	34,987	44,658	57,916	74,765
	Finnish	205M	246M	32,119	44,658	53,365	74,771

Table 1: Corpus statistics for various language pairs. The target language is always English. The source languages are grouped according to their language family.

Language	morpholog. operations
Danish	-/ε, s/ε
German	-/ε, s/ε, es/ε, n/ε, e/ε, en/ε
Norwegian	-/ε, s/ε, e/ε
Swedish	-/ε, s/ε
Greek	ε/α, ε/ς, ε/η, o/ί, o/ί, o/ν
Estonian	-/ε, e/ε, se/ε
Finnish	ε/n, n/ε, ε/en

Table 2: Examples of morphological operations that were extracted from bilingual corpora.

compound parts were collected and sorted according to their frequencies. Those which were used at least 2 times were kept in the final compound parts lists. Table 3 reports the number of compound parts kept after each pass. For example, the Finnish news vocabulary list initially contained 1.7M entries. After removing non-alpha and infrequent words in the first filter step, we obtained 190K entries. Using the preliminary compound splitter in the second filter step resulted in 73K compound part entries.

The finally obtained compound splitter was integrated into the preprocessing pipeline of a state-of-the-art statistical phrase-based machine translation system that works similar to the Moses decoder (Koehn et al., 2007). By applying the compound splitter during both training and decoding we ensured that source language tokens were split in the same way. Table 4 presents results for various language-pairs with and without decompounding. Both the Germanic and the Uralic languages show significant BLEU score improvements of 1.3 BLEU points on average. The confidence intervals were computed using the bootstrap resampling normal approximation method described in (Noreen, 1989). While the compounding process for Germanic languages is rather simple and requires only a few linking morphemes, compounds used in Uralic languages have a richer morphology. In contrast to the Germanic and Uralic languages, we did not observe improvements for Greek. To investigate this lack of performance, we turned off transliteration and kept unknown source words in their original script. We analyzed the number of remaining source characters in the baseline system and the system using compound splitting by counting the number of Greek characters in the translation output. The number of remaining Greek characters in the translation output was reduced from $6,715$ in the baseline system to $3,624$ in the system which used decompounding. In addition, a few other metrics like the number of source words that consisted of more than 15 characters decreased as well. Because we do not know how many compounds are actually contained in the Greek source sentences[6] and because the frequency of using compounds might vary across languages, we cannot expect the same performance gains across languages belonging to different language families. An interesting observation is, however, that if one language from a language family shows performance gains, then there are performance gains for all the languages in that family. We also investigated the effect of not using any morphological operations. Disallowing all morphological operations accounts for a loss of 0.1 - 0.2 BLEU points across translation systems and increases the compound parts vocabulary lists by up to 20%, which means that most of the gains can be achieved with simple concatenation.

The exception lists were generated according to the procedure described in Section 4.4. Since we aimed for precision rather than recall when constructing these lists, we inserted only those source

[6]Quite a few of the remaining Greek characters belong to rare named entities.

Language	initial vocab size	#parts after 1st pass	#parts after 2nd pass
Danish	918,708	132,247	49,592
German	7,908,927	247,606	45,059
Norwegian	1,417,129	237,099	62,107
Swedish	1,907,632	284,660	82,120
Greek	877,313	136,436	33,130
Estonian	742,185	81,132	36,629
Finnish	1,704,415	190,507	73,568

Table 3: Number of remaining compound parts for various languages after the first and second filter step.

System	BLEU[%] w/o splitting	BLEU[%] w splitting	Δ	CI 95%
Danish	42.55	44.39	1.84	($\pm$ 0.65)
German WMT-07	25.76	26.60	0.84	($\pm$ 0.70)
Norwegian	42.77	44.58	1.81	($\pm$ 0.64)
Swedish	36.28	38.04	1.76	($\pm$ 0.62)
Greek	31.85	31.91	0.06	($\pm$ 0.61)
Estonian	20.52	21.20	0.68	($\pm$ 0.50)
Finnish	25.24	26.64	1.40	($\pm$ 0.57)

Table 4: BLEU score results for various languages translated into English with and without compound splitting.

Language	Split	source	translation
German	no	Die EU ist nicht einfach ein Freundschaftsclub.	The EU is not just a Freundschaftsclub.
	yes	Die EU ist nicht einfach ein Freundschaft Club	The EU is not simply a friendship club.
Greek	no	Τι είναι παλμοκωδική διαμόρφωση;	What παλμοκωδική configuration?
	yes	Τι είναι παλμο κωδικη διαμόρφωση;	What is pulse code modulation?
Finnish	no	Lisävuodevaatteet ja pyyheliinat ovat kaapissa.	Lisävuodevaatteet and towels are in the closet.
	yes	Lisä Vuode Vaatteet ja pyyheliinat ovat kaapissa.	Extra bed linen and towels are in the closet.

Table 5: Examples of translations into English with and without compound splitting.

words whose co-occurrence count with a unigram translation was at least $1,000$ and whose translation probability was larger than 0.1. Furthermore, we required that at least 70% of all target phrase entries for a given source word had to be unigrams. All decompounding results reported in Table 4 were generated using these exception lists, which prevented wrong splits caused by otherwise overly eager splitting.

6 Conclusion and Outlook

We have presented a language-independent method for decompounding that improves translations for compounds that otherwise rarely occur in the bilingual training data. We learned a set of morphological operations from a translation phrase table and determined suitable compound part candidates from monolingual data in a two pass process. This allowed us to learn morphemes and operations for languages where these lists are not available. In addition, we have used the bilingual information stored in the phrase table to avoid splitting non-compounds as well as frequent named entities. All knowledge sources were combined in a cost function that was applied in a compound splitter based on dynamic programming. Finally, we have shown this improves translation performance on languages from different language families.

The weights were not optimized in a systematic way but set manually to their respective values. In the future, the weights of the cost function should be learned automatically by optimizing an appropriate error function. Instead of using gold data, the development data for optimizing the error function could be collected without supervision using the methods proposed in this paper.

References

Enrique Alfonseca, Slaven Bilac, and Stefan Paries. 2008a. Decompounding query keywords from compounding languages. In *Proc. of the 46th Annual Meeting of the Association for Computational Linguistics (ACL): Human Language Technologies (HLT)*, pages 253--256, Columbus, Ohio, USA, June.

Enrique Alfonseca, Slaven Bilac, and Stefan Paries. 2008b. German decompounding in a difficult corpus. In A. Gelbukh, editor, *Lecture Notes in Computer Science (LNCS): Proc. of the 9th Int. Conf. on Intelligent Text Processing and Computational Linguistics (CICLING)*, volume 4919, pages 128--139. Springer Verlag, February.

Ralf D. Brown. 2002. Corpus-Driven Splitting of Compound Words. In *Proc. of the 9th Int. Conf. on Theoretical and Methodological Issues in Machine Translation (TMI)*, pages 12--21, Keihanna, Japan, March.

Chris Dyer. 2009. Using a maximum entropy model to build segmentation lattices for mt. In *Proc. of the Human Language Technologies (HLT): The Annual Conf. of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 406--414, Boulder, Colorado, June.

Nikesh Garera and David Yarowsky. 2008. Translating Compounds by Learning Component Gloss Translation Models via Multiple Languages. In *Proc. of the 3rd Internation Conference on Natural Language Processing (IJCNLP)*, pages 403--410, Hyderabad, India, January.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proc. of the 10th Conf. of the European Chapter of the Association for Computational Linguistics (EACL)*, volume 1, pages 187--193, Budapest, Hungary, April.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 177--180, Prague, Czech Republic, June.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses*. John Wiley & Sons, Canada.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311--318, Philadelphia, Pennsylvania, July.

Parsing the Internal Structure of Words: A New Paradigm for Chinese Word Segmentation

Zhongguo Li
State Key Laboratory on Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology
Tsinghua University, Beijing 100084, China
eemath@gmail.com

Abstract

Lots of Chinese characters are very productive in that they can form many structured words either as prefixes or as suffixes. Previous research in Chinese word segmentation mainly focused on identifying only the word boundaries without considering the rich internal structures of many words. In this paper we argue that this is unsatisfying in many ways, both practically and theoretically. Instead, we propose that word structures should be recovered in morphological analysis. An elegant approach for doing this is given and the result is shown to be promising enough for encouraging further effort in this direction. Our probability model is trained with the Penn Chinese Treebank and actually is able to parse both word and phrase structures in a unified way.

1 Why Parse Word Structures?

Research in Chinese word segmentation has progressed tremendously in recent years, with state of the art performing at around 97% in precision and recall (Xue, 2003; Gao et al., 2005; Zhang and Clark, 2007; Li and Sun, 2009). However, virtually all these systems focus exclusively on recognizing the word boundaries, giving no consideration to the internal structures of many words. Though it has been the standard practice for many years, we argue that this paradigm is inadequate both in theory and in practice, for at least the following four reasons.

The first reason is that if we confine our definition of word segmentation to the identification of word boundaries, then people tend to have divergent opinions as to whether a linguistic unit is a word or not (Sproat et al., 1996). This has led to many different annotation standards for Chinese word segmentation. Even worse, this could cause inconsistency in the same corpus. For instance, 副主席 'vice president' is considered to be one word in the Penn Chinese Treebank (Xue et al., 2005), but is split into two words by the Peking University corpus in the SIGHAN Bakeoffs (Sproat and Emerson, 2003). Meanwhile, 副导演 'vice director' and 副经理 'deputy manager' are both segmented into two words in the same Penn Chinese Treebank. In fact, all these words are composed of the prefix 副 'vice' and a root word. Thus the structure of 副主席 'vice president' can be represented with the tree in Figure 1. Without a doubt, there is complete agreement on the correctness of this structure among native Chinese speakers. So if instead of annotating only word boundaries, we annotate the structures of every word, [1] then the annotation tends to be more

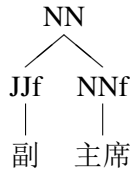


Figure 1: Example of a word with internal structure.

[1] Here it is necessary to add a note on terminology used in this paper. Since there is no universally accepted definition of the "word" concept in linguistics and especially in Chinese, whenever we use the term "word" we might mean a linguistic unit such as 副主席 'vice president' whose structure is shown as the tree in Figure 1, or we might mean a smaller unit such as 主席 'president' which is a substructure of that tree. Hopefully,

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1405–1414,
Portland, Oregon, June 19-24, 2011.

consistent and there could be less duplication of efforts in developing the expensive annotated corpus.

The second reason is applications have different requirements for granularity of words. Take the personal name 周树人 'Zhou Shuren' as an example. It's considered to be one word in the Penn Chinese Treebank, but is segmented into a surname and a given name in the Peking University corpus. For some applications such as information extraction, the former segmentation is adequate, while for others like machine translation, the later finer-grained output is more preferable. If the analyzer can produce a structure as shown in Figure 4(a), then every application can extract what it needs from this tree. A solution with tree output like this is more elegant than approaches which try to meet the needs of different applications in post-processing (Gao et al., 2004).

The third reason is that traditional word segmentation has problems in handling many phenomena in Chinese. For example, the telescopic compound 大中小学 'universities, middle schools and primary schools' is in fact composed of three coordinating elements 大学 'university', 中学 'middle school' and 小学 'primary school'. Regarding it as one flat word loses this important information. Another example is separable words like 游泳 'swim'. With a linear segmentation, the meaning of 'swimming' as in 游完泳 'after swimming' cannot be properly represented, since 游泳 'swim' will be segmented into discontinuous units. These language usages lie at the boundary between syntax and morphology, and are not uncommon in Chinese. They can be adequately represented with trees (Figure 2).

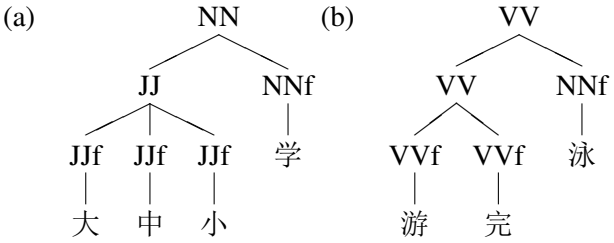


Figure 2: Example of telescopic compound (a) and separable word (b).

The last reason why we should care about word structures is related to head driven statistical parsers (Collins, 2003). To illustrate this, note that in the Penn Chinese Treebank, the word 英格兰人 'English People' does not occur at all. Hence constituents headed by such words could cause some difficulty for head driven models in which out-of-vocabulary words need to be treated specially both when they are generated and when they are conditioned upon. But this word is in turn headed by its suffix 人 'people', and there are 2,233 such words in Penn Chinese Treebank. If we annotate the structure of every compound containing this suffix (e.g. Figure 3), such data sparsity simply goes away.

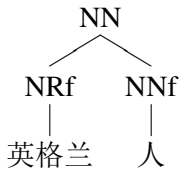


Figure 3: Structure of the out-of-vocabulary word 英格兰人 'English People'.

Had there been only a few words with internal structures, current Chinese word segmentation paradigm would be sufficient. We could simply recover word structures in post-processing. But this is far from the truth. In Chinese there is a large number of such words. We just name a few classes of these words and give one example for each class (a dot is used to separate roots from affixes):

personal name: 长尾·真 'Nagao Makoto'
location name: 纽约·州 'New York State'
noun with a suffix: 分类·器 'classifier'
noun with a prefix: 准·妈妈 'mother-to-be'
verb with a suffix: 自动·化 'automatize'
verb with a prefix: 防·水 'waterproof'
adjective with a suffix: 复合·型 'composite'
adjective with a prefix: 非·正式 'informal'
pronoun with a prefix: 各·位 'everybody'
time expression: 一九九五·年 'the year 1995'
ordinal number: 第·十一 'eleventh'
retroflex suffixation: 花朵·儿 'flower'

This list is not meant to be complete, but we can get a feel of how extensive the words with non-trivial structures can be. With so many productive suffixes and prefixes, analyzing word structures in post-processing is difficult, because a character may or may not act as an affix depending on the context.

the context will always make it clear what is being referred to with the term "word".

For example, the character 人 'people' in 植树人 'the one who plants' is a suffix, but in the personal name 周树人 'Zhou Shuren' it isn't. The structures of these two words are shown in Figure 4.

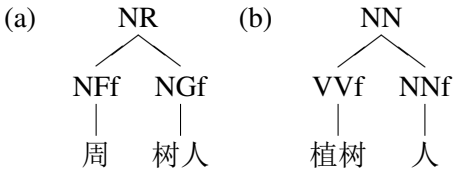


Figure 4: Two words that differ only in one character, but have different internal structures. The character 人 'people' is part of a personal name in tree (a), but is a suffix in (b).

A second reason why generally we cannot recover word structures in post-processing is that some words have very complex structures. For example, the tree of 无政府主义者 'anarchist' is shown in Figure 5. Parsing this structure correctly without a principled method is difficult and messy, if not impossible.

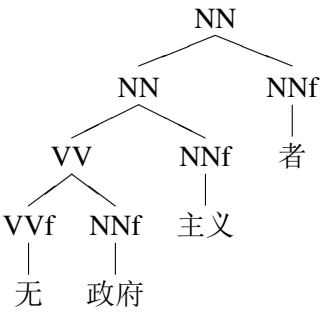


Figure 5: An example word which has very complex structures.

Finally, it must be mentioned that we cannot store all word structures in a dictionary, as the word formation process is very dynamic and productive in nature. Take 馆 'hall' as an example. Standard Chinese dictionaries usually contain 图书馆 'library', but not many other words such as 海洋馆 'aquarium' generated by this same character. This is understandable since the character 馆 'hall' is so productive that it is impossible for a dictionary to list every word with this character as a suffix. The same thing happens for natural language processing systems. Thus it is necessary to have a dynamic mechanism for parsing word structures.

In this paper, we propose a new paradigm for Chinese word segmentation in which not only word boundaries are identified but the internal structures of words are recovered (Section 3). To achieve this, we design a joint morphological and syntactic parsing model of Chinese (Section 4). Our generative story describes the complete process from sentence and word structures to the surface string of characters in a top-down fashion. With this probability model, we give an algorithm to find the parse tree of a raw sentence with the highest probability (Section 5). The output of our parser incorporates word structures naturally. Evaluation shows that the model can learn much of the regularity of word structures, and also achieves reasonable accuracy in parsing higher level constituent structures (Section 6).

2 Related Work

The necessity of parsing word structures has been noticed by Zhao (2009), who presented a character-level dependency scheme as an alternative to the linear representation of words. Although our work is based on the same notion, there are two key differences. The first one is that part-of-speech tags and constituent labels are fundamental for our parsing model, while Zhao focused on unlabeled dependencies between characters in a word, and part-of-speech information was not utilized. Secondly, we distinguish explicitly the generation of flat words such as 华盛顿 'Washington' and words with internal structures. Our parsing algorithm also has to be adapted accordingly. Such distinction was not made in Zhao's parsing model and algorithm.

Many researchers have also noticed the awkwardness and insufficiency of current boundary-only Chinese word segmentation paradigm, so they tried to customize the output to meet the requirements of various applications (Wu, 2003; Gao et al., 2004). In a related research, Jiang et al. (2009) presented a strategy to transfer annotated corpora between different segmentation standards in the hope of saving some expensive human labor. We believe the best solution to the problem of divergent standards and requirements is to annotate and analyze word structures. Then applications can make use of these structures according to their own convenience.

Since the distinction between morphology and syntax in Chinese is somewhat blurred, our model for word structure parsing is integrated with constituent parsing. There has been many efforts to integrate Chinese word segmentation, part-of-speech tagging and parsing (Wu and Zixin, 1998; Zhou and Su, 2003; Luo, 2003; Fung et al., 2004). However, in these research all words were considered to be flat, and thus word structures were not parsed. This is a crucial difference with our work. Specifically, consider the word 橄榄油 'olive oil'. Our parser output tree Figure 6(a), while Luo (2003) output tree (b), giving no hint to the structure of this word since the result is the same with a real flat word 洛杉矶 'Los Angeles'(c).

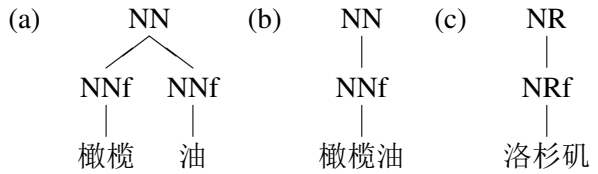


Figure 6: Difference between our output (a) of parsing the word 橄榄油 'olive oil' and the output (b) of Luo (2003). In (c) we have a true flat word, namely the location name 洛杉矶 'Los Angeles'.

The benefits of joint modeling has been noticed by many. For example, Li et al. (2010) reported that a joint syntactic and semantic model improved the accuracy of both tasks, while Ng and Low (2004) showed it's beneficial to integrate word segmentation and part-of-speech tagging into one model. The later result is confirmed by many others (Zhang and Clark, 2008; Jiang et al., 2008; Kruengkrai et al., 2009). Goldberg and Tsarfaty (2008) showed that a single model for morphological segmentation and syntactic parsing of Hebrew yielded an error reduction of 12% over the best pipelined models. This is because an integrated approach can effectively take into account more information from different levels of analysis.

Parsing of Chinese word structures can be reduced to the usual constituent parsing, for which there has been great progress in the past several years. Our generative model for unified word and phrase structure parsing is a direct adaptation of the model presented by Collins (2003). Many other approaches of constituent parsing also use this kind of head-driven generative models (Charniak, 1997; Bikel and Chiang, 2000) .

3 The New Paradigm

Given a raw Chinese sentence like 林志浩是总工程师, a traditional word segmentation system would output some result like 林志浩 是 总工程师('Lin Zhihao', 'is', 'chief engineer'). In our new paradigm, the output should at least be a linear sequence of trees representing the structures of each word as in Figure 7.

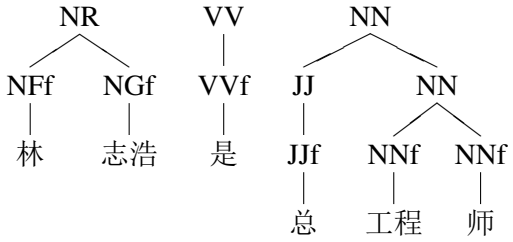


Figure 7: Proposed output for the new Chinese word segmentation paradigm.

Note that in the proposed output, all words are annotated with their part-of-speech tags. This is necessary since part-of-speech plays an important role in the generation of compound words. For example, 者 'person' usually combines with a verb to form a compound noun such as 设计者 'designer'.

In this paper, we will actually design an integrated morphological and syntactical parser trained with a treebank. Therefore, the real output of our system looks like Figure 8. It's clear that besides all the information of the proposed output for the new

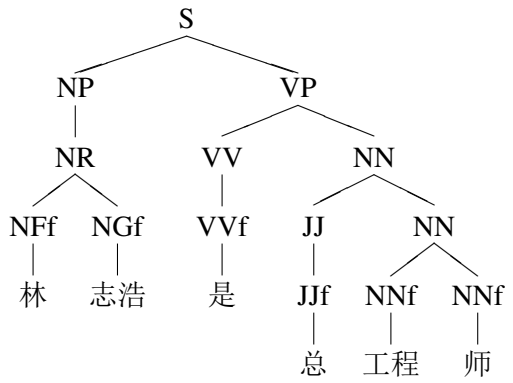


Figure 8: The actual output of our parser trained with a fully annotated treebank.

paradigm, our model's output also includes higher-level syntactic parsing results.

3.1 Training Data

We employ a statistical model to parse phrase and word structures as illustrated in Figure 8. The currently available treebank for us is the Penn Chinese Treebank (CTB) 5.0 (Xue et al., 2005). Because our model belongs to the family of head-driven statistical parsing models (Collins, 2003), we use the head-finding rules described by Sun and Jurafsky (2004).

Unfortunately, this treebank or any other treebanks for that matter, does not contain annotations of word structures. Therefore, we must annotate these structures by ourselves. The good news is that the annotation is not too complicated. First, we extract all words in the treebank and check each of them manually. Words with non-trivial structures are thus annotated. Finally, we install these small trees of words into the original treebank. Whether a word has structures or not is mostly context independent, so we only have to annotate each word once.

There are two noteworthy issues in this process. Firstly, as we'll see in Section 4, flat words and non-flat words will be modeled differently, thus it's important to adapt the part-of-speech tags to facilitate this modeling strategy. For example, the tag for nouns is NN as in 伊拉克 'Iraq' and 前总统 'former president'. After annotation, the former is flat, but the later has a structure (Figure 9). So we change the POS tag for flat nouns to NNf, then during bottom up parsing, whenever a new constituent ending with 'f' is found, we can assign it a probability in a way different from a structured word or phrase.

Secondly, we should record the head position of each word tree in accordance with the requirements of head driven parsing models. As an example, the right tree in Figure 9 has the context free rule "NN → JJf NNf", the head of which should be the rightmost NNf. Therefore, in 前总统 'former president' the head is 总统 'president'.

In passing, the readers should note the fact that in Figure 9, we have to add a parent labeled NN to the flat word 伊拉克 'Iraq' so as not to change the context-free rules contained inherently in the original treebank.

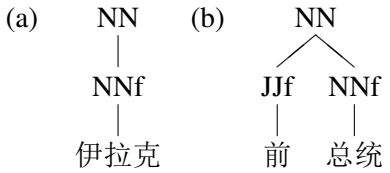


Figure 9: Example word structure annotation. We add an 'f' to the POS tags of words with no further structures.

4 The Model

Given an observed raw sentences S, our generative model tells a story about how this surface sequence of Chinese characters is generated with a linguistically plausible morphological and syntactical process, thereby defining a joint probability $\Pr(T, S)$ where T is a parse tree carrying word structures as well as phrase structures. With this model, the parsing problem is to search for the tree T^* such that

$$T^* = \arg\max_T \Pr(T, S) \tag{1}$$

The generation of S is defined in a top down fashion, which can be roughly summarized as follows. First, the lexicalized constituent structures are generated, then the lexicalized structure of each word is generated. Finally, flat words with no structures are generated. As soon as this is done, we get a tree whose leaves are Chinese characters and can be concatenated to get the surface character sequence S.

4.1 Generation of Constituent Structures

Each node in the constituent tree corresponds to a lexicalized context free rule

$$P \rightarrow L_n\, L_{n-1} \cdots L_1 H R_1\, R_2 \cdots R_m \tag{2}$$

where P, L_i, R_i and H are lexicalized nonterminals and H is the head. To generate this constituent, first P is generated, then the head child H is generated conditioned on P, and finally each L_i and R_j are generated conditioned on P and H and a distance metric. This breakdown of lexicalized PCFG rules is essentially the Model 2 defined by Collins (1999). We refer the readers to Collins' thesis for further details.

4.2 Generation of Words with Internal Structures

Words with rich internal structures can be described using a context-free grammar formalism as

$$\text{word} \rightarrow \text{root} \quad (3)$$
$$\text{word} \rightarrow \text{word suffix} \quad (4)$$
$$\text{word} \rightarrow \text{prefix word} \quad (5)$$

Here the root is any word without interesting internal structures, and the prefixes and suffixes are not limited to single characters. For example, 主义 'ism' as in 现代主义 'modernism' is a well known and very productive suffix. Also, we can see that rules (4) and (5) are recursive and hence can handle words with very complex structures.

By (3)–(5), the generation of word structures is exactly the same as that of ordinary phrase structures. Hence the probabilities of these words can be defined in the same way as higher level constituents in (2). Note that in our case, each word with structures is naturally lexicalized, since in the annotation process we have been careful to record the head position of each complex word.

As an example, consider a word $w = R(r)\,S(s)$ where R is the root part-of-speech headed by the word r, and S is the suffix part-of-speech headed by s. If the head of this word is its suffix, then we can define the probability of w by

$$\Pr(w) = \Pr(S, s) \cdot \Pr(R, r|S, s) \quad (6)$$

This is equivalent to saying that to generate w, we first generate its head $S(s)$, then conditioned on this head, other components of this word are generated. In actual parsing, because a word always occurs in some contexts, the above probability should also be conditioned on these contexts, such as its parent and the parent's head word.

4.3 Generation of Flat Words

We say a word is flat if it contains only one morpheme such as 伊拉克 'Iraq', or if it is a compound like 开发 'develop' which does not have a productive component we are currently interested in. Depending on whether a flat word is known or not, their generative probabilities are computed also differently. Generation of flat words seen in training is trivial and deterministic since every phrase and word structure rules are lexicalized.

However, the generation of unknown flat words is a different story. During training, words that occur less than 6 times are substituted with the symbol UNKNOWN. In testing, unknown words are generated after the generation of symbol UNKNOWN, and we define their probability by a first-order Markov model. That is, given a flat word $w = c_1 c_2 \cdots c_n$ not seen in training, we define its probability conditioned with the part-of-speech p as

$$\Pr(w|p) = \prod_{i=1}^{n+1} \Pr(c_i|c_{i-1}, p) \quad (7)$$

where c_0 is taken to be a START symbol indicating the left boundary of a word and c_{n+1} is the STOP symbol to indicate the right boundary. Note that the generation of w is only conditioned on its part-of-speech p, ignoring the larger constituent or word in which w occurs.

We use a back-off strategy to smooth the probabilities in (7):

$$\begin{aligned}\tilde{\Pr}(c_i|c_{i-1}, p) &= \lambda_1 \cdot \hat{\Pr}(c_i|c_{i-1}, p) \\ &\quad + \lambda_2 \cdot \hat{\Pr}(c_i|c_{i-1}) \\ &\quad + \lambda_3 \cdot \hat{\Pr}(c_i) \end{aligned} \quad (8)$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$ to ensure the conditional probability is well formed. These λs will be estimated with held-out data. The probabilities on the right side of (8) can be estimated with simple counts:

$$\hat{\Pr}(c_i|c_{i-1}, p) = \frac{\text{COUNT}(c_{i-1}c_i, p)}{\text{COUNT}(c_{i-1}, p)} \quad (9)$$

The other probabilities can be estimated in the same way.

4.4 Summary of the Generative Story

We make a brief summary of our generative story for the integrated morphological and syntactic parsing model. For a sentence S and its parse tree T, if we denote the set of lexicalized phrase structures in T by $\mathcal{C}$, the set of lexicalized word structures by $\mathcal{W}$, and the set of unknown flat words by $\mathcal{F}$, then the joint probability $\Pr(T, S)$ according to our model is

$$\Pr(T, S) = \prod_{c \in \mathcal{C}} \Pr(c) \prod_{w \in \mathcal{W}} \Pr(w) \prod_{f \in \mathcal{F}} \Pr(f) \quad (10)$$

In practice, the logarithm of this probability can be calculated instead to avoid numerical difficulties.

5 The Parsing Algorithm

To find the parse tree with highest probability we use a chart parser adapted from Collins (1999). Two key changes must be made to the search process, though. Firstly, because we are proposing a new paradigm for Chinese word segmentation, the input to the parser must be raw sentences by definition. Hence to use the bottom-up parser, we need a lexicon of all characters together with what roles they can play in a flat word. We can get this lexicon from the treebank. For example, from the word 中央/NNf 'center', we can extract a role bNNf for character 中 'middle' and a role eNNf for character 央 'center'. The role bNNf means the beginning of the flat label NNf, while eNNf stands for the end of the label NNf. This scheme was first proposed by Luo (2003) in his character-based Chinese parser, and we find it quite adequate for our purpose here.

Secondly, in the bottom-up parser for head driven models, whenever a new edge is found, we must assign it a probability and a head word. If the newly discovered constituent is a flat word (its label ends with 'f'), then we set its head word to be the concatenation of all its child characters, i.e. the word itself. If it is an unknown word, we use (7) to assign the probability, otherwise its probability is set to be 1. On the other hand, if the new edge is a phrase or word with internal structures, the probability is set according to (2), while the head word is found with the appropriate head rules. In this bottom-up way, the probability for a complete parse tree is known as soon as it is completed. This probability includes both word generation probabilities and constituent probabilities.

6 Evaluation

For several reasons, it is a little tricky to evaluate the accuracy of our model for integrated morphological and syntactic parsing. First and foremost, we currently know of no other same effort in parsing the structures of Chinese words, and we have to annotate word structures by ourselves. Hence there is no baseline performance to compare with. Secondly, simply reporting the accuracy of labeled precision and recall is not very informative because our parser takes raw sentences as input, and its output includes a lot of easy cases like word segmentation and part-of-speech tagging results.

Despite these difficulties, we note that higher-level constituent parsing results are still somewhat comparable with previous performance in parsing Penn Chinese Treebank, because constituent parsing does not involve word structures directly. Having said that, it must be pointed out that the comparison is meaningful only in a limited sense, as in previous literatures on Chinese parsing, the input is always word segmented or even part-of-speech tagged. That is, the bracketing in our case is around characters instead of words. Another observation is we can still evaluate Chinese word segmentation and part-of-speech tagging accuracy, by reading off the corresponding result from parse trees. Again because we split the words with internal structures into their components, comparison with other systems should be viewed with that in mind.

Based on these discussions, we divide the labels of all constituents into three categories:

Phrase labels are the labels in Peen Chinese Treebank for nonterminal phrase structures, including NP, VP, PP, etc.

POS labels represent part-of-speech tags such as NN, VV, DEG, etc.

Flat labels are generated in our annotation for words with no interesting structures. Recall that they always end with an 'f' such as NNf, VVf and DEGf, etc.

With this classification, we report our parser's accuracy for phrase labels, which is approximately the accuracy of constituent parsing of Penn Chinese Treebank. We report our parser's word segmentation accuracy based on the flat labels. This accuracy is in fact the joint accuracy of segmentation and part-of-speech tagging. Most importantly, we can report our parser's accuracy in recovering word structures based on POS labels and flat labels, since word structures may contain only these two kinds of labels.

With the standard split of CTB 5.0 data into training, development and test sets (Zhang and Clark,

2009), the result are summarized in Table 1. For all label categories, the PARSEEVAL measures (Abney et al., 1991) are used in computing the labeled precision and recall.

Types	LP	LR	F_1
Phrase	79.3	80.1	79.7
Flat	93.2	93.8	93.5
Flat*	97.1	97.6	97.3
POS & Flat	92.7	93.2	92.9

Table 1: Labeled precision and recall for the three types of labels. The line labeled 'Flat*' is for unlabeled metrics of flat words, which is effectively the ordinary word segmentation accuracy.

Though not directly comparable, we can make some remarks to the accuracy of our model. For constituent parsing, the best result on CTB 5.0 is reported to be 78% F_1 measure for unlimited sentences with automatically assigned POS tags (Zhang and Clark, 2009). Our result for phrase labels is close to this accuracy. Besides, the result for flat labels compares favorably with the state of the art accuracy of about 93% F_1 for joint word segmentation and part-of-speech tagging (Jiang et al., 2008; Kruengkrai et al., 2009). For ordinary word segmentation, the best result is reported to be around 97% F_1 on CTB 5.0 (Kruengkrai et al., 2009), while our parser performs at 97.3%, though we should remember that the result concerns flat words only. Finally, we see the performance of word structure recovery is almost as good as the recognition of flat words. This means that parsing word structures accurately is possible with a generative model.

It is interesting to see how well the parser does in recognizing the structure of words that were not seen during training. For this, we sampled 100 such words including those with prefixes or suffixes and personal names. We found that for 82 of these words, our parser can correctly recognize their structures. This means our model has learnt something that generalizes well to unseen words.

In error analysis, we found that the parser tends to over generalize for prefix and suffix characters. For example, 大作家 'great writer' is a noun phrase consisting of an adjective 大 'great' and a noun 作家 'writer', as shown in Figure 10(a), but our parser incorrectly analyzed it into a root 大作 'masterpiece' and a suffix 家 'expert', as in Figure 10(b). This

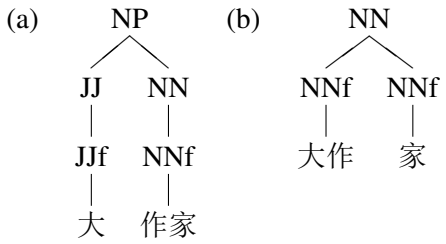


Figure 10: Example of parser error. Tree (a) is correct, and (b) is the wrong result by our parser.

is because the character 家 'expert' is a very productive suffix, as in 化学家 'chemist' and 外交家 'diplomat'. This observation is illuminating because most errors of our parser follow this pattern. Currently we don't have any non-ad hoc way of preventing such kind of over generalization.

7 Conclusion and Discussion

In this paper we proposed a new paradigm for Chinese word segmentation in which not only flat words were identified but words with structures were also parsed. We gave good reasons why this should be done, and we presented an effective method showing how this could be done. With the progress in statistical parsing technology and the development of large scale treebanks, the time has now come for this paradigm shift to happen. We believe such a new paradigm for word segmentation is linguistically justified and pragmatically beneficial to real world applications. We showed that word structures can be recovered with high precision, though there's still much room for improvement, especially for higher level constituent parsing.

Our model is generative, but discriminative models such as maximum entropy technique (Berger et al., 1996) can be used in parsing word structures too. Many parsers using these techniques have been proved to be quite successful (Luo, 2003; Fung et al., 2004; Wang et al., 2006). Another possible direction is to combine generative models with discriminative reranking to enhance the accuracy (Collins and Koo, 2005; Charniak and Johnson, 2005).

Finally, we must note that the use of flat labels such as "NNf" is less than ideal. The most impor-

tant reason these labels are used is we want to compare the performance of our parser with previous results in constituent parsing, part-of-speech tagging and word segmentation, as we did in Section 6. The problem with this approach is that word structures and phrase structures are then not treated in a truly unified way, and besides the 33 part-of-speech tags originally contained in Penn Chinese Treebank, another 33 tags ending with 'f' are introduced. We leave this problem open for now and plan to address it in future work.

Acknowledgments

I would like to thank Professor Maosong Sun for many helpful discussions on topics of Chinese morphological and syntactic analysis. The author is supported by NSFC under Grant No. 60873174. Heartfelt thanks also go to the reviewers for many pertinent comments which have greatly improved the presentation of this paper.

References

S. Abney, S. Flickenger, C. Gdaniec, C. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. Procedure for quantitatively comparing the syntactic coverage of English grammars. In E. Black, editor, *Proceedings of the workshop on Speech and Natural Language*, HLT '91, pages 306–311, Morristown, NJ, USA. Association for Computational Linguistics.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Daniel M. Bikel and David Chiang. 2000. Two statistical parsing models applied to the Chinese treebank. In *Second Chinese Language Processing Workshop*, pages 1–6, Hong Kong, China, October. Association for Computational Linguistics.

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

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the fourteenth national conference on artificial intelligence and ninth conference on Innovative applications of artificial intelligence*, AAAI'97/IAAI'97, pages 598–603. AAAI Press.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31:25–70, March.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Pascale Fung, Grace Ngai, Yongsheng Yang, and Benfeng Chen. 2004. A maximum-entropy Chinese parser augmented by transformation-based learning. *ACM Transactions on Asian Language Information Processing*, 3:159–168, June.

Jianfeng Gao, Andi Wu, Cheng-Ning Huang, Hong qiao Li, Xinsong Xia, and Hauwei Qin. 2004. Adaptive Chinese word segmentation. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 462–469, Barcelona, Spain, July.

Jianfeng Gao, Mu Li, Andi Wu, and Chang-Ning Huang. 2005. Chinese word segmentation and named entity recognition: A pragmatic approach. *Computational Linguistics*, 31(4):531–574.

Yoav Goldberg and Reut Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *Proceedings of ACL-08: HLT*, pages 371–379, Columbus, Ohio, June. Association for Computational Linguistics.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008. A cascaded linear model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL-08: HLT*, pages 897–904, Columbus, Ohio, June. Association for Computational Linguistics.

Wenbin Jiang, Liang Huang, and Qun Liu. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and POS tagging – a case study. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 522–530, Suntec, Singapore, August. Association for Computational Linguistics.

Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa, and Hitoshi Isahara. 2009. An error-driven word-character hybrid model for joint Chinese word segmentation and POS tagging. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Process-*

ing of the AFNLP, pages 513–521, Suntec, Singapore, August. Association for Computational Linguistics.

Zhongguo Li and Maosong Sun. 2009. Punctuation as implicit annotations for Chinese word segmentation. *Computational Linguistics*, 35:505–512, December.

Junhui Li, Guodong Zhou, and Hwee Tou Ng. 2010. Joint syntactic and semantic parsing of Chinese. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1108–1117, Uppsala, Sweden, July. Association for Computational Linguistics.

Xiaoqiang Luo. 2003. A maximum entropy Chinese character-based parser. In Michael Collins and Mark Steedman, editors, *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 192–199.

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 277–284, Barcelona, Spain, July. Association for Computational Linguistics.

Richard Sproat and Thomas Emerson. 2003. The first international Chinese word segmentation bakeoff. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, pages 133–143, Sapporo, Japan, July. Association for Computational Linguistics.

Richard Sproat, William Gale, Chilin Shih, and Nancy Chang. 1996. A stochastic finite-state word-segmentation algorithm for Chinese. *Computational Linguistics*, 22(3):377–404.

Honglin Sun and Daniel Jurafsky. 2004. Shallow semantc parsing of Chinese. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 249–256, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Mengqiu Wang, Kenji Sagae, and Teruko Mitamura. 2006. A fast, accurate deterministic parser for chinese. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 425–432, Sydney, Australia, July. Association for Computational Linguistics.

Andi Wu and Jiang Zixin. 1998. Word segmentation in sentence analysis. In *Proceedings of the 1998 International Conference on Chinese information processing*, Beijing, China.

Andi Wu. 2003. Customizable segmentation of morphologically derived words in Chinese. *Computational Linguistics and Chinese language processing*, 8(1):1–28.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese Treebank: phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 840–847, Prague, Czech Republic, June. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL-08: HLT*, pages 888–896, Columbus, Ohio, June. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the Chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies*, IWPT '09, pages 162–171, Morristown, NJ, USA. Association for Computational Linguistics.

Hai Zhao. 2009. Character-level dependencies in Chinese: Usefulness and learning. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 879–887, Athens, Greece, March. Association for Computational Linguistics.

Guodong Zhou and Jian Su. 2003. A Chinese efficient analyser integrating word segmentation, part-of-speech tagging, partial parsing and full parsing. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, pages 78–83, Sapporo, Japan, July. Association for Computational Linguistics.

A Simple Measure to Assess Non-response

Anselmo Peñas and Alvaro Rodrigo
UNED NLP & IR Group
Juan del Rosal, 16
28040 Madrid, Spain
{anselmo,alvarory@lsi.uned.es}

Abstract

There are several tasks where is preferable not responding than responding incorrectly. This idea is not new, but despite several previous attempts there isn't a commonly accepted measure to assess non-response. We study here an extension of *accuracy* measure with this feature and a very easy to understand interpretation. The measure proposed (*c@1*) has a good balance of discrimination power, stability and sensitivity properties. We show also how this measure is able to reward systems that maintain the same number of correct answers and at the same time decrease the number of incorrect ones, by leaving some questions unanswered. This measure is well suited for tasks such as Reading Comprehension tests, where multiple choices per question are given, but only one is correct.

1 Introduction

There is some tendency to consider that an incorrect result is simply the absence of a correct one. This is particularly true in the evaluation of Information Retrieval systems where, in fact, the absence of results sometimes is the worse output.

However, there are scenarios where we should consider the possibility of not responding, because this behavior has more value than responding incorrectly. For example, during the process of introducing new features in a search engine it is important to preserve users' confidence in the system. Thus, a system must decide whether it should give or not a result in the new fashion or keep on with the old kind of output. A similar example is the decision about showing or not ads related to the query. Showing wrong ads harms the business model more than showing nothing. A third example more related to Natural Language Processing is the Machine Reading evaluation through reading comprehension tests. In this case, where multiple choices for a question are offered, choosing a wrong option should be punished against leaving the question unanswered.

In the latter case, the use of utility functions is a very common option. However, utility functions give arbitrary value to not responding and ignore the system's behavior showed when it responds (see Section 2). To avoid this, we present c@1 measure (Section 2.2), as an extension of accuracy (the proportion of correctly answered questions). In Section 3 we show that no other extension produces a sensible measure. In Section 4 we evaluate c@1 in terms of stability, discrimination power and sensibility, and some real examples of its behavior are given in the context of Question Answering. Related work is discussed in Section 5.

2 Looking for the Value of Not Responding

Lets take the scenario of Reading Comprehension tests to argue about the development of the measure. Our scenario assumes the following:

- There are several questions.
- Each question has several options.
- One option is correct (and only one).

The first step is to consider the possibility of not responding. If the system responds, then the assessment will be one of two: correct or wrong. But if

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1415–1424,
Portland, Oregon, June 19-24, 2011.

the system doesn't respond there is no assessment. Since every question has a correct answer, non response is not correct but it is not incorrect either. This is represented in contingency Table 1, where:

- n_{ac}: number of questions for which the answer is correct
- n_{aw}: number of questions for which the answer is incorrect
- n_u: number of questions not answered
- n: number of questions ($n = n_{ac} + n_{aw} + n_u$)

	Correct (C)	**Incorrect (¬C)**
Answered (A)	n_{ac}	n_{aw}
Unanswered (¬A)	n_u	

Table 1: Contingency table for our scenario

Let's start studying a simple utility function able to establish the preference order we want:

- -1 if question receives an incorrect response
- 0 if question is left unanswered
- 1 if question receives a correct response

Let *U(i)* be the utility function that returns one of the above values for a given question *i*. Thus, if we want to consider *n* questions in the evaluation, the measure would be:

$$UF = \frac{1}{n}\sum_{i=1}^{n} U(i) = \frac{n_{ac} - n_{aw}}{n} \qquad (1)$$

The rationale of this utility function is intuitive: not answering adds no value and wrong answers add negative values. Positive values of *UF* indicate more correct answers than incorrect ones, while negative values indicate the opposite. However, the utility function is giving an arbitrary value to the preferences (-1, 0, 1).

Now we want to interpret in some way the value that Formula (1) assigns to unanswered questions. For this purpose, we need to transform Formula (1) into a more meaningful measure with a parameter for the number of unanswered questions (n_u). A monotonic transformation of (1) permit us to preserve the ranking produced by the measure. Let f(x)=0.5x+0.5 be the monotonic function to be used for the transformation. Applying this function to Formula (1) results in Formula (2):

$$\begin{aligned} 0.5\frac{n_{ac} - n_{aw}}{n} + 0.5 &= \frac{0.5}{n}\left[n_{ac} - n_{aw} + n\right] = \\ &= \frac{0.5}{n}\left[n_{ac} - n_{aw} + n_{ac} + n_{aw} + n_u\right] \\ &= \frac{0.5}{n}\left[2n_{ac} + n_u\right] = \frac{n_{ac}}{n} + 0.5\frac{n_u}{n} \end{aligned} \qquad (2)$$

Measure (2) provides the same ranking of systems than measure (1). The first summand of Formula (2) corresponds to *accuracy*, while the second is adding an arbitrary constant weight of 0.5 to the proportion of unanswered questions. In other words, *unanswered questions are receiving the same value as if half of them had been answered correctly*.

This does not seem correct given that not answering is being rewarded in the same proportion to all the systems, without taking into account the performance they have shown with the answered questions. We need to propose a more sensible estimation for the weight of unanswered questions.

2.1 A rationale for the Value of Unanswered Questions

According to the utility function suggested, unanswered questions would have value as if half of them had been answered correctly. Why half and not other value? Even more, Why a constant value? Let's generalize this idea and estate more clearly our hypothesis:

Unanswered questions have the same value as if a proportion of them would have been answered correctly.

We can express this idea according to contingency Table 1 in the following way:

$$\begin{aligned} P(C) &= P(C \cap A) + P(C \cap \neg A) = \\ &= P(C \cap A) + P(C/\neg A) * P(\neg A) \end{aligned} \qquad (3)$$

$P(C \cap A)$ can be estimated by n_{ac}/n, $P(\neg A)$ can be estimated by n_u/n, and we have to estimate $P(C/\neg A)$. Our hypothesis is saying that $P(C/\neg A)$

is different from 0. The utility measure (2) corresponds to P(C) in Formula (3) where $P(C/\neg A)$ receives a constant value of 0.5. It is assuming arbitrarily that $P(C/\neg A) = P(C/A)$.

Following this, our measure must consist of two parts: The overall *accuracy* and a better estimation of correctness over the unanswered questions.

2.2 The Measure Proposed: c@1

From the answered questions we have already observed the proportion of questions that received a correct answer ($P(C \cap A)$ = n_{ac}/n). We can use this observation as our estimation for $P(C/\neg A)$ instead of the arbitrary value of 0.5.

Thus, the measure we propose is c@1 (correctness at one) and is formally represented as follows:

$$c@1 = \frac{n_{ac}}{n} + \frac{n_{ac}}{n}\frac{n_u}{n} = \frac{1}{n}(n_{ac} + \frac{n_{ac}}{n}n_u) \quad (4)$$

The most important features of c@1 are:

1. A system that answers all the questions will receive a score equal to the traditional *accuracy* measure: n_u=0 and therefore *c@1*=n_{ac}/n.

2. Unanswered questions will add value to *c@1* as if they were answered with the *accuracy* already shown.

3. A system that does not return any answer would receive a score equal to 0 due to n_{ac}=0 in both summands.

According to the reasoning above, we can interpret *c@1* in terms of probability as $P(C)$ where $P(C/\neg A)$ has been estimated with $P(C \cap A)$. In the following section we will show that there is no other estimation for $P(C/\neg A)$ able to provide a reasonable evaluation measure.

3 Other Estimations for $P(C/\neg A)$

In this section we study whether other estimations of $P(C/\neg A)$ can provide a sensible measure for QA when unanswered questions are taken into account. They are:

1. $P(C/\neg A) \equiv 0$
2. $P(C/\neg A) \equiv 1$
3. $P(C/\neg A) \equiv P(\neg C/\neg A) \equiv 0.5$
4. $P(C/\neg A) \equiv P(C/A)$
5. $P(C/\neg A) \equiv P(\neg C/A)$

3.1 $P(C/\neg A) \equiv 0$

This estimation considers the absence of response as incorrect response and we have the traditional *accuracy* (n_{ac}/n).

Obviously, this is against our purposes.

3.2 $P(C/\neg A) \equiv 1$

This estimation considers all unanswered questions as correctly answered. This option is not reasonable and is given for completeness: systems giving no answer would get maximum score.

3.3 $P(C/\neg A) \equiv P(\neg C/\neg A) \equiv 0.5$

It could be argued that since we cannot have observations of correctness for unanswered questions, we should assume equiprobability between $P(C/\neg A)$ and $P(\neg C/\neg A)$. In this case, P(C) corresponds to the expression (2) already discussed. As previously explained, in this case we are giving an arbitrary constant value to unanswered questions independently of the system's performance shown with answered ones. This seems unfair. We should be aiming at rewarding those systems not responding instead of giving wrong answers, not reward the sole fact that the system is not responding.

3.4 $P(C/\neg A) \equiv P(C/A)$

An alternative is to estimate the probability of correctness for the unanswered questions as the precision observed over the answered ones: P(C/A)= n_{ac}/(n_{ac}+ n_{aw}). In this case, our measure would be like the one shown in Formula (5):

$$\begin{aligned} P(C) &= P(C \cap A) + P(C/\neg A) * P(\neg A) = \\ &= P(C/A) * P(A) + P(C/A) * P(\neg A) = \\ &= P(C/A) = \frac{n_{ac}}{n_{ac} + n_{aw}} \end{aligned} \quad (5)$$

The resulting measure is again the observed precision over the answered ones. This is not a sensible measure, as it would reward a cheating system that decides to leave all questions unanswered except one for which it is sure to have a correct answer.

Furthermore, from the idea that $P(C/\neg A)$ is equal to $P(C/A)$ the underlying assumption is that systems choose to answer or not to answer randomly, whereas we want to reward the systems that choose not responding because they are able to decide that their candidate options are wrong or because they are unable to decide which candidate is correct.

3.5 $P(C/\neg A) \equiv P(\neg C/A)$

The last option to be considered explores the idea that systems fail not responding in the same proportion that they fail when they give an answer (i.e. proportion of incorrect answers).

Estimating $P(C/\neg A)$ as n_{aw} / (n_{ac}+ n_{aw}), the measure would be:

$$\begin{aligned} P(C) &= P(C \cap A) + P(C/\neg A) * P(\neg A) = \\ &= P(C \cap A) * P(\neg C/A) * P(\neg A) = \\ &= \frac{n_{ac}}{n} + \frac{n_{aw}}{n_{ac} + n_{aw}} * \frac{n_u}{n} \end{aligned} \quad (6)$$

This measure is very easy to cheat. It is possible to obtain almost a perfect score just by answering incorrectly only one question and leaving unanswered the rest of the questions.

4 Evaluation of c@1

When a new measure is proposed, it is important to study the reliability of the results obtained using that measure. For this purpose, we have chosen the method described by Buckley and Voorhees (2000) for assessing the stability and discrimination power, as well as the method described by Voorhees and Buckley (2002) for examining the sensitivity of our measure. These methods have been used for studying IR metrics (showing similar results with the methods based on statistics (Sakai, 2006)), as well as for evaluating the reliability of other QA measures different to the ones studied here (Sakai, 2007a; Voorhees, 2002; Voorhees, 2003).

We have compared the results over c@1 with the ones obtained using both *accuracy* and the *utility function* (UF) defined in Formula (1). This comparison is useful to show how confident can a researcher be with the results obtained using each evaluation measure.

In the following subsections we will first show the data used for our study. Then, the experiments about stability and sensitivity will be described.

4.1 Data sets

We used the test collections and runs from the Question Answering track at the Cross Language Evaluation Forum 2009 (CLEF) (Peñas et al., 2010). The collection has a set of 500 questions with their answers. The 44 runs in different languages contain the human assessments for the answers given by actual participants. Systems could chose not to answer a question. In this case, they had the chance to submit their best candidate in order to assess the performance of their validation module (the one that decides whether to give or not the answer).

This data collection allows us to compare *c@1* and *accuracy* over the same runs.

4.2 Stability vs. Discrimination Power

The more stable a measure is, the lower the probability of errors associated with the conclusion *"system A is better than system B"* is. Measures with a high error must be used more carefully performing more experiments than in the case of using a measure with lower error.

In order to study the stability of c@1 and to compare it with *accuracy* we used the method described by Buckley and Voorhees (2000). This method allows also to study the number of times systems are deemed to be equivalent with respect to a certain measure, which reflects the *discrimination power* of that measure. The less discriminative the measure is, the more ties between systems there will be. This means that longer difference in scores will be needed for concluding which system is better (Buckley and Voorhees, 2000).

The method works as follows: let *S* denote a set of runs. Let *x* and *y* denote a pair of runs from *S*. Let *Q* denote the entire evaluation collection. Let *f* represents the fuzziness value, which is the percent difference between scores such that if the difference is smaller than *f* then the two scores are deemed to be equivalent. We apply the algorithm of Figure 1 to obtain the information needed for computing the error rate (Formula (7)). *Stability* is inverse to this value, the lower the error rate is, the more stable the measure is. The same algorithm gives us the

proportion of ties (Formula (8)), which we use for measuring *discrimination power*, that is the lower the proportion of ties is, the more discriminative the measure is.

```
for each pair of runs x,y ϵ S
  for each trial from 1 to 100
    Q_i = select at random subcol of size c from Q;
    margin = f * max (M(x,Q_i),M(y,Q_i));
    if(|M(x,Q_i) - M(y,Q_i)| < |margin|)
      EQ_M(x,y)++;
    else if(|M(x,Q_i) > M(y,Q_i)|)
      GT_M(x,y)++;
    else
      GT_M(y,x)++;
```

Figure 1: Algorithm for computing $EQ_M(x,y)$, $GT_M(x,y)$ and $GT_M(y,x)$ in the stability method

We assume that for each measure the correct decision about whether run x is better than run y happens when there are more cases where the value of x is better than the value of y. Then, the number of times y is better than x is considered as the number of times the test is misleading, while the number of times the values of x and y are equivalent is considered the number of ties.

On the other hand, it is clear that larger fuzziness values decrease the error rate but also decrease the discrimination power of a measure. Since a fixed fuzziness value might imply different trade-offs for different metrics, we decided to vary the fuzziness value from 0.01 to 0.10 (following the work by Sakai (2007b)) and to draw for each measure a *proportion-of-ties / error-rate* curve. Figure 2 shows these curves for the *c@1*, *accuracy* and *UF* measures. In the Figure we can see how there is a consistent decrease of the error rate of all measures when the proportion of ties increases (this corresponds to the increase in the fuzziness value). Figure 2 shows that the curves of *accuracy* and *c@1* are quite similar (slightly better behavior of c@1) , which means that they have a similar stability and discrimination power.

The results suggest that the three measures are quite stable, having *c@1* and *accuracy* a lower error rate than *UF* when the proportion of ties grows. These curves are similar to the ones obtained for other QA evaluation measures (Sakai, 2007a).

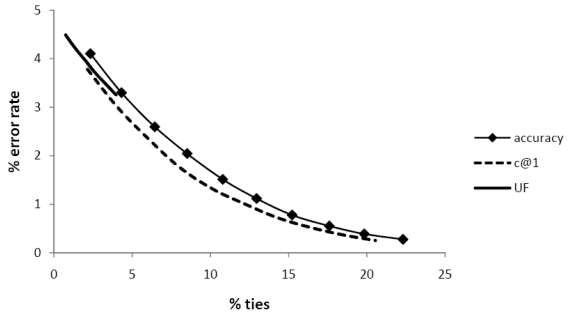


Figure 2: Error-rate / Proportion of ties curves for *accuracy*, *c@1* and *UF* with c = 250

4.3 Sensitivity

The *swap-rate* (Voorhees and Buckley, 2002) represents the chance of obtaining a discrepancy between two question sets (of the same size) as to whether a system is better than another given a certain difference bin. Looking at the swap-rates of all the difference performance bins, the performance difference required in order to conclude that a run is better than another for a given confidence value can be estimated. For example, if we want to know the required difference for concluding that system A is better than system B with a confidence of 95%, then we select the difference that represents the first bin where the swap-rate is lower or equal than 0.05.

The sensitivity of the measure is the number of times among all the comparisons in the experiment where this performance difference is obtained (Sakai, 2007b). That is, the more comparisons accomplish the estimated performance difference, the more *sensitive* is the measure. The more sensitive the measure, the more useful it is for system discrimination.

The swap method works as follows: let S denote a set of runs, let x and y denote a pair of runs from S. Let Q denote the entire evaluation collection. And let d denote a performance difference between two runs. Then, we first define 21 *performance difference bins*: the first bin represents performance differences between systems such that $0 \leq d < 0.01$; the second bin represents differences such that $0.01 \leq d < 0.02$; and the limits for the remaining bins increase by increments of 0.01, with the last bin containing all the differences equal or higher than 0.2.

$$Error\ rate_M = \frac{\sum_{x,y \epsilon S} min(GT_M(x,y), GT_M(y,x))}{\sum_{x,y \epsilon S}(GT_M(x,y) + GT_M(y,x) + EQ_M(x,y))} \quad (7)$$

$$Prop\ Ties_M = \frac{\sum_{x,y \epsilon S} EQ_M(x,y)}{\sum_{x,y \epsilon S}(GT_M(x,y) + GT_M(y,x) + EQ_M(x,y))} \quad (8)$$

Let BIN(d) denote a mapping from a difference *d* to one of the 21 bins where it belongs. Thus, algorithm in Figure 3 is applied for calculating the *swap-rate* of each bin.

```
for each pair of runs x,y ϵ S
  for each trial from 1 to 100
    select Q_i , Q_i' ⊂ Q, where
      Q_i ∩ Q_i' == φ and |Q_i| == |Q_i'| == c;
    d_M(Q_i) = M(x,Q_i) − M(y,Q_i);
    d_M(Q_i') = M(x,Q_i') − M(y,Q_i');
    counter(BIN(|d_M(Q_i)|))++;
    if(d_M(Q_i) * d_M(Q_i') < 0)
      swap_counter(BIN(|d_M(Q_i)|))++;
for each bin b
  swap_rate(b) = swap_counter(b)/counter(b);
```

Figure 3: Algorithm for computing swap-rates

	(i)	(ii)	(iii)	(iv)
UF	0.17	0.48	35.12%	59.30%
c@1	0.09	0.77	11.69%	58.40%
accuracy	0.09	0.68	13.24%	55.00%

Table 2: Results obtained applying the swap method to *accuracy*, *c@1* and *UF* at 95% of confidence, with c = 250: (i) Absolute difference required; (ii) Highest value obtained; (iii) Relative difference required ((i)/(ii)); (iv) percentage of comparisons that accomplish the required difference (sensitivity)

Given that Q_i and $Q_i^{'}$ must be disjoint, their size can only be up to half of the size of the original collection. Thus, we use the value c=250 for our experiment[1]. Table 2 shows the results obtained by applying the swap method to *accuracy*, *c@1* and *UF*, with c = 250, swap-rate ≤ 5, and sensitivity given a confidence of 95% (Column (iv)). The range of values are similar to the ones obtained for other measures according to (Sakai, 2007a).

According to Column (i), a higher absolute difference is required for concluding that a system is better than another using *UF*. However, the relative difference is similar to the one required by *c@1*. Thus, similar percentage of comparisons using *c@1* and *UF* accomplish the required difference (Column (iv)). These results show that their sensitivity values are similar, and higher than the value for *accuracy*.

4.4 Qualitative evaluation

In addition to the theoretical study, we undertook a study to interpret the results obtained by real systems in a real scenario. The aim is to compare the results of the proposed *c@1* measure with *accuracy* in order to compare their behavior. For this purpose we inspected the real systems runs in the data set.

System	c@1	accuracy	(i)	(ii)	(iii)
icia091ro	0.58	0.47	237	156	107
uaic092ro	0.47	0.47	236	264	0
loga092de	0.44	0.37	187	230	83
base092de	0.38	0.38	189	311	0

Table 3: Example of system results in QA@CLEF 2009. (i) number of questions correctly answered; (ii) number of questions incorrectly answered; (iii) number of unanswered questions.

Table 3 shows a couple of examples where two systems have answered correctly a similar number of questions. For example, this is the case of *icia091ro* and *uaic092ro* that, therefore, obtain almost the same *accuracy* value. However, *icia091ro* has returned less incorrect answers by not responding some questions. This is the kind of behavior we want to measure and reward. Table 3 shows how accuracy is sensitive only to the number of correct answers whereas *c@1* is able to distinguish when

[1]We use the same size for experiments in Section 4.2 for homogeneity reasons.

systems keep the number of correct answers but reduce the number of incorrect ones by not responding to some. The same reasoning is applicable to *loga092de* compared to *base092de* for German.

5 Related Work

The decision of leaving a query without response is related to the system ability to measure accurately its self-confidence about the correctness of their candidate answers. Although there have been one attempt to make the self-confidence score explicit and use it (Herrera et al., 2005), rankings are, usually, the implicit way to evaluate this self-confidence. Mean Reciprocal Rank (MRR) has traditionally been used to evaluate Question Answering systems when several answers per question were allowed and given in order (Fukumoto et al., 2002; Voorhees and Tice, 1999). However, as it occurs with *Accuracy* (proportion of questions correctly answered), the risk of giving a wrong answer is always preferred better than not responding.

The QA track at TREC 2001 was the first evaluation campaign in which systems were allowed to leave a question unanswered (Voorhees, 2001). The main evaluation measure was MRR, but performance was also measured by means of the percentage of answered questions and the portion of them that were correctly answered. However, no combination of these two values into a unique measure was proposed.

TREC 2002 discarded the idea of including unanswered questions in the evaluation. Only one answer by question was allowed and all answers had to be ranked according to the system's self-confidence in the correctness of the answer. Systems were evaluated by means of *Confidence Weighted Score* (*CWS*), rewarding those systems able to provide more correct answers at the top of the ranking (Voorhees, 2002). The formulation of CWS is the following:

$$CWS = \frac{1}{n}\sum_{i=1}^{n}\frac{C(i)}{i} \tag{9}$$

Where *n* is the number of questions, and *C(i)* is the number of correct answers up to the position *i* in the ranking. Formally:

$$C(i) = \sum_{j=1}^{i} I(j) \tag{10}$$

where *I(j)* is a function that returns 1 if answer *j* is correct and 0 if it is not. The formulation of *CWS* is inspired by the *Average Precision* (*AP*) over the ranking for one question:

$$AP = \frac{1}{R}\sum_{r} I(r)\frac{C(r)}{r} \tag{11}$$

where *R* is the number of known relevant results for a topic, and *r* is a position in the ranking. Since only one answer per question is requested, *R* equals to *n* (the number of questions) in *CWS*. However, in *AP* formula the summands belong to the positions of the ranking where there is a relevant result (product of I(r)), whereas in *CWS* every position of the ranking add value to the measure regardless of whether there is a relevant result or not in that position. Therefore, *CWS* gives much more value to some questions over others: questions whose answers are at the top of the ranking are giving almost the complete value to *CWS*, whereas those questions whose answers are at the bottom of the ranking are almost not counting in the evaluation.

Although *CWS* was aimed at promoting the development of better self-confidence scores, it was discussed as a measure for evaluating QA systems performance. *CWS* was discarded in the following campaigns of TREC in favor of *accuracy* (Voorhees, 2003). Subsequently, *accuracy* was adopted by the QA track at the Cross-Language Evaluation Forum from the beginning (Magnini et al., 2005).

There was an attempt to consider explicitly systems confidence self-score (Herrera et al., 2005): the use of the Pearson's correlation coefficient and the proposal of measures *K* and *K1* (see Formula 12). These measures are based in a utility function that returns -1 if the answer is incorrect and 1 if it is correct. This positive or negative value is weighted with the normalized confidence self-score given by the system to each answer. *K* is a variation of *K1* for being used in evaluations where more than an answer per question is allowed.

If the self-score is 0, then the answer is ignored and thus, this measure is permitting to leave a question unanswered. A system that always returns a

$$K1 = \frac{\sum_{i \epsilon \{correct_answers\}} self\_score(i) - \sum_{i \epsilon \{incorrect_answers\}} self\_score(i)}{n} \epsilon [-1, 1] \quad (12)$$

self-score equals to 0 (no answer) obtains a *K1* value of 0. However, the final value of *K1* is difficult to interpret: a positive value does not indicate necessarily more correct answers than incorrect ones, but that the sum of scores of correct answers is higher than the sum resulting from the scores of incorrect answers. This could explain the little success of this measure for evaluating QA systems in favor, again, of *accuracy* measure.

Accuracy is the simplest and most intuitive evaluation measure. At the same time is able to reward those systems showing good performance. However, together with MRR belongs to the set of measures that pushes in favor of giving always a response, even wrong, since there is no punishment for it. Thus, the development of better validation technologies (systems able to decide whether the candidate answers are correct or not) is not promoted, despite new QA architectures require them.

In effect, most QA systems during TREC and CLEF campaigns had an upper bound of accuracy around 60%. An explanation for this was the effect of error propagation in the most extended pipeline architecture: Passage Retrieval, Answer Extraction, Answer Ranking. Even with performances higher than 80% in each step, the overall performance drops dramatically just because of the product of partial performances. Thus, a way to break the pipeline architecture is the development of a module able to decide whether the QA system must continue or not its searching for new candidate answers: the Answer Validation module. This idea is behind the architecture of IBM's Watson (DeepQA project) that successfully participated at Jeopardy (Ferrucci et al., 2010).

In 2006, the first Answer Validation Exercise (AVE) proposed an evaluation task to advance the state of the art in Answer Validation technologies (Peñas et al., 2007). The starting point was the reformulation of Answer Validation as a Recognizing Textual Entailment problem, under the assumption that hypotheses can be automatically generated by combining the question with the candidate answer (Peñas et al., 2008a). Thus, validation was seen as a binary classification problem whose evaluation must deal with unbalanced collections (different proportion of positive and negative examples, correct and incorrect answers). For this reason, AVE 2006 used F-measure based on precision and recall for correct answers selection (Peñas et al., 2007). Other option is an evaluation based on the analysis of Receiver Operating Characteristic (ROC) space, sometimes preferred for classification tasks with unbalanced collections. A comparison of both approaches for Answer Validation evaluation is provided in (Rodrigo et al., 2011).

AVE 2007 changed its evaluation methodology with two objectives: the first one was to bring systems based on Textual Entailment to the Automatic Hypothesis Generation problem which is not part itself of the Recognising Textual Entailment (RTE) task but an Answer Validation need. The second one was an attempt to quantify the gain in QA performance when more sophisticated validation modules are introduced (Peñas et al., 2008b). With this aim, several measures were proposed to assess: the correct selection of candidate answers, the correct rejection of wrong answer and finally estimate the potential gain (in terms of accuracy) that Answer Validation modules can provide to QA (Rodrigo et al., 2008). The idea was to give value to the correctly rejected answers as if they could be correctly answered with the accuracy shown selecting the correct answers. This extension of accuracy in the Answer Validation scenario inspired the initial development of *c@1* considering non-response.

6 Conclusions

The central idea of this work is that not responding has more value than responding incorrectly. This idea is not new, but despite several attempts in TREC and CLEF there wasn't a commonly accepted mea-

sure to assess non-response. We have studied here an extension of *accuracy* measure with this feature, and with a very easy to understand rationale: Unanswered questions have the same value as if a proportion of them had been answered correctly, and the value they add is related to the performance (*accuracy*) observed over the answered questions. We have shown that no other estimation of this value produce a sensible measure.

We have shown also that the proposed measure *c@1* has a good balance of discrimination power, stability and sensitivity properties. Finally, we have shown how this measure rewards systems able to maintain the same number of correct answers and at the same time reduce the number of incorrect ones, by leaving some questions unanswered.

Among other tasks, measure *c@1* is well suited for evaluating Reading Comprehension tests, where multiple choices per question are given, but only one is correct. Non-response must be assessed if we want to measure effective reading and not just the ability to rank options. This is clearly not enough for the development of reading technologies.

Acknowledgments

This work has been partially supported by the Research Network MA2VICMR (S2009/TIC-1542) and Holopedia project (TIN2010-21128-C02).

References

Chris Buckley and Ellen M. Voorhees. 2000. Evaluating evaluation measure stability. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 33–40. ACM.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3).

Junichi Fukumoto, Tsuneaki Kato, and Fumito Masui. 2002. Question and Answering Challenge (QAC-1): Question Answering Evaluation at NTCIR Workshop 3. In *Working Notes of the Third NTCIR Workshop Meeting Part IV: Question Answering Challenge (QAC-1), pages 1-10*.

Jesús Herrera, Anselmo Peñas, and Felisa Verdejo. 2005. Question Answering Pilot Task at CLEF 2004. In *Multilingual Information Access for Text, Speech and Images, CLEF 2004, Revised Selected Papers.*, volume 3491 of *Lecture Notes in Computer Science, Springer*, pages 581–590.

Bernardo Magnini, Alessandro Vallin, Christelle Ayache, Gregor Erbach, Anselmo Peñas, Maarten de Rijke, Paulo Rocha, Kiril Ivanov Simov, and Richard F. E. Sutcliffe. 2005. Overview of the CLEF 2004 Multilingual Question Answering Track. In *Multilingual Information Access for Text, Speech and Images, CLEF 2004, Revised Selected Papers.*, volume 3491 of *Lecture Notes in Computer Science, Springer*, pages 371–391.

Anselmo Peñas, Álvaro Rodrigo, Valentín Sama, and Felisa Verdejo. 2007. Overview of the Answer Validation Exercise 2006. In *Evaluation of Multilingual and Multi-modal Information Retrieval, CLEF 2006, Revised Selected Papers*, volume 4730 of *Lecture Notes in Computer Science, Springer*, pages 257–264.

Anselmo Peñas, Álvaro Rodrigo, Valentín Sama, and Felisa Verdejo. 2008a. Testing the Reasoning for Question Answering Validation. In *Journal of Logic and Computation. 18(3)*, pages 459–474.

Anselmo Peñas, Álvaro Rodrigo, and Felisa Verdejo. 2008b. Overview of the Answer Validation Exercise 2007. In *Advances in Multilingual and Multimodal Information Retrieval, CLEF 2007, Revised Selected Papers*, volume 5152 of *Lecture Notes in Computer Science, Springer*, pages 237–248.

Anselmo Peñas, Pamela Forner, Richard Sutcliffe, Álvaro Rodrigo, Corina Forascu, Iñaki Alegria, Danilo Giampiccolo, Nicolas Moreau, and Petya Osenova. 2010. Overview of ResPubliQA 2009: Question Answering Evaluation over European Legislation. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments, CLEF 2009, Revised Selected Papers*, volume 6241 of *Lecture Notes in Computer Science, Springer*.

Alvaro Rodrigo, Anselmo Peñas, and Felisa Verdejo. 2008. Evaluating Answer Validation in Multi-stream Question Answering. In *Proceedings of the Second International Workshop on Evaluating Information Access (EVIA 2008)*.

Alvaro Rodrigo, Anselmo Peñas, and Felisa Verdejo. 2011. Evaluating Question Answering Validation as a classification problem. *Language Resources and Evaluation, Springer Netherlands (In Press)*.

Tetsuya Sakai. 2006. Evaluating Evaluation Metrics based on the Bootstrap. In *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, pages 525–532.

Tetsuya Sakai. 2007a. On the Reliability of Factoid Question Answering Evaluation. *ACM Trans. Asian Lang. Inf. Process.*, 6(1).

Tetsuya Sakai. 2007b. On the reliability of information retrieval metrics based on graded relevance. *Inf. Process. Manage.*, 43(2):531–548.

Ellen M. Voorhees and Chris Buckley. 2002. The effect of Topic Set Size on Retrieval Experiment Error. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 316–323.

Ellen M. Voorhees and Dawn M. Tice. 1999. The TREC-8 Question Answering Track Evaluation. In *Text Retrieval Conference TREC-8*, pages 83–105.

Ellen M. Voorhees. 2001. Overview of the TREC 2001 Question Answering Track. In *E. M. voorhees, D. K. Harman, editors: Proceedings of the Tenth Text REtrieval Conference (TREC 2001). NIST Special Publication 500-250*.

Ellen M. Voorhees. 2002. Overview of TREC 2002 Question Answering Track. In *E.M. Voorhees, L. P. Buckland, editors: Proceedings of the Eleventh Text REtrieval Conference (TREC 2002). NIST Publication 500-251*.

Ellen M. Voorhees. 2003. Overview of the TREC 2003 Question Answering Track. In *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003)*.

Improving Question Recommendation by Exploiting Information Need

Shuguang Li
Department of Computer Science
University of York, YO10 5DD, UK
sgli@cs.york.ac.uk

Suresh Manandhar
Department of Computer Science
University of York, YO10 5DD, UK
suresh@cs.york.ac.uk

Abstract

In this paper we address the problem of question recommendation from large archives of community question answering data by exploiting the users' information needs. Our experimental results indicate that questions based on the same or similar information need can provide excellent question recommendation. We show that translation model can be effectively utilized to predict the information need given only the user's query question. Experiments show that the proposed information need prediction approach can improve the performance of question recommendation.

1 Introduction

There has recently been a rapid growth in the number of community question answering (CQA) services such as Yahoo! Answers[1], Askville[2] and WikiAnswer[3] where people answer questions posted by other users. These CQA services have built up very large archives of questions and their answers. They provide a valuable resource for question answering research. Table 1 is an example from Yahoo! Answers web site. In the CQA archives, the title part is the user's query question, and the user's information need is usually expressed as natural language statements mixed with questions expressing their interests in the question body part.

In order to avoid the lag time involved with waiting for a personal response and to enable high quality answers from the archives to be retrieved, we need to search CQA archives of previous questions that are closely associated with answers. If a question is found to be interesting to the user, then a previous answer can be provided with very little delay. Question search and question recommendation are proposed to facilitate finding highly relevant or potentially interesting questions. Given a user's question as the query, *question search* tries to return the most semantically similar questions from the question archives. As the complement of question search, we define *question recommendation* as recommending questions whose information need is the same or similar to the user's original question. For example, the question "*What aspects of my computer do I need to upgrade ...*" with the information need "*... making a skate movie, my computer freezes, ...*" and the question "*What is the most cost effective way to expend memory space ...*" with information need "*... in need of more space for music and pictures ...*" are both good recommendation questions for the user in Table 1. So the recommended questions are not necessarily identical or similar to the query question.

In this paper, we discuss methods for question recommendation based on using the similarity between information need in the archive. We also propose two models to predict the information need based on the query question even if there's no information need expressed in the body of the question. We show that with the proposed models it is possible to recommend questions that have the same or similar information need.

The remainder of the paper is structured as fol-

[1] http://answers.yahoo.com
[2] http://askville.amazon.com
[3] http://wiki.answers.com

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1425–1434,
Portland, Oregon, June 19-24, 2011.

Q Title	If I want a faster computer should I buy more memory or storage space? ...
Q Body	I edit pictures and videos so I need them to work quickly. Any advice?
Answer	... If you are running out of space on your hard drive, then ... to boost your computer speed usually requires more RAM ...

Table 1: Yahoo! Answers question example

lows. In section 2, we briefly describe the related work on question search and recommendation. Section 3 addresses in detail how we measure the similarity between short texts. Section 4 describes two models for information need prediction that we use for the experiment. Section 5 tests the performance of the proposed models for the task of question recommendation. Section 7 is the conclusion of this paper.

2 Related Work

2.1 Question Search

Burke et al. (1997) combined a lexical metric and a simple semantic knowledge-based (WordNet) similarity method to retrieve semantically similar questions from frequently asked question (FAQ) data. Jeon et al. (2005a) retrieved semantically similar questions from Korean CQA data by calculating the similarity between their answers. The assumption behind their research is that questions with very similar answers tend to be semantically similar. Jeon et al. (2005b) also discussed methods for grouping similar questions based on using the similarity between answers in the archive. These grouped question pairs were further used as training data to estimate probabilities for a translation-based question retrieval model. Wang et al. (2009) proposed a tree kernel framework to find similar questions in the CQA archive based on syntactic tree structures. Wang et al. (2010) mined lexical and syntactic features to detect question sentences in CQA data.

2.2 Question Recommendation

Wu et al. (2008) presented an incremental automatic question recommendation framework based on probabilistic latent semantic analysis. Question recommendation in their work considered both the users' interests and feedback. Duan et al. (2008) made use of a tree-cut model to represent questions as graphs of topic terms. Questions were recommended based on this topic graph. The recommended questions can provide different aspects around the topic of the query question.

The above question search and recommendation research provide different ways to retrieve questions from large archives of question answering data. However, none of them considers the similarity or diversity between questions by exploring their information needs.

3 Short Text Similarity Measures

In question retrieval systems accurate similarity measures between documents are crucial. Most traditional techniques for measuring the similarity between two documents mainly focus on comparing word co-occurrences. The methods employing this strategy for documents can usually achieve good results, because they may share more common words than short text snippets. However the state-of-the-art techniques usually fail to achieve desired results due to short questions and information need texts.

In order to measure the similarity between short texts, we make use of three kinds of text similarity measures: TFIDF based, Knowledge based and Latent Dirichlet Allocation (LDA) based similarity measures in this paper. We will compare their performance for the task of question recommendation in the experiment section.

3.1 TFIDF

Baeza-Yates and Ribeiro-Neto (1999) provides a TFIDF method to calculate the similarity between two texts. Each document is represented by a term vector using TFIDF score. The similarity between two text D_i and D_j is the cosine similarity in the vector space model:

$$cos(D_i, D_j) = \frac{D_i^T D_j}{\|D_i\|\|D_j\|}$$

This method is used in most information retrieval systems as it is both efficient and effective. However if the query text contains only one or two words this method will be biased to shorter answer texts (Jeon et al., 2005a). We also found that in CQA data short contents in the question body cannot provide any information about the users' information needs. Based on the above two reasons, in the test data sets we do not include the questions whose information need parts contain only a few noninformative words .

3.2 Knowledge-based Measure

Mihalcea et al. (2006) proposed several knowledge-based methods for measuring the semantic level similarity of texts to solve the lexical chasm problem between short texts. These knowledge-based similarity measures were derived from word semantic similarity by making use of WordNet. The evaluation on a paraphrase recognition task showed that knowledge-based measures outperform the simpler lexical level approach.

We follow the definition in (Mihalcea et al., 2006) to derive a text-to-text similarity metric mcs for two given texts D_i and D_j:

$$mcs(D_i, D_j) = \frac{\sum_{w \in D_i} maxSim(w, D_j) * idf(w)}{\sum_{w \in D_i} idf(w)} + \frac{\sum_{w \in D_j} maxSim(w, D_i) * idf(w)}{\sum_{w \in D_j} idf(w)}$$

For each word w in D_i, $maxSim(w, D_j)$ computes the maximum semantic similarity between w and any word in D_j. In this paper we choose lin (Lin, 1998) and jcn (Jiang and Conrath, 1997) to compute the word-to-word semantic similarity.

We only choose nouns and verbs for calculating mcs. Additionally, when w is a noun we restrict the words in document D_i (and D_j) to just nouns. Similarly, when w is a verb, we restrict the words in document D_i (and D_j) to just verbs.

3.3 Probabilistic Topic Model

Celikyilmaz et al. (2010) presented probabilistic topic model based methods to measure the similarity between question and candidate answers. The candidate answers were ranked based on the hidden topics discovered by Latent Dirichlet Allocation (LDA) methods.

In contrast to the TFIDF method which measures "common words", short texts are not compared to each other directly in probabilistic topic models. Instead, the texts are compared using some "third-party" topics that relate to them. A passage D in the retrieved documents (document collection) is represented as a mixture of fixed topics, with topic z getting weight $\theta_z^{(D)}$ in passage D and each topic is a distribution over a finite vocabulary of words, with word w having a probability $\phi_w^{(z)}$ in topic z. Gibbs Sampling can be used to estimate the corresponding expected posterior probabilities $P(z|D) = \hat{\theta}_z^{(D)}$ and $P(w|z) = \hat{\phi}_w^{(z)}$ (Griffiths and Steyvers, 2004).

In this paper we use two LDA based similarity measures in (Celikyilmaz et al., 2010) to measure the similarity between short information need texts. The first LDA similarity method uses KL divergence to measure the similarity between two documents under each given topic:

$$sim_{LDA1}(D_i, D_j) = \frac{1}{K} \sum_{k=1}^{K} 10^{W(D_i^{(z=k)}, D_j^{(z=k)})}$$

$$W(D_i^{(z=k)}, D_j^{(z=k)}) = - KL(D_i^{(z=k)} \| \frac{D_i^{(z=k)} + D_j^{(z=k)}}{2}) - KL(D_j^{(z=k)} \| \frac{D_i^{(z=k)} + D_j^{(z=k)}}{2})$$

$W(D_i^{(z=k)}, D_j^{(z=k)})$ calculates the similarity between two documents under topic $z = k$ using KL divergence measure. $D_i^{(z=k)}$ is the probability distribution of words in document D_i given a fixed topic z.

The second LDA similarity measure from (Griffiths and Steyvers, 2004) treats each document as a probability distribution of topics:

$$sim_{LDA2}(D_i, D_j) = 10^{W(\hat{\theta}^{(D_i)}, \hat{\theta}^{(D_j)})}$$

where $\hat{\theta}^{(D_i)}$ is document D_i's probability distribution of topics as defined earlier.

4 Information Need Prediction using Statistical Machine Translation Model

There are two reasons that we need to predict information need. It is often the case that the query question does not have a question body part. So we need a model to predict the information need part based on the query question in order to recommend questions based on the similarity of their information needs. Another reason is that information need prediction plays a crucial part not only in Question Answering but also in information retrieval (Liu et al., 2008). In this paper we propose an information need prediction method based on a statistical machine translation model.

4.1 Statistical Machine Translation Model

($\mathbf{f}^{(s)}$, $\mathbf{e}^{(s)}$), s = 1,...,S is a parallel corpus. In a sentence pair (**f**, **e**), source language String, **f** = $f_1 f_2 ... f_J$ has J words, and **e** = $e_1 e_2 ... e_I$ has I words. And alignment **a** = $a_1 a_2 ... a_J$ represents the mapping information from source language words to target words.

Statistical machine translation models estimate $Pr(\mathbf{f}|\mathbf{e})$, the translation probability from source language string **e** to target language string **f** (Och et al., 2003):

$$Pr(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} Pr(\mathbf{f}, \mathbf{a}|\mathbf{e})$$

EM-algorithm is usually used to train the alignment models to estimate lexicon parameters $p(f|e)$.

In E-step, the counts for one sentence pair (**f** ,**e**) are:

$$c(f|e; \mathbf{f}, \mathbf{e}) = \sum_{\mathbf{a}} Pr(\mathbf{a}|\mathbf{f}, \mathbf{e}) \sum_{i,j} \delta(f, f_j)\delta(e, e_{a_j})$$

$$Pr(\mathbf{a}|\mathbf{f}, \mathbf{e}) = Pr(\mathbf{f}, \mathbf{a}|\mathbf{e}) / Pr(\mathbf{a}|\mathbf{e})$$

In the M-step, lexicon parameters become:

$$p(f|e) \propto \sum_{s} c(f|e; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})$$

Different alignment models such as IBM-1 to IBM-5 (Brown et al., 1993) and HMM model (Och and Ney, 2000) provide different decompositions of $Pr(\mathbf{f}, \mathbf{a}|\mathbf{e})$. For different alignment models different approaches were proposed to estimate the corresponding alignments and parameters. The details can be found in (Och et al., 2003; Brown et al., 1993).

4.2 Information Need Prediction

After estimating the statistical translation probabilities, we treat the information need prediction as the process of ranking words by $p(w|Q)$, the probability of generating word w from question Q:

$$P(w|Q) = \lambda \sum_{t \in Q} P_{tr}(w|t)P(t|Q) + (1-\lambda)P(w|C)$$

The word-to-word translation probability $P_{tr}(w|t)$ is the probability of word w is translated from a word t in question Q using the translation model. The above formula uses linear interpolation smoothing of the document model with the background language model $P(t|C)$. λ is the smoothing parameter. $P(t|Q)$ and $P(t|C)$ are estimated using the maximum likelihood estimator.

One important consideration is that statistical machine translation models first estimate $Pr(\mathbf{f}|\mathbf{e})$ and then calculate $Pr(\mathbf{e}|\mathbf{f})$ using Bayes' theorem to minimize ordering errors (Brown et al., 1993):

$$Pr(\mathbf{e}|\mathbf{f}) = \frac{Pr(\mathbf{f}|\mathbf{e})Pr(\mathbf{e})}{Pr(\mathbf{f})}$$

But in this paper, we skip this step as we found out the order of words in information need part is not an important factor. In our collected CQA archive, question title and information need pairs can be considered as a type of parallel corpus, which is used for estimating word-to-word translation probabilities. More specifically, we estimated the IBM-4 model by *GIZA++*[4] with the question part as the source language and information need part as the target language.

5 Experiments and Results

5.1 Text Preprocessing

The questions posted on community QA sites often contain spelling or grammar errors. These errors in-

[4] http://fjoch.com/GIZA++.html

	Test_c			Test_t		
Methods	MRR	Precision@5	Precision@10	MRR	Precision@5	Precision@10
TFIDF	84.2%	67.1%	61.9%	92.8%	74.8%	63.3%
Knowledge1	82.2%	65.0%	65.6%	78.1%	67.0%	69.6%
Knowledge2	76.7%	54.9%	59.3%	61.6%	53.3%	58.2%
LDA1	92.5%	68.8%	64.7%	91.8%	75.4%	69.8%
LDA2	61.5%	55.3%	60.2%	52.1%	57.4%	54.5%

Table 2: Question recommendation results without information need prediction

	Test_c			Test_t		
Methods	MRR	Precision@5	Precision@10	MRR	Precision@5	Precision@10
TFIDF	86.2%	70.8%	64.3%	95.1%	77.8%	69.3%
Knowledge1	82.2%	65.0%	66.6%	76.7%	68.0%	68.7%
Knowledge2	76.7%	54.9%	60.2%	61.6%	53.3%	58.2%
LDA1	95.8%	72.4%	68.2%	96.2%	79.5%	69.2%
LDA2	61.5%	55.3%	58.9%	68.1%	58.3%	53.9%

Table 3: Question recommendation results with information need predicted by translation model

fluence the calculation of similarity and the performance of information retrieval (Zhao et al., 2007; Bunescu and Huang, 2010). In this paper, we use an open source software *afterthedeadline*[5] to automatically correct the spelling errors in the question and information need texts first. We also made use of Web 1T 5-gram[6] to implement an N-Gram based method (Cheng et al., 2008) to further filter out the false positive corrections and re-rank correction suggestions (Mudge, 2010). The texts are tagged by Brill's Part-of-Speech Tagger[7] as the rule-based tagger is more robust than the state-of-art statistical taggers for raw web contents. This tagging information is only used for WordNet similarity calculation. Stop word removal and lemmatization are applied to the all the raw texts before feeding into machine translation model training, the LDA model estimating and similarity calculation.

5.2 Construction of Training and Testing Sets

We made use of the questions crawled from Yahoo! Answers for the estimating models and evaluation. More specifically, we obtained 2 million questions under two categories at Yahoo! Answers: 'travel' (1 million), and 'computers&internet' (1 million). Depending on whether the best answers have been chosen by the asker, questions from Yahoo! answers can be divided into 'resolved' and 'unresolved' categories. From each of the above two categories, we randomly selected 200 resolved questions to construct two testing data sets: 'Test_t' ('travel'), and 'Test_c' ('computers&internet'). In order to measure the information need similarity in our experiment we selected only those questions whose information needs part contained at least 3 informative words after stop word removal. The rest of the questions 'Train_t' and 'Train_c' under the two categories are left for estimating the LDA topic models and the translation models. We will show how we obtain these models later.

5.3 Experimental Setup

For each question (query question) in 'Test_t' or 'Test_c', we used the words in the question title part as the main search query and the other words in the information need part as search query expansion to retrieve candidate recommended questions from Yahoo! Answers website. We obtained an average of 154 resolved questions under 'travel' or 'computers&internet' category, and three assessors were involved in the manual judgments.

Given a question returned by a recommendation

[5]http://afterthedeadline.com

[6]http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T13

[7]http://www.umiacs.umd.edu/ jimmylin/resources.html

method, two assessors are asked to label it with 'good' or 'bad'. The third assessor will judge the conflicts. The assessors are also asked to read the information need and answer parts. If a recommended question is considered to express the same or similar information need, the assessor will label it 'good'; otherwise, the assessor will label it as 'bad'.

Three measures for evaluating the recommendation performance are utilized. They are Mean Reciprocal Rank (MRR), top five prediction accuracy (precision@5) and top ten prediction accuracies (precision@10) (Voorhees and Tice, 2004; Cao et al., 2008). In MRR the reciprocal rank of a query question is the multiplicative inverse of the rank of the first 'good' recommended question. The top five prediction accuracy for a query question is the number of 'good' recommended questions out of the top five ranked questions and the top ten accuracy is calculated out of the top ten ranked questions.

5.4 Similarity Measure

The first experiment conducted question recommendation based on their information need parts. Different text similarity methods described in section 3 were used to measure the similarity between the information need texts. In TFIDF similarity measure (TFIDF), the idf values for each word were computed from frequency counts over the entire *Aquaint* corpus[8]. For calculating the word-to-word knowledge-based similarity, a WordNet::Similarity Java implementation[9] of the similarity measures lin (Knowledge2) and jcn (Knowledge1) is used in this paper. For calculating topic model based similarity, we estimated two LDA models from 'Train_t' and 'Train_c' using *GibbsLDA++*[10]. We treated each question including the question title and the information need part as a single document of a sequence of words. These documents were preprocessed before being fed into LDA model. 1800 iterations for Gibbs sampling 200 topics parameters were set for each LDA model estimation.

The results in table 2 show that TFIDF and LDA1 methods perform better for recommending questions than the others. After further analysis of the questions recommended by both methods, we discov-

Q1:	If I want a faster computer should I buy more memory or storage space?
InfoN	If I want a faster computer should I buy more memory or storage space? Whats the difference? I edit pictures and videos so I need them to work quickly. ...
RQ1	Would buying 1gb memory upgrade make my computer faster?
InfoN	I have an inspiron B130. It has 512mb memory now. I would add another 1gb into 2nd slot ...
RQ2	whats the difference between memory and hard drive space on a computer and why is.....?
InfoN	see I am starting edit videos on my computer but i am running out of space. why is so expensive to buy memory but not external drives? ...
Q2:	Where should my family go for spring break?
InfoN	... family wants to go somewhere for a couple days during spring break ... prefers a warmer climate and we live in IL, so it shouldn't be SUPER far away. ... a family road trip. ...
RQ1	Whats a cheap travel destination for spring break?
InfoN	I live in houston texas and i'm trying to find i inexpensive place to go for spring break with my family.My parents don't want to spend a lot of money due to the economy crisis, ... a fun road trip...
RQ2	Alright you creative deal-seekers, I need some help in planning a spring break trip for my family
InfoN	Spring break starts March 13th and goes until the 21st ... Someplace WARM!!! Family-oriented hotel/resort ... North American Continent (Mexico, America, Jamaica, Bahamas, etc.) Cost= Around $5,000 ...

Table 4: Question recommendation results by LDA measuring the similarity between information needs

[8]http://ldc.upenn.edu/Catalog/docs/LDC2002T31

[9]http://cogs.susx.ac.uk/users/drh21/

[10]http://gibbslda.sourceforge.net

ered that the ordering of the recommended questions from TFIDF and LDA1 are quite different. TFIDF similarity method prefers texts with more common words, while the LDA1 method can find the relation between the non-common words between short texts based on a series of third-party topics. The LDA1 method outperforms the TFIDF method in two ways: (1) the top recommended questions' information needs share less common words with the query question's; (2) the top recommended questions span wider topics. The questions highly recommended by LDA1 can suggest more useful topics to the user.

Knowledge-based methods are also shown to perform worse than TFIDF and LDA1. We found that some words were mis-tagged so that they were not included in the word-to-word similarity calculation. Another reason for the worse performance is that the words out of the WordNet dictionary were also not included in the similarity calculation.

The Mean Reciprocal Rank score for TFIDF and LDA1 are more than 80%. That is to say, we are able to recommend questions to the users by measuring their information needs. The first two recommended questions for Q1 and Q2 using LDA1 method are shown in table 4. InfoN is the information need part associated with each question.

In the preprocessing step, some words were successfully corrected such as "*What should I do this saturday? ... and staying in a* ***hotell*** *...*" and "*my* ***faimly*** *is traveling to* ***florda*** *...*". However, there are still a small number of texts such as "*How come my* ***Gforce*** *visualization doesn't work?*" and "*Do i need an Id to travel from new york to* ***maimi****?*" failed to be corrected. So in the future, a better method is expected to correct these failure cases.

5.5 Information Need Prediction

There are some retrieved questions whose information need parts are empty or become empty or almost empty (one or two words left) after the preprocessing step. The average number of such retrieved questions for each query question is 10 in our experiment. The similarity ranking scores of these questions are quite low or zero in the previous experiment. In this experiment, we will apply information need prediction to the questions whose information needs are missing in order to find out whether we improve the recommendation task.

The question and information need pairs in both 'Train_t' and 'Train_c' training sets were used to train two IBM-4 translation models by *GIZA++* toolkit. These pairs were also preprocessed before training. And the pairs whose information need part become empty after preprocessing were disregarded.

During the experiment, we found that some of the generated words in the information need parts are themselves. This is caused by the self translation problem in translation model: the highest translation score for a word is usually given to itself if the target and source languages are the same (Xue et al., 2008). This has always been a tough question: not using self-translated words can reduce retrieval performance as the information need parts need the terms to represent the semantic meanings; using self-translated words does not take advantage of the translation approach. To tackle this problem, we control the number of the words predicted by the translation model to be exactly twice the number of words in the corresponding preprocessed question.

The predicted information need words for the retrieved questions are shown in Table 5. In Q1, the information need behind question "*recommend website for custom built computer parts*" may imply that the users need to know some information about building computer parts such as "*ram*" and "*motherboard*" for a different purpose such as "*gaming*". While in Q2, the user may want to compare computers in different brands such as "*dell*" and "*mac*" or consider the "*price*" factor for "*purchasing a laptop for a college student*".

We also did a small scale comparison between the generated information needs against the real questions whose information need parts are not empty. Q3 and Q4 in Table 5 are two examples. The original information need for Q3 is "*looking for beautiful beaches and other things to do such as museums, zoos, shopping, and great seafood*" in CQA. The generated content for Q3 contains words in wider topics such as '*wedding*', '*surf*' and the price information ('*cheap*'). This reflects that there are some other users asking similar questions with the same or other interests.

From the results in Table 3, we can see that the performance of most similarity methods were improved by making use of information need predic-

tion. Different similarity measures received different degrees of improvement. LDA1 obtained the highest improvement followed by the TFIDF based method. These two approaches are more sensitive to the contents generated by a translation model.

However we found out that in some cases the LDA1 model failed to give higher scores to good recommendation questions. For example, Q5, Q6, and Q7 in table 5 were retrieved as recommendation candidates for the query question in Table 1. All of the three questions were good recommendation candidates, but only Q6 ranked fifth while Q5 and Q7 were out of the top 30 by LDA1 method. Moreover, in a small number of cases bad recommendation questions received higher scores and jeopardized the performance. For example, for query question "*How can you add subtitles to videos?*" with information need "*... add subtitles to a music video ... got off youtube ...download for this ...*", a retrieved question "*How would i add a music file to a video clip. ...*" was highly recommended by TFIDF approach as predicted information need contained '*youtube*', '*video*', '*music*', '*download*', ...

The MRR score received an improvement from 92.5% to 95.8% in the 'Test_c' and from 91.8% to 96.2% in 'Test_t'. This means that the top one question recommended by our methods can be quite well catering to the users' information needs. The top five precision and the top ten precision scores using TFIDF and LDA1 methods also received different degrees of improvement. Thus, we can improve the performance of question recommendation by predicting information needs.

Q1:	Please recommend A good website for Custom Built Computer parts?
InfoN	custom, site, ram, recommend, price, motherboard, gaming, ...
Q2:	What is the best laptop for a college student?
InfoN	know, brand, laptop, college, buy, price, dell, mac, ...
Q3:	What is the best Florida beach for a honeymoon?
InfoN	Florida, beach, honeymoon, wedding, surf, cheap, fun, ...
Q4:	Are there any good clubs in Manchester
InfoN	club, bar, Manchester, music, age, fun, drink, dance, ...
Q5:	If i buy a video card for my computer will that make it faster?
InfoN	nvidia, video, ati, youtube, card, buy, window, slow, computer, graphics, geforce, faster, ...
Q6:	If I buy a bigger hard drive for my laptop, will it make my computer run faster or just increase the memory?
InfoN	laptop, ram, run, buy, bigger, memory, computer, increase, gb, hard, drive, faster, ...
Q7:	Is there a way I can make my computer work faster rather than just increasing the ram or harware space?
InfoN	space, speed, ram, hardware, main, gig, slow, computer, increase, work, gb, faster, ...

Table 5: Information need prediction examples using IBM-4 translation model

6 Conclusions

In this paper we addressed the problem of recommending questions from large archives of community question answering data based on users' information needs. We also utilized a translation model and a LDA topic model to predict the information need only given the user's query question. Different information need similarity measures were compared to prove that it is possible to satisfy user's information need by recommending questions from large archives of community QA. The Latent Dirichlet allocation based approach was proved to perform better on measuring the similarity between short

texts in the semantic level than traditional methods. Experiments showed that the proposed translation based language model for question information need prediction further enhanced the performance of question recommendation methods.

References

Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, Robert L. Mercer. 1993. *The mathematics of statistical machine translation: parameter estimation.* Computational Linguistics, v.19 n.2, June 1993.

Razvan Bunescu and Yunfeng Huang. 2010. *Learning the Relative Usefulness of Questions in Community QA.* Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP) , Cambridge, MA.

Robin D. Burke and Kristian J. Hammond and Vladimir A. Kulyukin and Steven L. Lytinen and Noriko Tomuro and Scott Schoenberg. 1997. *Question answering from frequently-asked question files: Experiences with the FAQ Finder system.* AI Magazine, 18, 57C66.

Yunbo Cao, Huizhong Duan, Chin-Yew Lin, Yong Yu, and Hsiao-Wuen Hon. 2008. *Recommending Questions Using the MDL-based Tree Cut Model.* In: Proc. of the 17th Int. Conf. on World Wide Web, pp. 81-90.

Asli Celikyilmaz and Dilek Hakkani-Tur and Gokhan Tur. 2010. *LDA Based Similarity Modeling for Question Answering.* In NAACL 2010 C Workshop on Semantic Search.

Charibeth Cheng, Cedric Paul Alberto, Ian Anthony Chan, and Vazir Joshua Querol. 2008. *SpellCheF: Spelling Checker and Corrector for Filipino.* Journal of Research in Science, Computing and Engineering, North America, 4, sep. 2008.

Lynn Silipigni Connaway and Chandra Prabha. 2005. *An overview of the IMLS Project "Sense-making the information confluence: The whys and hows of college and university user satisficing of information needs"*. Presented at Library of Congress Forum, American Library Association Midwinter Conference, Boston, MA, Jan 16, 2005.

Huizhong Duan, Yunbo Cao, Chin-Yew Lin, and Yong Yu. 2008. *Searching questions by identifying question topic and question focus.* In HLT-ACL, pages 156C164.

Thomas L. Griffiths and Mark Steyvers. 2004. *Finding scientific topics.* Natl Acad Sci 101:5228C5235.

Jiwoon Jeon, W. Bruce Croft and Joon Ho Lee. 2005a. *Finding semantically similar questions based on their answers.* In Proc. of SIGIR05.

Jiwoon Jeon, W. Bruce Croft and Joon Ho Lee. 2005b. *Finding similar questions in large question and answer archives.* In CIKM, pages 84C90.

Jay J. Jiang and David W. Conrath. 1997. *Semantic similarity based on corpus statistics and lexical taxonomy.* In Proceedings of International Conference on Research in Computational Linguistics, Taiwan.

Dekang Lin. 1998. *An Information-Theoretic Definition of Similarity.* In Proceedings of the Fifteenth International Conference on Machine Learning (ICML '98), Jude W. Shavlik (Ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 296-304.

Yandong Liu, Jiang Bian, and Eugene Agichtein. 2008. *Predicting information seeker satisfaction in community question answering.* In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '08). ACM, New York, NY, USA, 483-490.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. *Corpus-based and knowledge-based measures of text semantic similarity.* In Proceedings of the 21st national conference on Artificial intelligence (AAAI '06), pages 775C780. AAAI Press.

Raphael Mudge. 2010. *The design of a proofreading software service.* In Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics and Writing: Writing Processes and Authoring Aids (CL&W '10). Association for Computational Linguistics, Morristown, NJ, USA, 24-32.

Franz Josef Och, Hermann Ney. 2000. *A comparison of alignment models for statistical machine translation.* Proceedings of the 18th conference on Computational linguistics, July 31-August 04, Saarbrucken, Germany.

Franz Josef Och, Hermann Ney. 2003.*A Systematic Comparison of Various Statistical Alignment Models.* Computational Linguistics, volume 29, number 1, pp. 19-51 March 2003.

Jahna Otterbacher, Gunes Erkan, Dragomir R. Radev. 2009. *Biased LexRank: Passage retrieval using random walks with question-based priors.* Information Processing and Management: an International Journal, v.45 n.1, p.42-54, January, 2009.

Chandra Prabha, Lynn Silipigni Connaway, Lawrence Olszewski, Lillie R. Jenkins. 2007. *What is enough? Satisficing information needs*. Journal of Documentation (January, 63,1).

Ellen Voorhees and Dawn Tice. 2000. *The TREC-8 question answering track evaluation.* In Text Retrieval Conference TREC-8, Gaithersburg, MD.

Kai Wang, Yanming Zhao, and Tat-Seng Chua. 2009. *A syntactic tree matching approach to finding similar*

questions in community-based qa services. In SIGIR, pages 187C194.

Kai Wang and Tat-Seng Chua. 2010. *Exploiting salient patterns for question detection and question retrieval in community-based question answering.* In Proceedings of the 23rd International Conference on Computational Linguistics (COLING '10). Association for Computational Linguistics, Stroudsburg, PA, USA, 1155-1163.

Hu Wu, Yongji Wang, and Xiang Cheng. 2008. *Incremental probabilistic latent semantic analysis for automatic question recommendation.* In RecSys.

Xiaobing Xue, Jiwoon Jeon, W. Bruce Croft. 2008. *Retrieval models for question and answer archives.* In SIGIR'08, pages 475C482. ACM.

Shiqi Zhao, Ming Zhou, and Ting Liu. 2007. *Learning Question Paraphrases for QA from Encarta Logs.* In Proceedings of International Joint Conferences on Artificial Intelligence (IJCAI), pages 1795-1800.

Semi-Supervised Frame-Semantic Parsing for Unknown Predicates

Dipanjan Das and **Noah A. Smith**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{dipanjan,nasmith}@cs.cmu.edu

Abstract

We describe a new approach to disambiguating semantic frames evoked by lexical predicates previously unseen in a lexicon or annotated data. Our approach makes use of large amounts of unlabeled data in a graph-based semi-supervised learning framework. We construct a large graph where vertices correspond to potential predicates and use label propagation to learn possible semantic frames for new ones. The label-propagated graph is used within a frame-semantic parser and, for unknown predicates, results in over 15% absolute improvement in frame identification accuracy and over 13% absolute improvement in full frame-semantic parsing F_1 score on a blind test set, over a state-of-the-art supervised baseline.

1 Introduction

Frame-semantic parsing aims to extract a shallow semantic structure from text, as shown in Figure 1. The FrameNet lexicon (Fillmore et al., 2003) is a rich linguistic resource containing expert knowledge about lexical and predicate-argument semantics. The lexicon suggests an analysis based on the theory of frame semantics (Fillmore, 1982). Recent approaches to frame-semantic parsing have broadly focused on the use of two statistical classifiers corresponding to the aforementioned subtasks: the first one to identify the most suitable semantic frame for a marked lexical predicate (*target*, henceforth) in a sentence, and the second for performing semantic role labeling (SRL) given the frame.

The FrameNet lexicon, its exemplar sentences containing instantiations of semantic frames, and full-text annotations provide supervision for learning frame-semantic parsers. Yet these annotations lack coverage, including only 9,300 annotated target types. Recent papers have tried to address the coverage problem. Johansson and Nugues (2007) used WordNet (Fellbaum, 1998) to expand the list of targets that can evoke frames and trained classifiers to identify the best-suited frame for the newly created targets. In past work, we described an approach where latent variables were used in a probabilistic model to predict frames for unseen targets (Das et al., 2010a).[1] Relatedly, for the argument identification subtask, Matsubayashi et al. (2009) proposed a technique for generalization of semantic roles to overcome data sparseness. Unseen targets continue to present a major obstacle to domain-general semantic analysis.

In this paper, we address the problem of idenfifying the semantic frames for targets unseen either in FrameNet (including the exemplar sentences) or the collection of full-text annotations released along with the lexicon. Using a standard model for the argument identification stage (Das et al., 2010a), our proposed method improves overall frame-semantic parsing, especially for unseen targets. To better handle these unseen targets, we adopt a graph-based semi-supervised learning stategy (§4). We construct a large graph over potential targets, most of which

[1] Notwithstanding state-of-the-art results, that approach was only able to identify the correct frame for 1.9% of unseen targets in the test data available at that time. That system achieves about 23% on the test set used in this paper.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1435–1444,
Portland, Oregon, June 19-24, 2011.

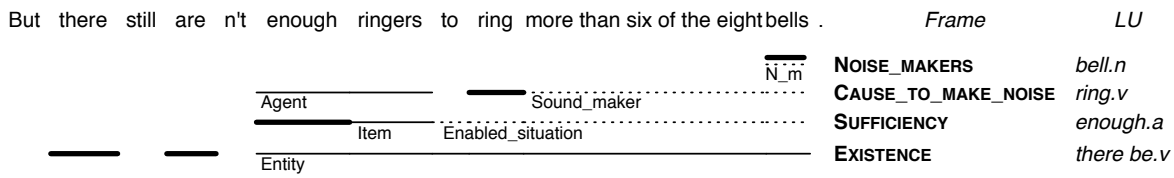


Figure 1: An example sentence from the PropBank section of the full-text annotations released as part of FrameNet 1.5. Each row under the sentence correponds to a semantic frame and its set of corresponding arguments. Thick lines indicate targets that evoke frames; thin solid/dotted lines with labels indicate arguments. N_m under "bells" is short for the Noise_maker role of the NOISE_MAKERS frame.

are drawn from unannotated data, and a fraction of which come from seen FrameNet annotations. Next, we perform label propagation on the graph, which is initialized by frame distributions over the *seen* targets. The resulting smoothed graph consists of posterior distributions over semantic frames for each target in the graph, thus increasing coverage. These distributions are then evaluated within a frame-semantic parser (§5). Considering unseen targets in test data (although few because the test data is also drawn from the training domain), significant absolute improvements of 15.7% and 13.7% are observed for frame identification and full frame-semantic parsing, respectively, indicating improved coverage for hitherto unobserved predicates (§6).

2 Background

Before going into the details of our model, we provide some background on two topics relevant to this paper: frame-semantic parsing and graph-based learning applied to natural language tasks.

2.1 Frame-semantic Parsing

Gildea and Jurafsky (2002) pioneered SRL, and since then there has been much applied research on predicate-argument semantics. Early work on frame-semantic role labeling made use of the exemplar sentences in the FrameNet corpus, each of which is annotated for a single frame and its arguments (Thompson et al., 2003; Fleischman et al., 2003; Shi and Mihalcea, 2004; Erk and Padó, 2006, *inter alia*). Most of this work was done on an older, smaller version of FrameNet. Recently, since the release of full-text annotations in SemEval'07 (Baker et al., 2007), there has been work on identifying multiple frames and their corresponding sets of arguments in a sentence. The LTH system of Johansson and Nugues (2007) performed the best in the SemEval'07 shared task on frame-semantic parsing. Our probabilistic frame-semantic parser outperforms LTH on that task and dataset (Das et al., 2010a). The current paper builds on those probabilistic models to improve coverage on unseen predicates.[2]

Expert resources have limited coverage, and FrameNet is no exception. Automatic induction of semantic resources has been a major effort in recent years (Snow et al., 2006; Ponzetto and Strube, 2007, *inter alia*). In the domain of frame semantics, previous work has sought to extend the coverage of FrameNet by exploiting resources like VerbNet, WordNet, or Wikipedia (Shi and Mihalcea, 2005; Giuglea and Moschitti, 2006; Pennacchiotti et al., 2008; Tonelli and Giuliano, 2009), and projecting entries and annotations within and across languages (Boas, 2002; Fung and Chen, 2004; Padó and Lapata, 2005). Although these approaches have increased coverage to various degrees, they rely on other lexicons and resources created by experts. Fürstenau and Lapata (2009) proposed the use of unlabeled data to improve coverage, but their work was limited to verbs. Bejan (2009) used self-training to improve frame identification and reported improvements, but did not explicitly model unknown targets. In contrast, we use statistics gathered from large volumes of unlabeled data to improve the coverage of a frame-semantic parser on several syntactic categories, in a novel framework that makes use of graph-based semi-supervised learning.

[2]SEMAFOR, the system presented by Das et al. (2010a) is publicly available at `http://www.ark.cs.cmu.edu/SEMAFOR` and has been extended in this work.

2.2 Graph-based Semi-Supervised Learning

In graph-based semi-supervised learning, one constructs a graph whose vertices are labeled and unlabeled examples. Weighted edges in the graph, connecting pairs of examples/vertices, encode the degree to which they are expected to have the same label (Zhu et al., 2003). Variants of label propagation are used to transfer labels from the labeled to the unlabeled examples. There are several instances of the use of graph-based methods for natural language tasks. Most relevant to our work an approach to word-sense disambiguation due to Niu et al. (2005). Their formulation was transductive, so that the test data was part of the constructed graph, and they did not consider predicate-argument analysis. In contrast, we make use of the smoothed graph during inference in a probabilistic setting, in turn using it for the full frame-semantic parsing task. Recently, Subramanya et al. (2010) proposed the use of a graph over substructures of an underlying sequence model, and used a smoothed graph for domain adaptation of part-of-speech taggers. Subramanya et al.'s model was extended by Das and Petrov (2011) to induce part-of-speech dictionaries for unsupervised learning of taggers. Our semi-supervised learning setting is similar to these two lines of work and, like them, we use the graph to arrive at better final structures, in an inductive setting (i.e., where a parametric model is learned and then separately applied to test data, following most NLP research).

3 Approach Overview

Our overall approach to handling unobserved targets consists of four distinct stages. Before going into the details of each stage individually, we provide their overview here:

Graph Construction: A graph consisting of vertices corresponding to targets is constructed using a combination of frame similarity (for observed targets) and distributional similarity as edge weights. This stage also determines a fixed set of nearest neighbors for each vertex in the graph.

Label Propagation: The observed targets (a small subset of the vertices) are initialized with empirical frame distributions extracted from FrameNet annotations. Label propagation results in a distribution of frames for each vertex in the graph.

Supervised Learning: Frame identification and argument identification models are trained following Das et al. (2010a). The graph is used to define the set of candidate frames for unseen targets.

Parsing: The frame identification model of Das et al. disambiguated among only those frames associated with a seen target in the annotated data. For an unseen target, all frames in the FrameNet lexicon were considered (a large number). The current work replaces that strategy, considering only the top M frames in the distribution produced by label propagation. This strategy results in large improvements in frame identification for the unseen targets and makes inference much faster. Argument identification is done exactly like Das et al. (2010a).

4 Semi-Supervised Learning

We perform semi-supervised learning by constructing a graph of vertices representing a large number of targets, and learn frame distributions for those which were not observed in FrameNet annotations.

4.1 Graph Construction

We construct a graph with targets as vertices. For us, each target corresponds to a lemmatized word or phrase appended with a coarse POS tag, and it resembles the *lexical units* in the FrameNet lexicon. For example, two targets corresponding to the same lemma would look like *boast*.N and *boast*.V. Here, the first target is a noun, while the second is a verb. An example multiword target is *chemical weapon*.N.

We use two resources for graph construction. First, we take all the words and phrases present in the dependency-based thesaurus constructed using syntactic cooccurrence statistics (Lin, 1998).[3] To construct this resource, a corpus containing 64 million words was parsed with a fast dependency parser (Lin, 1993; Lin, 1994), and syntactic contexts were used to find similar lexical items for a given word

[3]This resource is available at `http://webdocs.cs.ualberta.ca/~lindek/Downloads/sim.tgz`

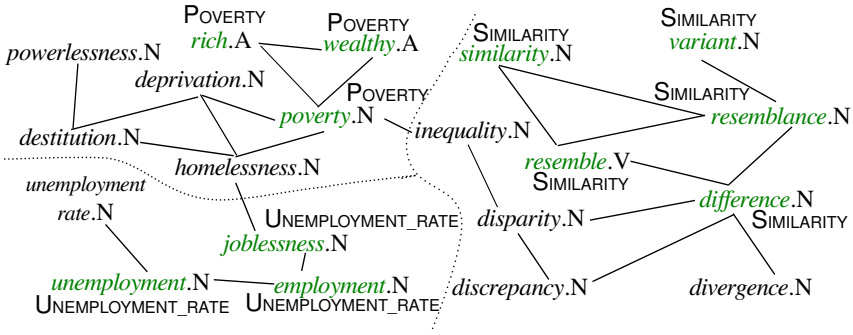


Figure 2: Excerpt from a graph over targets. Green targets are observed in the FrameNet data. Above/below them are shown the most frequently observed frame that these targets evoke. The black targets are unobserved and label propagation produces a distribution over most likely frames that they could evoke.

or phrase. Lin separately treated nouns, verbs and adjectives/adverbs and the thesaurus contains three parts for each of these categories. For each item in the thesaurus, 200 nearest neighbors are listed with a symmetric similarity score between 0 and 1. We processed this thesaurus in two ways: first, we lowercased and lemmatized each word/phrase and merged entries which shared the same lemma; second, we separated the adjectives and adverbs into two lists from Lin's original list by scanning a POS-tagged version of the Gigaword corpus (Graff, 2003) and categorizing each item into an adjective or an adverb depending on which category the item associated with more often in the data. The second step was necessary because FrameNet treats adjectives and adverbs separately. At the end of this processing step, we were left with 61,702 units—approximately six times more than the targets found in FrameNet annotations—each labeled with one of 4 coarse tags. We considered only the top 20 most similar targets for each target, and noted Lin's similarity between two targets t and u, which we call $\mathsf{sim}_{DL}(t, u)$.

The second component of graph construction comes from FrameNet itself. We scanned the exemplar sentences in FrameNet 1.5[4] and the training section of the full-text annotations that we use to train the probabilistic frame parser (see §6.1), and gathered a distribution over frames for each target. For a pair of targets t and u, we measured the Euclidean distance[5] between their frame distributions. This distance was next converted to a similarity score, namely, $\mathsf{sim}_{FN}(t, u)$ between 0 and 1 by subtracting each one from the maximum distance found in the whole data, followed by normalization. Like $\mathsf{sim}_{DL}(t, u)$, this score is symmetric. This resulted in 9,263 targets, and again for each, we considered the 20 most similar targets. Finally, the overall similarity between two given targets t and u was computed as:

$$\mathsf{sim}(t, u) = \alpha \cdot \mathsf{sim}_{FN}(t, u) + (1 - \alpha) \cdot \mathsf{sim}_{DL}(t, u)$$

Note that this score is symmetric because its two components are symmetric. The intuition behind taking a linear combination of the two types of similarity functions is as follows. We hope that distributionally similar targets would have the same semantic frames because ideally, lexical units evoking the same set of frames appear in similar syntactic contexts. We would also like to involve the annotated data in graph construction so that it can eliminate some noise in the automatically constructed thesaurus.[6] Let $\mathcal{K}(t)$ denote the K most similar targets to target t, under the score sim. We link vertices t and u in the graph with edge weight w_{tu}, defined as:

$$w_{tu} = \begin{cases} \mathsf{sim}(t, u) & \text{if } t \in \mathcal{K}(u) \text{ or } u \in \mathcal{K}(t) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The hyperparameters α and K are tuned by cross-validation (§6.3).

4.2 Label Propagation

First, we softly label those vertices of the constructed graph for which frame distributions are available from the FrameNet data (the same distributions that are used to compute sim_{FN}). Thus, initially, a small fraction of the vertices in the graph

[4] http://framenet.icsi.berkeley.edu

[5] This could have been replaced by an entropic distance metric like KL- or JS-divergence, but we leave that exploration to future work.

[6] In future work, one might consider *learning* a similarity metric from the annotated data, so as to exactly suit the frame identification task.

have soft frame labels on them. Figure 2 shows an excerpt from a constructed graph. For simplicity, only the most probable frames under the empirical distribution for the observed targets are shown; we actually label each vertex with the full empirical distribution over frames for the corresponding observed target in the data. The dotted lines demarcate parts of the graph that associate with different frames. Label propagation helps propagate the initial soft labels throughout the graph. To this end, we use a variant of the quadratic cost criterion of Bengio et al. (2006), also used by Subramanya et al. (2010) and Das and Petrov (2011).[7]

Let V denote the set of all vertices in the graph, $V_l \subset V$ be the set of known targets and $\mathcal{F}$ denote the set of all frames. Let $\mathcal{N}(t)$ denote the set of neighbors of vertex $t \in V$. Let $\mathrm{q} = \{\mathrm{q}_1, \mathrm{q}_2, \ldots, \mathrm{q}_{|V|}\}$ be the set of frame distributions, one per vertex. For each known target $t \in V_l$, we have an initial frame distribution r_t. For every edge in the graph, weights are defined as in Eq. 1. We find q by solving:

$$\begin{aligned} \arg\min_{\mathrm{q}} \quad & \textstyle\sum_{t \in V_l} \|\mathrm{r}_t - \mathrm{q}_t\|^2 \\ & + \mu \textstyle\sum_{t \in V, u \in \mathcal{N}(t)} w_{tu} \|\mathrm{q}_t - \mathrm{q}_u\|^2 \\ & + \nu \textstyle\sum_{t \in V} \|\mathrm{q}_t - \frac{1}{|\mathcal{F}|}\|^2 \\ \text{s.t.} \quad & \forall t \in V, \ \textstyle\sum_{f \in \mathcal{F}} \mathrm{q}_t(f) = 1 \\ & \forall t \in V, f \in \mathcal{F}, \ \mathrm{q}_t(f) \geq 0 \end{aligned} \tag{2}$$

We use a squared loss to penalize various pairs of distributions over frames: $\|\mathrm{a} - \mathrm{b}\|^2 = \sum_{f \in \mathcal{F}} (\mathrm{a}(f) - \mathrm{b}(f))^2$. The first term in Eq. 2 requires that, for known targets, we stay close to the initial frame distributions. The second term is the graph smoothness regularizer, which encourages the distributions of similar nodes (large w_{tu}) to be similar. The final term is a regularizer encouraging all distributions to be uniform to the extent allowed by the first two terms. (If an unlabeled vertex does not have a path to any labeled vertex, this term ensures that its converged marginal will be uniform over all frames.) μ and ν are hyperparameters whose choice we discuss in §6.3.

Note that Eq. 2 is convex in q. While it is possible to derive a closed form solution for this objective function, it would require the inversion of a $|V| \times |V|$ matrix. Hence, like Subramanya et al. (2010), we employ an iterative method with updates defined as:

$$\begin{aligned} \gamma_t(f) \quad &\leftarrow \quad \mathrm{r}_t(f)\mathbf{1}\{t \in V_l\} \\ &\quad + \ \mu \sum_{u \in \mathcal{N}(t)} w_{tu} \mathrm{q}_u^{(m-1)}(f) + \frac{\nu}{|\mathcal{F}|} \end{aligned} \tag{3}$$

$$\kappa_t \quad \leftarrow \quad \mathbf{1}\{t \in V_l\} + \nu + \mu \sum_{u \in \mathcal{N}(t)} w_{tu} \tag{4}$$

$$\mathrm{q}_t^{(m)}(f) \quad \leftarrow \quad \gamma_t(f)/\kappa_t \tag{5}$$

Here, $\mathbf{1}\{\cdot\}$ is an indicator function. The iterative procedure starts with a uniform distribution for each $\mathrm{q}_t^{(0)}$. For all our experiments, we run 10 iterations of the updates. The final distribution of frames for a target t is denoted by q_t^*.

5 Learning and Inference for Frame-Semantic Parsing

In this section, we briefly review learning and inference techniques used in the frame-semantic parser, which are largely similar to Das et al. (2010a), except the handling of unknown targets. Note that in all our experiments, we assume that the targets are marked in a given sentence of which we want to extract a frame-semantic analysis. Therefore, unlike the systems presented in SemEval'07, we do not define a target identification module.

5.1 Frame Identification

For a given sentence $\mathbf{x}$ with frame-evoking targets $\mathbf{t}$, let t_i denote the ith target (a word sequence). We seek a list $\mathbf{f} = \langle f_1, \ldots, f_m \rangle$ of frames, one per target. Let $\mathcal{L}$ be the set of targets found in the FrameNet annotations. Let $\mathcal{L}_f \subseteq \mathcal{L}$ be the subset of these targets annotated as evoking a particular frame f.

The set of candidate frames $\mathcal{F}_i$ for t_i is defined to include every frame f such that $t_i \in \mathcal{L}_f$. If $t_i \notin \mathcal{L}$ (in other words, t_i is unseen), then Das et al. (2010a) considered all frames $\mathcal{F}$ in FrameNet as candidates. Instead, in our work, we check whether $t_i \in V$, where V are the vertices of the constructed graph, and set:

$$\mathcal{F}_i = \{f : f \in M\text{-best frames under } \mathrm{q}_{t_i}^*\} \tag{6}$$

The integer M is set using cross-validation (§6.3). If $t_i \notin V$, then all frames $\mathcal{F}$ are considered as $\mathcal{F}_i$.

[7] Instead of a quadratic cost, an entropic distance measure could have been used, e.g., KL-divergence, considered by Subramanya and Bilmes (2009). We do not explore that direction in the current paper.

The frame prediction rule uses a probabilistic model over frames for a target:

$$f_i \leftarrow \arg\max_{f \in \mathcal{F}_i} \sum_{\ell \in \mathcal{L}_f} p(f, \ell \mid t_i, \mathbf{x}) \quad (7)$$

Note that a latent variable $\ell \in \mathcal{L}_f$ is used, which is marginalized out. Broadly, lexical semantic relationships between the "prototype" variable ℓ (belonging to the set of seen targets for a frame f) and the target t_i are used as features for frame identification, but since ℓ is unobserved, it is summed out both during inference and training. A conditional log-linear model is used to model this probability: for $f \in \mathcal{F}_i$ and $\ell \in \mathcal{L}_f$, $p_{\boldsymbol{\theta}}(f, \ell \mid t_i, \mathbf{x}) =$

$$\frac{\exp \boldsymbol{\theta}^\top \mathbf{g}(f, \ell, t_i, \mathbf{x})}{\sum_{f' \in \mathcal{F}_i} \sum_{\ell' \in \mathcal{L}_{f'}} \exp \boldsymbol{\theta}^\top \mathbf{g}(f', \ell', t_i, \mathbf{x})} \quad (8)$$

where $\boldsymbol{\theta}$ are the model weights, and $\mathbf{g}$ is a vector-valued feature function. This discriminative formulation is very flexible, allowing for a variety of (possibly overlapping) features; e.g., a feature might relate a frame f to a prototype ℓ, represent a lexical-semantic relationship between ℓ and t_i, or encode part of the syntax of the sentence (Das et al., 2010b).

Given some training data, which is of the form $\left\langle \langle \mathbf{x}^{(j)}, \mathbf{t}^{(j)}, \mathbf{f}^{(j)}, \mathcal{A}^{(j)} \rangle \right\rangle_{j=1}^{N}$ (where N is the number of sentences in the data and $\mathcal{A}$ is the set of argument in a sentence), we discriminatively train the frame identification model by maximizing the following log-likelihood:[8]

$$\max_{\boldsymbol{\theta}} \sum_{j=1}^{N} \sum_{i=1}^{m_j} \log \sum_{\ell \in \mathcal{L}_{f_i^{(j)}}} p_{\boldsymbol{\theta}}(f_i^{(j)}, \ell \mid t_i^{(j)}, \mathbf{x}^{(j)}) \quad (9)$$

This non-convex objective function is locally optimized using a distributed implementation of L-BFGS (Liu and Nocedal, 1989).[9]

5.2 Argument Identification

Given a sentence $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$, the set of targets $\mathbf{t} = \langle t_1, \ldots, t_m \rangle$, and a list of evoked frames $\mathbf{f} = \langle f_1, \ldots, f_m \rangle$ corresponding to each target, argument identification or SRL is the task of choosing which of each f_i's roles are filled, and by which parts of $\mathbf{x}$. We directly adopt the model of Das et al. (2010a) for the argument identification stage and briefly describe it here.

Let $\mathcal{R}_{f_i} = \{r_1, \ldots, r_{|\mathcal{R}_{f_i}|}\}$ denote frame f_i's **roles** observed in FrameNet annotations. A set $\mathcal{S}$ of spans that are candidates for filling any role $r \in \mathcal{R}_{f_i}$ are identified in the sentence. In principle, $\mathcal{S}$ could contain any subsequence of $\mathbf{x}$, but we consider only the set of contiguous spans that (a) contain a single word or (b) comprise a valid subtree of a word and all its descendants in a dependency parse. The empty span is also included in $\mathcal{S}$, since some roles are not explicitly filled. During training, if an argument is not a valid subtree of the dependency parse (this happens due to parse errors), we add its span to $\mathcal{S}$. Let $\mathcal{A}_i$ denote the mapping of roles in $\mathcal{R}_{f_i}$ to spans in $\mathcal{S}$. The model makes a prediction for each $\mathcal{A}_i(r_k)$ (for all roles $r_k \in \mathcal{R}_{f_i}$):

$$\mathcal{A}_i(r_k) \leftarrow \arg\max_{s \in \mathcal{S}} p(s \mid r_k, f_i, t_i, \mathbf{x}) \quad (10)$$

A conditional log-linear model over spans for each role of each evoked frame is defined as:

$$p_{\boldsymbol{\psi}}(\mathcal{A}_i(r_k) = s \mid f_i, t_i, \mathbf{x}) = \frac{\exp \boldsymbol{\psi}^\top \mathbf{h}(s, r_k, f_i, t_i, \mathbf{x})}{\sum_{s' \in \mathcal{S}} \exp \boldsymbol{\psi}^\top \mathbf{h}(s', r_k, f_i, t_i, \mathbf{x})} \quad (11)$$

This model is trained by optimizing:

$$\max_{\boldsymbol{\psi}} \sum_{j=1}^{N} \sum_{i=1}^{m_j} \sum_{k=1}^{|\mathcal{R}_{f_i^{(j)}}|} \log p_{\boldsymbol{\psi}}(\mathcal{A}_i^{(j)}(r_k) \mid f_i^{(j)}, t_i^{(j)}, \mathbf{x}^{(j)})$$

This objective function is convex, and we globally optimize it using the distributed implementation of L-BFGS. We regularize by including $-\frac{1}{10}\|\boldsymbol{\psi}\|_2^2$ in the objective (the strength is not tuned). Naïve prediction of roles using Equation 10 may result in overlap among arguments filling different roles of a frame, since the argument identification model fills each role independently of the others. We want to enforce the constraint that two roles of a single frame cannot be filled by overlapping spans. Hence, illegal overlap is disallowed using a 10,000-hypothesis beam search.

[8] We found no benefit from using an L_2 regularizer.

[9] While training, in the partition function of the log-linear model, all frames $\mathcal{F}$ in FrameNet are summed up for a target t_i instead of only $\mathcal{F}_i$ (as in Eq. 8), to learn interactions between the latent variables and different sentential contexts.

Model	Unknown Targets		All Targets	
	Exact Match	**Partial Match**	**Exact Match**	**Partial Match**
SEMAFOR	23.08	46.62	82.97	90.51
Self-training	18.88	42.67	82.45	90.19
LinGraph	**36.36**	**59.47**	**83.40**	**90.93**
FullGraph	**39.86**	**62.35***	**83.51**	**91.02***

Table 1: Frame identification results in percentage accuracy on 4,458 test targets. Bold scores indicate significant improvements relative to SEMAFOR and (*) denotes significant improvements over LinGraph ($p < 0.05$).

6 Experiments and Results

Before presenting our experiments and results, we will describe the datasets used in our experiments, and the various baseline models considered.

6.1 Data

We make use of the FrameNet 1.5 lexicon released in 2010. This lexicon is a superset of previous versions of FrameNet. It contains 154,607 exemplar sentences with one marked target and frame-role annotations. 78 documents with full-text annotations with multiple frames per sentence were also released (a superset of the SemEval'07 dataset). We randomly selected 55 of these documents for training and treated the 23 remaining ones as our test set. After scanning the exemplar sentences and the training data, we arrived at a set of 877 frames, 1,068 roles,[10] and 9,263 targets. Our training split of the full-text annotations contained 3,256 sentences with 19,582 frame annotatations with corresponding roles, while the test set contained 2,420 sentences with 4,458 annotations (the test set contained fewer annotated targets per sentence). We also divide the 55 training documents into 5 parts for cross-validation (see §6.3). The raw sentences in all the training and test documents were preprocessed using MXPOST (Ratnaparkhi, 1996) and the MST dependency parser (McDonald et al., 2005) following Das et al. (2010a). In this work we assume the frame-evoking targets have been correctly identified in training and test data.

[10]Note that the number of listed roles in the lexicon is nearly 9,000, but their number in actual annotations is a lot fewer.

6.2 Baselines

We compare our model with three baselines. The first baseline is the purely supervised model of Das et al. (2010a) trained on the training split of 55 documents. Note that this is the strongest baseline available for this task;[11] we refer to this model as "SEMAFOR."

The second baseline is a semi-supervised self-trained system, where we used SEMAFOR to label 70,000 sentences from the Gigaword corpus with frame-semantic parses. For finding targets in a raw sentence, we used a relaxed target identification scheme, where we marked every target seen in the lexicon and all other words which were not prepositions, particles, proper nouns, foreign words and Wh-words as potential frame evoking units. This was done so as to find unseen targets and get frame annotations with SEMAFOR on them. We appended these automatic annotations to the training data, resulting in 711,401 frame annotations, more than 36 times the supervised data. These data were next used to train a frame identification model (§5.1).[12] This setup is very similar to Bejan (2009) who used self-training to improve frame identification. We refer to this model as "Self-training."

The third baseline uses a graph constructed only with Lin's thesaurus, without using supervised data. In other words, we followed the same scheme as in §4.1 but with the hyperparameter $\alpha = 0$. Next, label propagation was run on this graph (and hyperparameters tuned using cross validation). The posterior distribution of frames over targets was next used for frame identification (Eq. 6-7), with SEMAFOR as the trained model. This model, which is very similar to our full model, is referred to as "LinGraph."

"FullGraph" refers to our full system.

6.3 Experimental Setup

We used five-fold cross-validation to tune the hyperparameters α, K, μ, and M in our model. The

[11]We do not compare our model with other systems, e.g. the ones submitted to SemEval'07 shared task, because SEMAFOR outperforms them significantly (Das et al., 2010a) on the previous version of the data. Moreover, we trained our models on the new FrameNet 1.5 data, and training code for the SemEval'07 systems was not readily available.

[12]Note that we only self-train the frame identification model and not the argument identification model, which is fixed throughout.

Model	UNKNOWN TARGETS						ALL TARGETS					
	Exact Match			Partial Match			Exact Match			Partial Match		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
SEMAFOR	19.59	16.48	17.90	33.03	27.80	30.19	66.15	61.64	63.82	70.68	65.86	68.18
Self-training	15.44	13.00	14.11	29.08	24.47	26.58	65.78	61.30	63.46	70.39	65.59	67.90
LinGraph	**29.74**	**24.88**	**27.09**	**44.08**	**36.88**	**40.16**	**66.43**	**61.89**	**64.08**	**70.97**	**66.13**	**68.46**
FullGraph	**35.27***	**28.84***	**31.74***	**48.81***	**39.91***	**43.92***	**66.59***	**62.01***	**64.22***	**71.11***	**66.22***	**68.58***

Table 2: Full frame-semantic parsing precision, recall and F_1 score on 2,420 test sentences. Bold scores indicate significant improvements relative to SEMAFOR and (*) denotes significant improvements over LinGraph ($p < 0.05$).

uniform regularization hyperparameter ν for graph construction was set to 10^{-6} and not tuned. For each cross-validation split, four folds were used to train a frame identification model, construct a graph, run label propagation and then the model was tested on the fifth fold. This was done for all hyperparameter settings, which were $\alpha \in \{0.2, 0.5, 0.8\}$, $K \in \{5, 10, 15, 20\}$, $\mu \in \{0.01, 0.1, 0.3, 0.5, 1.0\}$ and $M \in \{2, 3, 5, 10\}$. The joint setting which performed the best across five-folds was $\alpha = 0.2, K = 10, \mu = 1.0, M = 2$. Similar tuning was also done for the baseline LinGraph, where α was set to 0, and rest of the hyperparameters were tuned (the selected hyperparameters were $K = 10$, $\mu = 0.1$ and $M = 2$). With the chosen set of hyperparameters, the test set was used to measure final performance.

The standard evaluation script from the SemEval'07 task calculates precision, recall, and F_1-score for frames and arguments; it also provides a score that gives partial credit for hypothesizing a frame *related* to the correct one in the FrameNet lexicon. We present precision, recall, and F_1-measure microaveraged across the test documents, report labels-only matching scores (spans must match exactly), and do not use named entity labels. This evaluation scheme follows Das et al. (2010a). Statistical significance is measured using a reimplementation of Dan Bikel's parsing evaluation comparator.[13]

6.4 Results

Tables 1 and 2 present results for frame identification and full frame-semantic parsing respectively. They also separately tabulate the results achieved for unknown targets. Our full model, denoted by "FullGraph," outperforms all the baselines for both tasks. Note that the Self-training model even falls short of the supervised baseline SEMAFOR, unlike what was observed by Bejan (2009) for the frame identification task. The model using a graph constructed solely from the thesaurus (LinGraph) outperforms both the supervised and the self-training baselines for all tasks, but falls short of the graph constructed using the similarity metric that is a linear combination of distributional similarity and supervised frame similarity. This indicates that a graph constructed with some knowledge of the supervised data is more powerful.

For unknown targets, the gains of our approach are impressive: 15.7% absolute accuracy improvement over SEMAFOR for frame identification, and 13.7% absolute F_1 improvement over SEMAFOR for full frame-semantic parsing (both significant). When all the test targets are considered, the gains are still significant, resulting in 5.4% relative error reduction over SEMAFOR for frame identification, and 1.3% relative error reduction over SEMAFOR for full-frame semantic parsing.

Although these improvements may seem modest, this is because only 3.2% of the test set targets are unseen in training. We expect that further gains would be realized in different text domains, where FrameNet coverage is presumably weaker than in news data. A semi-supervised strategy like ours is attractive in such a setting, and future work might explore such an application.

Our approach also makes decoding much faster. For the unknown component of the test set, SEMAFOR takes a total 111 seconds to find the best set of frames, while the FullGraph model takes only 19 seconds to do so, thus bringing disambiguation time down by a factor of nearly 6. This is because our model now disambiguates between only $M = 2$ frames instead of the full set of 877 frames in FrameNet. For the full test set too, the speedup

[13] http://www.cis.upenn.edu/~dbikel/software.html#comparator

$t =$ *discrepancy*.N		$t =$ *contribution*.N		$t =$ *print*.V		$t =$ *mislead*.V	
f	$\mathrm{q}_t^*(f)$	f	$\mathrm{q}_t^*(f)$	f	$\mathrm{q}_t^*(f)$	f	$\mathrm{q}_t^*(f)$
∗SIMILARITY	0.076	∗GIVING	0.167	∗TEXT_CREATION	0.081	EXPERIENCER_OBJ	0.152
NATURAL_FEATURES	0.066	MONEY	0.046	SENDING	0.054	∗PREVARICATION	0.130
PREVARICATION	0.012	COMMITMENT	0.046	DISPERSAL	0.054	MANIPULATE_INTO_DOING	0.046
QUARRELING	0.007	ASSISTANCE	0.040	READING	0.042	COMPLIANCE	0.041
DUPLICATION	0.007	EARNINGS_AND_LOSSES	0.024	STATEMENT	0.028	EVIDENCE	0.038

Table 3: Top 5 frames according to the graph posterior distribution $\mathrm{q}_t^*(f)$ for four targets: *discrepancy*.N, *contribution*.N, *print*.V and *mislead*.V. None of these targets were present in the supervised FrameNet data. $*$ marks the correct frame, according to the test data. EXPERIENCER_OBJ is described in FrameNet as "Some phenomenon (the Stimulus) provokes a particular emotion in an Experiencer."

is noticeable, as SEMAFOR takes 131 seconds for frame identification, while the FullGraph model only takes 39 seconds.

6.5 Discussion

The following is an example from our test set showing SEMAFOR's output (for one target):

> REASON
> Discrepancies between North Korean declarations and IAEA inspection findings Action indicate that North Korea might have reprocessed enough plutonium for one or two nuclear weapons.
> *discrepancy*.N

Note that the model identifies an incorrect frame REASON for the target *discrepancy*.N, in turn identifying the wrong semantic role Action for the underlined argument. On the other hand, the FullGraph model exactly identifies the right semantic frame, SIMILARITY, as well as the correct role, Entities. This improvement can be easily explained. The excerpt from our constructed graph in Figure 2 shows the same target *discrepancy*.N in black, conveying that it did not belong to the supervised data. However, it is connected to the target *difference*.N drawn from annotated data, which evokes the frame SIMILARITY. Thus, after label propagation, we expect the frame SIMILARITY to receive high probability for the target *discrepancy*.N.

Table 3 shows the top 5 frames that are assigned the highest posterior probabilities in the distribution q_t^* for four hand-selected test targets absent in supervised data, including *discrepancy*.N. For all of them, the FullGraph model identifies the correct frames for all four words in the test data by ranking these frames in the top $M = 2$. LinGraph also gets all four correct, Self-training only gets *print*.V/TEXT_CREATION, and SEMAFOR gets none.

Across unknown targets, on average the $M = 2$ most common frames in the posterior distribution q_t^* found by FullGraph have $\mathrm{q}_t^{(*)}(f) = \frac{7}{877}$, or seven times the average across all frames. This suggests that the graph propagation method is confident only in predicting the top few frames out of the whole possible set. Moreover, the automatically selected number of frames to extract per unknown target, $M = 2$, suggests that only a few meaningful frames were assigned to unknown predicates. This matches the nature of FrameNet data, where the average frame ambiguity for a target type is 1.20.

7 Conclusion

We have presented a semi-supervised strategy to improve the coverage of a frame-semantic parsing model. We showed that graph-based label propagation and resulting smoothed frame distributions over unseen targets significantly improved the coverage of a state-of-the-art semantic frame disambiguation model to previously unseen predicates, also improving the quality of full frame-semantic parses. The improved parser is available at `http://www.ark.cs.cmu.edu/SEMAFOR`.

Acknowledgments

We are grateful to Amarnag Subramanya for helpful discussions. We also thank Slav Petrov, Nathan Schneider, and the three anonymous reviewers for valuable comments. This research was supported by NSF grants IIS-0844507, IIS-0915187 and TeraGrid resources provided by the Pittsburgh Supercomputing Center under NSF grant number TG-DBS110003.

References

C. Baker, M. Ellsworth, and K. Erk. 2007. SemEval-2007 Task 19: frame semantic structure extraction. In *Proc. of SemEval*.

C. A. Bejan. 2009. *Learning Event Structures From Text*. Ph.D. thesis, The University of Texas at Dallas.

Y. Bengio, O. Delalleau, and N. Le Roux. 2006. Label propagation and quadratic criterion. In *Semi-Supervised Learning*. MIT Press.

H. C. Boas. 2002. Bilingual FrameNet dictionaries for machine translation. In *Proc. of LREC*.

D. Das and S. Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proc. of ACL-HLT*.

D. Das, N. Schneider, D. Chen, and N. A. Smith. 2010a. Probabilistic frame-semantic parsing. In *Proc. of NAACL-HLT*.

D. Das, N. Schneider, D. Chen, and N. A. Smith. 2010b. SEMAFOR 1.0: A probabilistic frame-semantic parser. Technical Report CMU-LTI-10-001, Carnegie Mellon University.

K. Erk and S. Padó. 2006. Shalmaneser - a toolchain for shallow semantic parsing. In *Proc. of LREC*.

C. Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press, Cambridge, MA.

C. J. Fillmore, C. R. Johnson, and M. R.L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16(3).

C. J. Fillmore. 1982. Frame semantics. In *Linguistics in the Morning Calm*, pages 111–137. Hanshin Publishing Co., Seoul, South Korea.

M. Fleischman, N. Kwon, and E. Hovy. 2003. Maximum entropy models for FrameNet classification. In *Proc. of EMNLP*.

P. Fung and B. Chen. 2004. BiFrameNet: bilingual frame semantics resource construction by cross-lingual induction. In *Proc. of COLING*.

H. Fürstenau and M. Lapata. 2009. Semi-supervised semantic role labeling. In *Proc. of EACL*.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3).

A.-M. Giuglea and A. Moschitti. 2006. Shallow semantic parsing based on FrameNet, VerbNet and PropBank. In *Proc. of ECAI 2006*.

D. Graff. 2003. English Gigaword. Linguistic Data Consortium.

R. Johansson and P. Nugues. 2007. LTH: semantic structure extraction using nonprojective dependency trees. In *Proc. of SemEval*.

D. Lin. 1993. Principle-based parsing without overgeneration. In *Proc. of ACL*.

D. Lin. 1994. Principar–an efficient, broadcoverage, principle-based parser. In *Proc. of COLING*.

D. Lin. 1998. Automatic retrieval and clustering of similar words. In *Proc. of COLING-ACL*.

D. C. Liu and J. Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Math. Programming*, 45(3).

Y. Matsubayashi, N. Okazaki, and J. Tsujii. 2009. A comparative study on generalization of semantic roles in FrameNet. In *Proc. of ACL-IJCNLP*.

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

Z.-Y. Niu, D.-H. Ji, and C. L. Tan. 2005. Word sense disambiguation using label propagation based semi-supervised learning. In *Proc. of ACL*.

S. Padó and M. Lapata. 2005. Cross-linguistic projection of role-semantic information. In *Proc. of HLT-EMNLP*.

M. Pennacchiotti, D. De Cao, R. Basili, D. Croce, and M. Roth. 2008. Automatic induction of FrameNet lexical units. In *Proc. of EMNLP*.

S. P. Ponzetto and M. Strube. 2007. Deriving a large scale taxonomy from wikipedia. In *Proc. of AAAI*.

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proc. of EMNLP*.

L. Shi and R. Mihalcea. 2004. An algorithm for open text semantic parsing. In *Proc. of Workshop on Robust Methods in Analysis of Natural Language Data*.

L. Shi and R. Mihalcea. 2005. Putting pieces together: combining FrameNet, VerbNet and WordNet for robust semantic parsing. In *Computational Linguistics and Intelligent Text Processing: Proc. of CICLing 2005*. Springer-Verlag.

R. Snow, D. Jurafsky, and A. Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proc. of COLING-ACL*.

A. Subramanya and J. A. Bilmes. 2009. Entropic graph regularization in non-parametric semi-supervised classification. In *Proc. of NIPS*.

A. Subramanya, S. Petrov, and F. Pereira. 2010. Efficient Graph-based Semi-Supervised Learning of Structured Tagging Models. In *Proc. of EMNLP*.

C. A. Thompson, R. Levy, and C. D. Manning. 2003. A generative model for semantic role labeling. In *Proc. of ECML*.

S. Tonelli and C. Giuliano. 2009. Wikipedia as frame information repository. In *Proc. of EMNLP*.

X. Zhu, Z. Ghahramani, and J. D. Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. of ICML*.

A Bayesian Model for Unsupervised Semantic Parsing

Ivan Titov
Saarland University
Saarbruecken, Germany
titov@mmci.uni-saarland.de

Alexandre Klementiev
Johns Hopkins University
Baltimore, MD, USA
aklement@jhu.edu

Abstract

We propose a non-parametric Bayesian model for unsupervised semantic parsing. Following Poon and Domingos (2009), we consider a semantic parsing setting where the goal is to (1) decompose the syntactic dependency tree of a sentence into fragments, (2) assign each of these fragments to a cluster of semantically equivalent syntactic structures, and (3) predict predicate-argument relations between the fragments. We use hierarchical Pitman-Yor processes to model statistical dependencies between meaning representations of predicates and those of their arguments, as well as the clusters of their syntactic realizations. We develop a modification of the Metropolis-Hastings split-merge sampler, resulting in an efficient inference algorithm for the model. The method is experimentally evaluated by using the induced semantic representation for the question answering task in the biomedical domain.

1 Introduction

Statistical approaches to semantic parsing have recently received considerable attention. While some methods focus on predicting a complete formal representation of meaning (Zettlemoyer and Collins, 2005; Ge and Mooney, 2005; Mooney, 2007), others consider more shallow forms of representation (Carreras and Màrquez, 2005; Liang et al., 2009). However, most of this research has concentrated on *supervised* methods requiring large amounts of labeled data. Such annotated resources are scarce, expensive to create and even the largest of them tend to have low coverage (Palmer and Sporleder, 2010), motivating the need for unsupervised or semi-supervised techniques.

Conversely, research in the closely related task of relation extraction has focused on unsupervised or minimally supervised methods (see, for example, (Lin and Pantel, 2001; Yates and Etzioni, 2009)). These approaches cluster semantically equivalent verbalizations of relations, often relying on syntactic fragments as features for relation extraction and clustering (Lin and Pantel, 2001; Banko et al., 2007). The success of these methods suggests that semantic parsing can also be tackled as clustering of syntactic realizations of predicate-argument relations. While a similar direction has been previously explored in (Swier and Stevenson, 2004; Abend et al., 2009; Lang and Lapata, 2010), the recent work of (Poon and Domingos, 2009) takes it one step further by not only predicting predicate-argument structure of a sentence but also assigning sentence fragments to clusters of semantically similar expressions. For example, for a pair of sentences on Figure 1, in addition to inducing predicate-argument structure, they aim to assign expressions *"Steelers"* and *"the Pittsburgh team"* to the same semantic class `Steelers`, and group expressions *"defeated"* and *"secured the victory over"*. Such semantic representation can be useful for entailment or question answering tasks, as an entailment model can abstract away from specifics of syntactic and lexical realization relying instead on the induced semantic representation. For example, the two sentences in Figure 1 have identical semantic representation, and therefore can be hypothesized to be equivalent.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1445–1455,
Portland, Oregon, June 19-24, 2011.

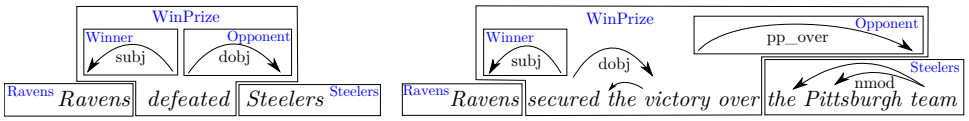


Figure 1: An example of two different syntactic trees with a common semantic representation `WinPrize(Ravens, Steelers)`.

From the statistical modeling point of view, joint learning of predicate-argument structure and discovery of semantic clusters of expressions can also be beneficial, because it results in a more compact model of selectional preference, less prone to the data-sparsity problem (Zapirain et al., 2010). In this respect our model is similar to recent LDA-based models of selectional preference (Ritter et al., 2010; Séaghdha, 2010), and can even be regarded as their recursive and non-parametric extension.

In this paper, we adopt the above definition of unsupervised semantic parsing and propose a Bayesian non-parametric approach which uses hierarchical Pitman-Yor (PY) processes (Pitman, 2002) to model statistical dependencies between predicate and argument clusters, as well as distributions over syntactic and lexical realizations of each cluster. Our non-parametric model automatically discovers granularity of clustering appropriate for the dataset, unlike the parametric method of (Poon and Domingos, 2009) which have to perform model selection and use heuristics to penalize more complex models of semantics. Additional benefits generally expected from Bayesian modeling include the ability to encode prior linguistic knowledge in the form of hyperpriors and the potential for more reliable modeling of smaller datasets. More detailed discussion of relation between the Markov Logic Network (MLN) approach of (Poon and Domingos, 2009) and our non-parametric method is presented in Section 3.

Hierarchical Pitman-Yor processes (or their special case, hierarchical Dirichlet processes) have previously been used in NLP, for example, in the context of syntactic parsing (Liang et al., 2007; Johnson et al., 2007). However, in all these cases the effective size of the state space (i.e., the number of sub-symbols in the infinite PCFG (Liang et al., 2007), or the number of adapted productions in the adaptor grammar (Johnson et al., 2007)) was not very large. In our case, the state space size equals the total number of distinct semantic clusters, and, thus, is expected to be exceedingly large even for moderate datasets: for example, the MLN model induces 18,543 distinct clusters from 18,471 sentences of the GENIA corpus (Poon and Domingos, 2009). This suggests that standard inference methods for hierarchical PY processes, such as Gibbs sampling, Metropolis-Hastings (MH) sampling with uniform proposals, or the structured mean-field algorithm, are unlikely to result in efficient inference: for example in standard Gibbs sampling all thousands of alternatives should be considered at each sampling move. Instead, we use a split-merge MH sampling algorithm, which is a standard and efficient inference tool for non-hierarchical PY processes (Jain and Neal, 2000; Dahl, 2003) but has not previously been used in hierarchical setting. We extend the sampler to include composition-decomposition of syntactic fragments in order to cluster fragments of variables size, as in the example Figure 1, and also include the argument role-syntax alignment move which attempts to improve mapping between semantic roles and syntactic paths for some fixed predicate.

Evaluating unsupervised models is a challenging task. We evaluate our model both qualitatively, examining the revealed clustering of syntactic structures, and quantitatively, on a question answering task. In both cases, we follow (Poon and Domingos, 2009) in using the corpus of biomedical abstracts. Our model achieves favorable results significantly outperforming the baselines, including state-of-the-art methods for relation extraction, and achieves scores comparable to those of the MLN model.

The rest of the paper is structured as follows. Section 2 begins with a definition of the semantic parsing task. Sections 3 and 4 give background on the MLN model and the Pitman-Yor processes, respectively. In Sections 5 and 6, we describe our model and the inference method. Section 7 provides both qualitative and quantitative evaluation. Finally, ad-

ditional related work is presented in Section 8.

2 Semantic Parsing

In this section, we briefly define the unsupervised semantic parsing task and underlying aspects and assumptions relevant to our model.

Unlike (Poon and Domingos, 2009), we do not use the lambda calculus formalism to define our task but rather treat it as an instance of frame-semantic parsing, or a specific type of semantic role labeling (Gildea and Jurafsky, 2002). The reason for this is two-fold: first, the frame semantics view is more standard in computational linguistics, sufficient to describe induced semantic representation and convenient to relate our method to the previous work. Second, lambda calculus is a considerably more powerful formalism than the predicate-argument structure used in frame semantics, normally supporting quantification and logical connectors (for example, negation and disjunction), neither of which is modeled by our model or in (Poon and Domingos, 2009).

In frame semantics, the meaning of a predicate is conveyed by a *frame*, a structure of related concepts that describes a situation, its participants and properties (Fillmore et al., 2003). Each frame is characterized by a set of semantic roles (frame elements) corresponding to the arguments of the predicate. It is evoked by a frame evoking element (a predicate). The same frame can be evoked by different but semantically similar predicates: for example, both verbs *"buy"* and *"purchase"* evoke frame `Commerce_buy` in FrameNet (Fillmore et al., 2003).

The aim of the semantic role labeling task is to identify all of the frames evoked in a sentence and label their semantic role fillers. We extend this task and treat semantic parsing as recursive prediction of predicate-argument structure and clustering of argument fillers. Thus, parsing a sentence into this representation involves (1) decomposing the sentence into lexical items (one or more words), (2) assigning a cluster label (a semantic frame or a cluster of argument fillers) to every lexical item, and (3) predicting argument-predicate relations between the lexical items. This process is illustrated in Figure 1. For the leftmost example, the sentence is decomposed into three lexical items: *"Ravens"*, *"defeated"* and *"Steelers"*, and they are assigned to clusters `Ravens`, `WinPrize` and `Steelers`, respectively. Then `Ravens` and `Steelers` are selected as a `Winner` and an `Opponent` in the `WinPrize` frame. In this work, we define a joint model for the labeling and argument identification stages. Similarly to core semantic roles in FrameNet, semantic roles are treated as frame-specific in our model, as our model does not try to discover any correspondences between roles in different frames.

As you can see from the above description, frames (which groups predicates with similar meaning such as the `WinPrize` frame in our example) and clusters of argument fillers (`Ravens` and `Steelers`) are treated in our definition in a similar way. For convenience, we will refer to both types of clusters as *semantic classes*.[1]

This definition of semantic parsing is closely related to a realistic relation extraction setting, as both clustering of syntactic forms of relations (or extraction patterns) and clustering of argument fillers for these relations is crucial for automatic construction of knowledge bases (Yates and Etzioni, 2009).

In this paper, we make three assumptions. First, we assume that each lexical item corresponds to a subtree of the syntactic dependency graph of the sentence. This assumption is similar to the adjacency assumption in (Zettlemoyer and Collins, 2005), though ours may be more appropriate for languages with free or semi-free word order, where syntactic structures are inherently non-projective. Second, we assume that the semantic arguments are local in the dependency tree; that is, one lexical item can be a semantic argument of another one only if they are connected by an arc in the dependency tree. This is a slight simplification of the semantic role labeling problem but one often made. Thus, the argument identification and labeling stages consist of labeling each syntactic arc with a semantic role label. In comparison, the MLN model does not explicitly assume contiguity of lexical items and does not make this directionality assumption but their clustering algorithm uses initialization and clusterization moves such that the resulting model also obeys both of these constraints. Third, as in (Poon and Domingos, 2009), we do not model polysemy as we assume

[1] Semantic classes correspond to lambda-form clusters in (Poon and Domingos, 2009) terminology.

that each syntactic fragment corresponds to a single semantic class. This is not a model assumption and is only used at inference as it reduces mixing time of the Markov chain. It is not likely to be restrictive for the biomedical domain studied in our experiments.

As in some of the recent work on learning semantic representations (Eisenstein et al., 2009; Poon and Domingos, 2009), we assume that dependency structures are provided for every sentence. This assumption allows us to construct models of semantics not Markovian within a sequence of words (see for an example a model described in (Liang et al., 2009)), but rather Markovian within a dependency tree. Though we include generation of the syntactic structure in our model, we would not expect that this syntactic component would result in an accurate syntactic model, even if trained in a supervised way, as the chosen independence assumptions are oversimplistic. In this way, we can use a simple generative story and build on top of the recent success in syntactic parsing.

3 Relation to the MLN Approach

The work of (Poon and Domingos, 2009) models joint probability of the dependency tree and its latent semantic representation using Markov Logic Networks (MLNs) (Richardson and Domingos, 2006), selecting parameters (weights of first-order clauses) to maximize the probability of the observed dependency structures. For each sentence, the MLN induces a Markov network, an undirected graphical model with nodes corresponding to ground atoms and cliques corresponding to ground clauses.

The MLN is a powerful formalism and allows for modeling complex interaction between features of the input (syntactic trees) and latent output (semantic representation), however, unsupervised learning of semantics with general MLNs can be prohibitively expensive. The reason for this is that MLNs are undirected models and when learned to maximize likelihood of syntactically annotated sentences, they would require marginalization over semantic representation but also over the entire space of syntactic structures and lexical units. Given the complexity of the semantic parsing task and the need to tackle large datasets, even approximate methods are likely to be infeasible. In order to overcome this problem, (Poon and Domingos, 2009) group parameters and impose local normalization constraints within each group. Given these normalization constraints, and additional structural constraints satisfied by the model, namely that the clauses should be engineered in such a way that they induce tree-structured graphs for every sentence, the parameters can be estimated by a variant of the EM algorithm.

The class of such restricted MLNs is equivalent to the class of directed graphical models over the same set of random variables corresponding to fragments of syntactic and semantic structure. Given that the above constraints do not directly fit into the MLN methodology, we believe that it is more natural to regard their model as a directed model with an underlying generative story specifying how the semantic structure is generated and how the syntactic parse is drawn for this semantic structure. This view would facilitate understanding what kind of features can easily be integrated into the model, simplify application of non-parametric Bayesian techniques and expedite the use of inference techniques designed specifically for directed models. Our approach makes one step in this direction by proposing a non-parametric version of such generative model.

4 Hierarchical Pitman-Yor Processes

The central component of our non-parametric Bayesian model are Pitman-Yor (PY) processes, which are a generalization of the Dirichlet processes (DPs) (Ferguson, 1973). We use PY processes to model distributions of semantic classes appearing as an argument of other semantic classes. We also use them to model distributions of syntactic realizations for each semantic class and distributions of syntactic dependency arcs for argument types. In this section we present relevant background on PY processes. For a more detailed consideration we refer the reader to (Teh et al., 2006).

The Pitman-Yor process over a set S, denoted $PY(\alpha, \beta, H)$, is a stochastic process whose samples G_0 constitute probability measures on partitions of S. In practice, we do not need to draw measures, as they can be analytically marginalized out. The conditional distribution of x_{j+1} given the previous j draws, with G_0 marginalized out, follows (Black-

well and MacQueen, 1973)

$$x_{j+1}|x_1,\dots x_j \sim \sum_{k=1}^{K} \frac{j_k-\beta}{j+\alpha}\delta_{\phi_k} + \frac{K\beta+\alpha}{j+\alpha}H. \quad (1)$$

where $\phi_1,\dots,\phi_K$ are K values assigned to $x_1, x_2, \dots, x_j$. The number of times ϕ_k was assigned is denoted j_k, so that $j = \sum_{k=1}^{K} j_k$. The parameter $\beta < 1$ controls how heavy the tail of the distribution is: when it approaches 1, a new value is assigned to every draw, when $\beta = 0$ the PY process reduces to DP. The expected value of K scales as $O(\alpha n^{\beta})$ with the number of draws n, while it scales only logarithmically for DP processes. PY processes are expected to be more appropriate for many NLP problems, as they model power-law type distributions common for natural language (Teh, 2006).

Hierarchical Dirichlet Processes (HDP) or hierarchical PY processes are used if the goal is to draw several related probability measures for the same set S. For example, they can be used to generate transition distributions of a Markov model, HDP-HMM (Teh et al., 2006; Beal et al., 2002). For such a HMM, the top-level state proportions are drawn from the top-level stick breaking construction $\gamma \sim GEM(\alpha,\beta)$, and then the individual transition distributions for every state $z = 1, 2, \dots$ ϕ_z are drawn from $PY(\gamma,\alpha',\beta')$. The parameters α' and β' control how similar the individual transition distributions ϕ_z are to the top-level state proportions γ, or, equivalently, how similar the transition distributions are to each other.

5 A Model for Semantic Parsing

Our model of semantics associates with each semantic class a set of distributions which govern the generation of corresponding syntactic realizations[2] and the selection of semantic classes for its arguments. Each sentence is generated starting from the root of its dependency tree, recursively drawing a semantic class, its syntactic realization, arguments and semantic classes for the arguments. Below we describe the model by first defining the set of the model parameters and then explaining the generation of individual sentences. The generative story is formally presented in Figure 2.

We associate with each semantic class c, $c = 1, 2, \dots$, a distribution of its syntactic realizations ϕ_c. For example, for the frame `WinPrize` illustrated in Figure 1 this distribution would concentrate at syntactic fragments corresponding to lexical items *"defeated"*, *"secured the victory"* and *"won"*. The distribution is drawn from $DP(w^{(C)}, H^{(C)})$, where $H^{(C)}$ is a base measure over syntactic subtrees. We use a simple generative process to define the probability of a subtree, the underlying model is similar to the base measures used in the Bayesian tree-substitution grammars (Cohn et al., 2009). We start by generating a word w uniformly from the treebank distribution, then we decide on the number of dependents of w using the geometric distribution $Geom(q^{(C)})$. For every dependent we generate a dependency relation r and a lexical form w' from $P(r|w)P(w'|r)$, where probabilities P are based on add-0.1 smoothed treebank counts. The process is continued recursively. The smaller the parameter $q^{(C)}$, the lower is the probability assigned to larger sub-trees.

Parameters $\psi_{c,t}$ and $\psi^{+}_{c,t}$, $t = 1,\dots,T$, define a distribution over vectors $(m_1, m_2, \dots, m_T)$ where m_t is the number of times an argument of type t appears for a given semantic frame occurrence[3]. For the frame `WinPrize` these parameters would enforce that there exists exactly one `Winner` and exactly one `Opponent` for each occurrence of `WinPrize`. The parameter $\psi_{c,t}$ defines the probability of having at least one argument of type t. If 0 is drawn from $\psi_{c,t}$ then $m_t = 0$, otherwise the number of additional arguments of type t $(m_t - 1)$ is drawn from the geometric distribution $Geom(\psi^{+}_{c,t})$. This generative story is flexible enough to accommodate both argument types which appear at most once per semantic class occurrence (e.g., agents), and argument types which frequently appear multiple times per semantic class occurrence (e.g., arguments corresponding to descriptors).

Parameters $\phi_{c,t}$, $t = 1,\dots,T$, define the dis-

[2]Syntactic realizations are syntactic tree fragments, and therefore they correspond both to syntactic and lexical variations.

[3]For simplicity, we assume that each semantic class has T associated argument types, note that this is not a restrictive assumption as some of the argument types can remain unused, and T can be selected to be sufficiently large to accommodate all important arguments.

Parameters:

$\gamma \sim GEM(\alpha_0, \beta_0)$ [top-level proportions of classes]
$\theta_{root} \sim PY(\alpha_{root}, \beta_{root}, \gamma)$ [distrib of sem classes at root]
for each sem class $c = 1, 2, \ldots$:
  $\phi_c \sim DP(w^{(C)}, H^{(C)})$ [distribs of synt realizations]
  for each arg type $t = 1, 2, \ldots T$:
    $\psi_{c,t} \sim Beta(\eta_0, \eta_1)$ [first argument generation]
    $\psi^+_{c,t} \sim Beta(\eta^+_0, \eta^+_1)$ [geom distr for more args]
    $\phi_{c,t} \sim DP(w^{(A)}, H^{(A)})$ [distribs of synt paths]
    $\theta_{c,t} \sim PY(\alpha, \beta, \gamma)$ [distrib of arg fillers]

Data Generation:

for each sentence:
  $c_{root} \sim \theta_{root}$ [choose sem class for root]
  GenSemClass(c_{root})

GenSemClass(c):
  $s \sim \phi_c$ [draw synt realization]
  for each arg type $t = 1, \ldots, T$:
    if $[n \sim \psi_{c,t}] = 1$: [at least one arg appears]
      GenArgument(c, t) [draw one arg]
      while $[n \sim \psi^+_{c,t}] = 1$: [continue generation]
        GenArgument(c, t) [draw more args]

GenArgument(c, t):
  $a_{c,t} \sim \phi_{c,t}$ [draw synt relation]
  $c'_{c,t} \sim \theta_{c,t}$ [draw sem class for arg]
  GenSemClass($c'_{c,t}$) [recurse]

Figure 2: The generative story for the Bayesian model for unsupervised semantic parsing.

tributions over syntactic paths for the argument type t. In our example, for argument type `Opponent`, this distribution would associate most of the probability mass with relations *pp_over*, *dobj* and *pp_against*. These distributions are drawn from $DP(w^{(A)}, H^{(A)})$. In this paper we only consider paths consisting of a single relation, therefore the base probability distribution $H^{(A)}$ is just normalized frequencies of dependency relations in the treebank.

The crucial part of the model are the selection-preference parameters $\theta_{c,t}$, the distributions of semantic classes c' for each argument type t of class c. For arguments `Winner` and `Opponent` of the frame `WinPrize` these distributions would assign most of the probability mass to semantic classes denoting teams or players. Distributions $\theta_{c,t}$ are drawn from a hierarchical PY process: first, top-level proportions of classes γ are drawn from $GEM(\alpha_0, \beta_0)$, and then the individual distributions $\theta_{c,t}$ over c' are chosen from $PY(\alpha, \beta, \gamma)$.

For each sentence, we first generate a class corresponding to the root of the dependency tree from the root-specific distribution of semantic classes θ_{root}. Then we recursively generate classes for the entire sentence. For a class c, we generate the syntactic realization s and for each of the T types, decide how many arguments of that type to generate (see **GenSemClass** in Figure 2). Then we generate each of the arguments (see **GenArgument**) by first generating a syntactic arc $a_{c,t}$, choosing a class as its filler $c'_{c,t}$ and, finally, recursing.

6 Inference

In our model, latent states, modeled with hierarchical PY processes, correspond to distinct semantic classes and, therefore, their number is expected to be very large for any reasonable model of semantics. As a result, many standard inference techniques, such as Gibbs sampling, or the structured mean-field method are unlikely to result in tractable inference. One of the standard and most efficient samplers for non-hierarchical PY processes are split-merge MH samplers (Jain and Neal, 2000; Dahl, 2003). In this section we explain how split-merge samplers can be applied to our model.

6.1 Split and Merge Moves

On each move, split-merge samplers decide either to merge two states into one (in our case, merge two semantic classes), or split one state into two. These moves can be computed efficiently for our model of semantics. Note that for any reasonable model of semantics only a small subset of the entire set of semantic classes can be used as an argument for some fixed semantic class due to selectional preferences exhibited by predicates. For instance, only teams or players can fill arguments of the frame `WinPrize` in our running example. As a result, only a small number of terms in the joint distribution has to be evaluated on every move we may consider.

When estimating the model, we start with assigning each distinct word (or, more precisely, a tuple of a word's stem and its part-of-speech tag) to an individual semantic class. Then, we would iterate by selecting a random pair of class occurrences, and decide, at random, whether we attempt to perform a split-merge move or a compose-decompose move.

6.2 Compose and Decompose Moves

The compose-decompose operations modify syntactic fragments assigned to semantic classes, composing two neighboring dependency sub-trees or decomposing a dependency sub-tree. If the two randomly-selected syntactic fragments s and s' correspond to different classes, c and c', we attempt to compose them into $\hat{s}$ and create a new semantic class $\hat{c}$. All occurrences of $\hat{s}$ are assigned to this new class $\hat{c}$. For example, if two randomly-selected occurrences have syntactic realizations *"secure"* and *"victory"* they can be composed to obtain the syntactic fragment *"secure* $\xrightarrow{dobj}$ *victory"*. This fragment will be assigned to a new semantic class which can later be merged with other classes, such as the ones containing syntactic realizations *"defeat"* or *"win"*.

Conversely, if both randomly-selected syntactic fragments are already composed in the corresponding class, we attempt to split them.

6.3 Role-Syntax Alignment Move

Merge, compose and decompose moves require recomputation of mapping between argument types (semantic roles) and syntactic fragments. Computing the best statistical mapping is infeasible and proposing a random mapping will result in many attempted moves being rejected. Instead we use a greedy randomized search method called *Gibbs scan* (Dahl, 2003). Though it is a part of the above 3 moves, this alignment move is also used on its own to induce semantic arguments for classes (frames) with a single syntactic realization.

The Gibbs scan procedure is also used during the split move to select one of the newly introduced classes for each considered syntactic fragment.

6.4 Informed Proposals

Since the number of classes is very large, selecting examples at random would result in a relatively low proportion of moves getting accepted, and, consequently, in a slow-mixing Markov chain. Instead of selecting both class occurrences uniformly, we select the first occurrence from a uniform distribution and then use a simple but effective proposal distribution for selecting the second class occurrence.

Let us denote the class corresponding to the first occurrence as c_1 and its syntactic realization as s_1 with a head word w_1. We begin by selecting uniformly randomly whether to attempt a compose-decompose or a split-merge move.

If we chose a compose-decompose move, we look for words (children) which can be attached below the syntactic fragment s_1. We use the normalized counts of these words conditioned on the parent s_1 to select the second word w_2. We then select a random occurrence of w_2; if it is a part of syntactic realization of c_1 then a decompose move is attempted. Otherwise, we try to compose the corresponding clusters together.

If we selected a split-merge move, we use a distribution based on the cosine similarity of lexical contexts of the words. The context is represented as a vector of counts of all pairs of the form (head word, dependency type) and (dependent, dependency type). So, instead of selecting a word occurrence uniformly, each occurrence of every word w_2 is weighted by its similarity to w_1, where the similarity is based on the cosine distance.

As the moves are dependent only on syntactic representations, all the proposal distributions can be computed once at the initialization stage.[4]

7 Empirical Evaluation

We induced a semantic representation over a collection of texts and evaluated it by answering questions about the knowledge contained in the corpus. We used the GENIA corpus (Kim et al., 2003), a dataset of 1999 biomedical abstracts, and a set of questions produced by (Poon and Domingos, 2009). A example question is shown in Figure 3.

All model hyperpriors were set to maximize the posterior, except for $w^{(A)}$ and $w^{(C)}$, which were set to $1.e-10$ and $1.e-35$, respectively. Inference was run for around 300,000 sampling iterations until the percentage of accepted split-merge moves became lower than 0.05%.

Let us examine some of the induced semantic classes (Table 1) before turning to the question answering task. Almost all of the clustered syntactic

[4]In order to minimize memory usage, we used frequency cut-off of 10. For split-merge moves, we select words based on the cosine distance if the distance is below 0.95 and sample the remaining words uniformly. This also reduces the required memory usage.

Class	Variations
1	motif, sequence, regulatory element, response element, element, dna sequence
2	donor, individual, subject
3	important, essential, critical
4	dose, concentration
5	activation, transcriptional activation, transactivation
6	b cell, t lymphocyte, thymocyte, b lymphocyte, t cell, t-cell line, human lymphocyte, t-lymphocyte
7	indicate, reveal, document, suggest, demonstrate
8	augment, abolish, inhibit, convert, cause, abrogate, modulate, block, decrease, reduce, diminish, suppress, up-regulate, impair, reverse, enhance
9	confirm, assess, examine, study, evaluate, test, resolve, determine, investigate
10	nf-kappab, nf-kappa b, nfkappab, nf-kb
11	antiserum, antibody, monoclonal antibody, ab, antisera, mab
12	tnfalpha, tnf-alpha, il-6, tnf

Table 1: Examples of the induced semantic classes.

realizations have a clear semantic connection. Cluster 6, for example, clusters lymphocytes with the exception of thymocyte, a type of cell which generates T cells. Cluster 8 contains verbs roughly corresponding to `Cause change of position on a scale` frame in FrameNet. Verbs in class 9 are used in the context of providing support for a finding or an action, and many of them are listed as evoking elements for the `Evidence` frame in FrameNet.

Argument types of the induced classes also show a tendency to correspond to semantic roles. For example, an argument type of class 2 is modeled as a distribution over two argument parts, *prep_of* and *prep_from*. The corresponding arguments define the origin of the cells (*transgenic mouse, smoker, volunteer, donor, . . .*).

We now turn to the QA task and compare our model (**USP-BAYES**) with the results of baselines considered in (Poon and Domingos, 2009). The first set of baselines looks for answers by attempting to match a verb and its argument in the question with the input text. The first version (**KW**) simply returns the rest of the sentence on the other side of the verb, while the second (**KW-SYN**) uses syntactic information to extract the subject or the object of the verb.

Other baselines are based on state-of-the-art relation extraction systems. When the extracted relation and one of the arguments match those in a given question, the second argument is returned as an answer. The systems include TextRunner (**TR**) (Banko et al., 2007), RESOLVER (**RS**) (Yates and Etzioni, 2009) and **DIRT** (Lin and Pantel, 2001). The **EXACT** versions of the methods return answers when they match the question argument exactly, and the **SUB** versions produce answers containing the question argument as a substring.

	Total	Correct	Accuracy
KW	150	67	45%
KW-SYN	87	67	77%
TR-EXACT	29	23	79%
TR-SUB	152	81	53%
RS-EXACT	53	24	45%
RS-SUB	196	81	41%
DIRT	159	94	59%
USP-MLN	334	295	88%
USP-BAYES	325	259	80%

Table 2: Performance on the QA task.

Similarly to the MLN system (**USP-MLN**), we generate answers as follows. We use our trained model to parse a question, i.e. recursively decompose it into lexical items and assign them to semantic classes induced at training. Using this semantic representation, we look for the type of an argument missing in the question, which, if found, is reported as an answer. It is clear that overly coarse clusters of argument fillers or clustering of semantically related but not equivalent relations can hurt precision for this evaluation method.

Each system is evaluated by counting the answers it generates, and computing the accuracy of those answers.[5] Table 2 summarizes the results. First, both USP models significantly outperform all other baselines: even though the accuracy of KW-SYN and TR-EXACT are comparable with our accuracy, the number of correct answers returned by USP-Bayes is 4 and 11 times smaller than those of KW-SYN and TR-EXACT, respectively. While we are not beating the MLN baseline, the difference is not significant. The effective number of questions is relatively small (less than 80 different questions are answered by any of the models). More than 50% of USP-BAYES mistakes were due to wrong interpretation of only 5 different questions. From another point of view, most of the mistakes are explained

[5]The true recall is not known, as computing it would require exhaustive annotation of the entire corpus.

Question: *What does cyclosporin A suppress?*
Answer: *expression of EGR-2*
Sentence: *As with EGR-3 , expression of EGR-2 was blocked by cyclosporin A .*

Question: *What inhibits tnf-alpha?*
Answer: *IL -10*
Sentence: *Our previous studies in human monocytes have demonstrated that interleukin (IL) -10 inhibits lipopolysaccharide (LPS) -stimulated production of inflammatory cytokines , IL-1 beta , IL-6 , IL-8 , and tumor necrosis factor (TNF) -alpha by blocking gene transcription .*

Figure 3: An example of questions, answers by our model and the corresponding sentences from the dataset.

by overly coarse clustering corresponding to just 3 classes, namely, 30%, 25% and 20% of errors are due to the clusters 6, 8 and 12 (Figure 1), respectively. Though all these clusters have clear semantic interpretation (white blood cells, predicates corresponding to changes and cykotines associated with cancer progression, respectively), they appear to be too coarse for the QA method we use in our experiments. Though it is likely that tuning and different heuristics may result in better scores, we chose not to perform excessive tuning, as the evaluation dataset is fairly small.

8 Related Work

There is a growing body of work on statistical learning for different versions of the semantic parsing problem (e.g., (Gildea and Jurafsky, 2002; Zettlemoyer and Collins, 2005; Ge and Mooney, 2005; Mooney, 2007)), however, most of these methods rely on human annotation, or some weaker forms of supervision (Kate and Mooney, 2007; Liang et al., 2009; Titov and Kozhevnikov, 2010; Clarke et al., 2010) and very little research has considered the unsupervised setting.

In addition to the MLN model (Poon and Domingos, 2009), another unsupervised method has been proposed in (Goldwasser et al., 2011). In that work, the task is to predict a logical formula, and the only supervision used is a lexicon providing a small number of examples for every logical symbol. A form of self-training is then used to bootstrap the model.

Unsupervised semantic role labeling with a generative model has also been considered (Grenager and Manning, 2006), however, they do not attempt to discover frames and deal only with isolated predicates. Another generative model for SRL has been proposed in (Thompson et al., 2003), but the parameters were estimated from fully annotated data.

The unsupervised setting has also been considering for the related problem of learning narrative schemas (Chambers and Jurafsky, 2009). However, their approach is quite different from our Bayesian model as it relies on similarity functions.

Though in this work we focus solely on the unsupervised setting, there has been some successful work on semi-supervised semantic-role labeling, including the Framenet version of the problem (Fürstenau and Lapata, 2009). Their method exploits graph alignments between labeled and unlabeled examples, and, therefore, crucially relies on the availability of labeled examples.

9 Conclusions and Future Work

In this work, we introduced a non-parametric Bayesian model for the semantic parsing problem based on the hierarchical Pitman-Yor process. The model defines a generative story for recursive generation of lexical items, syntactic and semantic structures. We extend the split-merge MH sampling algorithm to include composition-decomposition moves, and exploit the properties of our task to make it efficient in the hierarchical setting we consider.

We plan to explore at least two directions in our future work. First, we would like to relax some of unrealistic assumptions made in our model: for example, proper modeling of alterations requires joint generation of syntactic realizations for predicate-argument relations (Grenager and Manning, 2006; Lang and Lapata, 2010), similarly, proper modeling of nominalization implies support of arguments not immediately local in the syntactic structure. The second general direction is the use of the unsupervised methods we propose to expand the coverage of existing semantic resources, which typically require substantial human effort to produce.

Acknowledgements

The authors acknowledge the support of the MMCI Cluster of Excellence, and thank Chris Callison-Burch, Alexis Palmer, Caroline Sporleder, Ben Van Durme and the anonymous reviewers for their helpful comments and suggestions.

References

O. Abend, R. Reichart, and A. Rappoport. 2009. Unsupervised argument identification for semantic role labeling. In *Proceedings of ACL-IJCNLP*, pages 28–36, Singapore.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2670–2676.

Matthew J. Beal, Zoubin Ghahramani, and Carl E. Rasmussen. 2002. The infinite hidden markov model. In *Machine Learning*, pages 29–245. MIT Press.

David Blackwell and James B. MacQueen. 1973. Ferguson distributions via polya urn schemes. *The Annals of Statistics*, 1(2):353–355.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of the 9th Conference on Natural Language Learning, CoNLL-2005*, Ann Arbor, MI USA.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proc. of the Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*.

Trevor Cohn, Sharon Goldwater, and Phil Blunsom. 2009. Inducing compact but accurate tree-substitution grammars. In *HLT-NAACL*, pages 548–556.

David B. Dahl. 2003. An improved merge-split sampler for conjugate dirichlet process mixture models. Technical Report 1086, Department of Statistics, University of Wiscosin - Madison, November.

Jacob Eisenstein, James Clarke, Dan Goldwasser, and Dan Roth. 2009. Reading to learn: Constructing features from semantic abstracts. In *Proceedings of EMNLP*.

Thomas S. Ferguson. 1973. A bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230.

C. J. Fillmore, C. R. Johnson, and M. R. L. Petruck. 2003. Background to framenet. *International Journal of Lexicography*, 16:235–250.

Hagen Fürstenau and Mirella Lapata. 2009. Graph alignment for semi-supervised semantic role labeling. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

Ruifang Ge and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CONLL-05)*, Ann Arbor, Michigan.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labelling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Dan Goldwasser, Roi Reichart, James Clarke, and Dan Roth. 2011. Confidence driven unsupervised semantic parsing. In *Proc. of the Meeting of Association for Computational Linguistics (ACL)*, Portland, OR, USA.

Trond Grenager and Christoph Manning. 2006. Unsupervised discovery of a statistical verb lexicon. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

Sonia Jain and Radford Neal. 2000. A split-merge markov chain monte carlo procedure for the dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13:158–182.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, Rochester, USA.

Rohit J. Kate and Raymond J. Mooney. 2007. Learning language semantics from ambigous supervision. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 895–900.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19:i180–i182.

Joel Lang and Mirella Lapata. 2010. Unsupervised induction of semantic roles. In *Proceedings of the 48rd Annual Meeting of the Association for Computational Linguistics (ACL)*, Uppsala, Sweden.

Percy Liang, Slav Petrov, Michael Jordan, and Dan Klein. 2007. The infinite PCFG using hierarchical dirichlet processes. In *Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 688–697, Prague, Czech Republic.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proc. of the Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Dekang Lin and Patrick Pantel. 2001. Dirt – discovery of inference rules from text. In *Proc. of International Conference on Knowledge Discovery and Data Mining*, pages 323–328.

Raymond J. Mooney. 2007. Learning for semantic parsing. In *Proceedings of the 8th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 982–991.

Alexis Palmer and Caroline Sporleder. 2010. Evaluating framenet-style semantic parsing: the role of coverage gaps in framenet. In *Proceedings of the Conference on Computational Linguistics (COLING-2000)*, Beijing.

Jim Pitman. 2002. Poisson-dirichlet and gem invariant distributions for split-and-merge transformations of an interval partition. *Combinatorics, Probability and Computing*, 11:501–514.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, (EMNLP-09)*.

Matt Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning*, 62:107–136.

Alan Ritter, Mausam, and Oren Etzioni. 2010. A latent dirichlet allocation method for selectional preferences. In *Proceedings of the 48rd Annual Meeting of the Association for Computational Linguistics (ACL)*, Uppsala, Sweden.

Diarmuid Ó Séaghdha. 2010. Latent variable models of selectional preference. In *Proceedings of the 48rd Annual Meeting of the Association for Computational Linguistics (ACL)*, Uppsala, Sweden.

R. Swier and S. Stevenson. 2004. Unsupervised semantic role labelling. In *Proceedings of EMNLP*, pages 95–102, Barcelona, Spain.

Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Y. W. Teh. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 985–992.

Cynthia A. Thompson, Roger Levy, and Christopher D. Manning. 2003. A generative model for semantic role labeling. In *In Senseval-3*, pages 397–408.

Ivan Titov and Mikhail Kozhevnikov. 2010. Bootstrapping semantic analyzers from non-contradictory texts. In *Proceedings of the 48rd Annual Meeting of the Association for Computational Linguistics (ACL)*, Uppsala, Sweden.

Alexander Yates and Oren Etzioni. 2009. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34:255–296.

B. Zapirain, E. Agirre, L. L. Màrquez, and M. Surdeanu. 2010. Improving semantic role classification with selectional prefrences. In *Proceedings of the Meeting of the North American chapter of the Association for Computational Linguistics (NAACL 2010)*, Los Angeles.

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammar. In *Proceedings of the Twenty-first Conference on Uncertainty in Artificial Intelligence*, Edinburgh, UK, August.

Unsupervised Learning of Semantic Relation Composition

Eduardo Blanco and **Dan Moldovan**
Human Language Technology Research Institute
The University of Texas at Dallas
Richardson, TX 75080 USA
{eduardo,moldovan}@hlt.utdallas.edu

Abstract

This paper presents an unsupervised method for deriving inference axioms by composing semantic relations. The method is independent of any particular relation inventory. It relies on describing semantic relations using primitives and manipulating these primitives according to an algebra. The method was tested using a set of eight semantic relations yielding 78 inference axioms which were evaluated over PropBank.

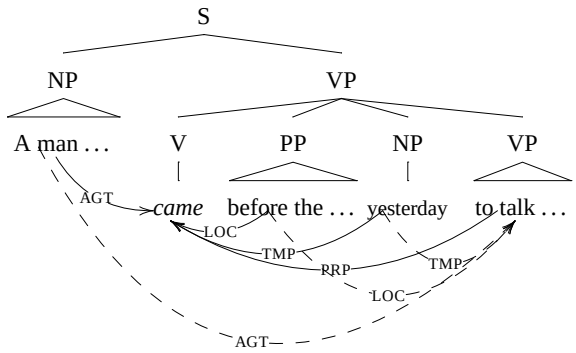


Figure 1: Semantic representation of *A man from the Bush administration came before the House Agricultural Committee yesterday to talk about* ... (wsj_0134, 0).

1 Introduction

Capturing the meaning of text is a long term goal within the NLP community. Whereas during the last decade the field has seen syntactic parsers mature and achieve high performance, the progress in semantics has been more modest. Previous research has mostly focused on relations between particular kind of arguments, e.g., semantic roles, noun compounds. Notwithstanding their significance, they target a fairly narrow text semantics compared to the broad semantics encoded in text.

Consider the sentence in Figure 1. Semantic role labelers exclusively detect the relations indicated with solid arrows, which correspond to the sentence syntactic dependencies. On top of those roles, there are at least three more relations (discontinuous arrows) that encode semantics other than the verb-argument relations.

In this paper, we venture beyond semantic relation extraction from text and investigate techniques to compose them. We explore the idea of inferring a new relation linking the ends of a chain of relations. This scheme, informally used previously for combining HYPERNYM with other relations, has not been studied for arbitrary pairs of relations.

For example, it seems adequate to state the following: if *x* is PART-OF *y* and *y* is HYPERNYM of *z*, then *x* is PART-OF *z*. An inference using this rule can be obtained instantiating *x*, *y* and *z* with *engine*, *car* and *convertible*. Going a step further, we consider nonobvious inferences involving AGENT, PURPOSE and other semantic relations.

The novelties of this paper are twofold. First, an extended definition for semantic relations is proposed, including (1) semantic restrictions for their domains and ranges, and (2) semantic primitives.

Second, an algorithm for obtaining inference axioms is described. Axioms take as their premises chains of two relations and output a new relation linking the ends of the chain. This adds an extra layer of semantics on top of previously extracted re-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1456–1465,
Portland, Oregon, June 19-24, 2011.

Primitive	Description	Inv.	Ref.
1: Composable	Relation can be meaningfully composed with other relations due to their fundamental characteristics	id.	[3]
2: Functional	*x* is in a specific spatial or temporal position with respect to *y* in order for the connection to exist	id.	[1]
3: Homeomerous	*x* must be the same kind of thing as *y*	id.	[1]
4: Separable	*x* can be temporally or spatially separated from *y*; they can exist independently	id.	[1]
5: Temporal	*x* temporally precedes *y*	op.	[2]
6: Connected	*x* is physically or temporally connected to *y*; connection might be indirect.	id.	[3]
7: Intrinsic	Relation is an attribute of the essence/stufflike nature of *x* and *y*	id.	[3]
8: Volitional	Relation requires volition between the arguments	id.	-
9: Universal	Relation is always true between *x* and *y*	id.	-
10: Fully Implicational	The existence of *x* implies the existence of *y*	op.	-
11: Weakly Implicational	The existence of *x* sometimes implies the existence of *y*	op.	-

Table 1: List of semantic primitives. In the fourth column, [1] stands for (Winston et al., 1987), [2] for (Cohen and Losielle, 1988) and [3] for (Huhns and Stephens, 1989).

lations. The conclusion of an axiom is identified using an algebra for composing semantic primitives.

We name this framework Composition of Semantic Relations (CSR). The extended definition, set of primitives, algebra to compose primitives and CSR algorithm are independent of any particular set of relations. We first presented CSR and used it over PropBank in (Blanco and Moldovan, 2011). In this paper, we extend that work using a different set of primitives and relations. Seventy eight inference axioms are obtained and an empirical evaluation shows that inferred relations have high accuracies.

2 Semantic Relations

Semantic relations are underlying relations between concepts. In general, they are defined by a textual definition accompanied by a few examples. For example, Chklovski and Pantel (2004) loosely define ENABLEMENT as a relation that holds between two verbs V_1 and V_2 when the pair can be glossed as *V_1 is accomplished by V_2* and gives two examples: *assess::review* and *accomplish::complete*.

We find this widespread kind of definition weak and prone to confusion. Following (Helbig, 2005), we propose an extended definition for semantic relations, including semantic restrictions for its arguments. For example, AGENT(*x*, *y*) holds between an animate concrete object *x* and a situation *y*.

Moreover, we propose to characterize relations by *semantic primitives*. Primitives indicate whether a property holds between the arguments of a relation, e.g., the primitive *temporal* indicates if the first argument must happen before the second.

Besides having a better understanding of each relation, this extended definition allows us to identify possible and not possible combinations of relations, as well as to automatically determine the conclusion of composing a possible combination.

Formally, for a relation R(*x*, *y*), the extended definitions specifies: (a) DOMAIN(R) and RANGE(R) (i.e., semantic restrictions for *x* and *y*); and (b) P_{R} (i.e., values for the primitives). The inverse relation R^{-1} can be obtained by switching domain and range, and defining $P_{\text{R}^{-1}}$ as depicted in Table 1.

2.1 Semantic Primitives

Semantic primitives capture deep characteristics of relations. They are independently determinable for each relation and specify a property between an element of the domain and an element of the range of the relation being described (Huhns and Stephens, 1989). Primitives are fundamental, they cannot be explained using other primitives.

For each primitive, each relation takes a value from the set $V = \{+, -, 0\}$. '+' indicates that the primitive holds, '$-$' that it does not hold, and '0' that it does not apply. Since a cause must precede its effect, we have $P^{temporal}_{\text{CAUSE}} = +$.

Primitives complement the definition of a relation and completely characterize it. Coupled with domain and range restrictions, primitives allow us to automatically manipulate and reason over relations.

1:Composable

R_1 \ R_2	−	0	+
−	×	0	×
0	0	0	0
+	×	0	+

2:Functional

R_1 \ R_2	−	0	+
−	−	0	+
0	0	0	0
+	+	0	+

3:Homeomerous

R_1 \ R_2	−	0	+
−	−	−	−
0	−	0	0
+	−	0	+

4:Separable

R_1 \ R_2	−	0	+
−	−	−	−
0	−	0	+
+	−	+	+

5:Temporal

R_1 \ R_2	−	0	+
−	−	−	×
0	−	0	+
+	×	+	+

6:Connected

R_1 \ R_2	−	0	+
−	−	−	+
0	−	0	+
+	+	+	+

7:Intrinsic

R_1 \ R_2	−	0	+
−	−	0	−
0	0	0	0
+	−	0	+

8:Volitional

R_1 \ R_2	−	0	+
−	−	0	+
0	0	0	+
+	+	+	+

9:Universal

R_1 \ R_2	−	0	+
−	−	0	−
0	0	0	0
+	−	0	+

10:F. Impl.

R_1 \ R_2	−	0	+
−	−	0	×
0	0	0	0
+	×	0	+

11:W. Impl.

R_1 \ R_2	−	0	+
−	−	−	×
0	−	0	+
+	×	+	+

Table 2: Algebra for composing semantic primitives.

The set of primitives used in this paper (Table 1) is heavily based on previous work in Knowledge Bases (Huhns and Stephens, 1989), but we considered some new primitives. The new primitives are justified by the fact that we aim at composing relations capturing the semantics from natural language. Whatever the set of relations, it will describe the characteristics of events (who / what / where / when / why / how) and connections between them (e.g., CAUSE, CORRELATION). Time, space and volition also play an important role. The third column in Table 1 indicates the value of the primitive for the inverse relation: *id.* means it takes the same; *op.* the opposite. The opposite of $-$ is $+$, the opposite of $+$ is $-$, and the opposite of 0 is 0.

2.1.1 An Algebra for Composing Semantic Primitives

The key to automatically obtain inference axioms is the ability to know the result of composing primitives. Given $P^i_{R_1}$ and $P^i_{R_2}$, i.e., the values of the ith primitive for R_1 and R_2, we define an algebra for $P^i_{R_1} \circ P^i_{R_2}$, i.e., the result of composing them. Table 2 depicts the algebra for all primitives. An '$\times$' means that the composition is prohibited.

Consider, for example, the Intrinsic primitive: if both relations are *intrinsic* ($+$), the composition is *intrinsic* ($+$); else if *intrinsic* does not apply to either relation (0), the primitive does not apply to the composition either (0); else the composition is not *intrinsic* ($-$).

3 Inference Axioms

Semantic relations are composed using inference axioms. An axiom is defined by using the composition operator '$\circ$'; it combines two relations called *premises* and yields a *conclusion*. We denote an axiom as $R_1(x, y) \circ R_2(y, z) \rightarrow R_3(x, z)$, where R_1 and R_2 are the premises and R_3 the conclusion. In order to instantiate an axiom, the premises must form a chain by having argument *y* in common.

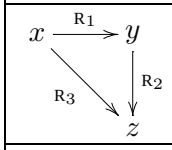
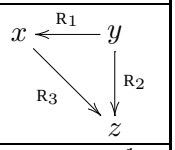
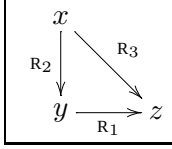
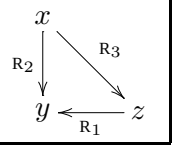
$R_1 \circ R_2$	${R_1}^{-1} \circ R_2$
	
$R_2 \circ R_1$	$R_2 \circ {R_1}^{-1}$
	

Table 3: The four unique possible axioms taking as premises R_1 and R_2. Conclusions are indicated by R_3 and are not guaranteed to be the same for the four axioms.

In general, for n relations there are $\binom{n}{2}$ pairs. For each pair, taking into account inverse relations, there are 16 possible combinations. Applying property $R_i \circ R_j = ({R_j}^{-1} \circ {R_i}^{-1})^{-1}$, only 10 are unique: (a) 4 combine R_1, R_2 and their inverses (Table 3); (b) 3 combine R_1 and ${R_1}^{-1}$; and (c) 3 combine R_2 and ${R_2}^{-1}$. The most interesting axioms fall into category (a) and there are $\binom{n}{2} \times 4 + 3n = 2 \times n(n-1) + 3n = 2n^2 + n$ potential axioms in this category.

Depending on n, the number of potential axioms to consider can be significantly large. For $n = 20$, there are 820 axioms to explore and for $n = 30$, 1,830. Manual examination of those potential ax-

Relation R		Domain	Range	P_R^1	P_R^2	P_R^3	P_R^4	P_R^5	P_R^6	P_R^7	P_R^8	P_R^9	P_R^{10}	P_R^{11}
a: CAU	CAUSE	si	si	+	+	-	+	+	-	+	0	-	+	+
b: INT	INTENT	si	aco	+	+	-	+	-	-	-	+	-	0	-
c: PRP	PURPOSE	si, ao	si, co, ao	+	-	-	+	-	-	-	-	-	0	-
d: AGT	AGENT	aco	si	+	+	-	+	0	-	-	+	-	0	0
e: MNR	MANNER	st, ao, ql	si	+	-	-	+	0	-	-	+	-	0	0
f: AT-L	AT-LOCATION	o, si	loc	+	+	-	0	0	+	-	0	-	0	0
g: AT-T	AT-TIME	o, si	tmp	+	+	-	0	0	+	-	0	-	0	0
h: SYN	SYNONYMY	ent	ent	+	-	+	0	0	0	+	0	+	0	0

Table 4: Extended definition for the set of relations.

ioms would be time-consuming and prone to errors. We avoid this by using the extended definition and the algebra for composing primitives.

3.1 Necessary Conditions for Composing Semantic Relations

There are two necessary conditions for composing R_1 and R_2:

- They have to be compatible. A pair of relations is compatible if it is possible, from a theoretical point of view, to compose them.
  Formally, R_1 and R_2 are compatible iff $\text{RANGE}(R_1) \cap \text{DOMAIN}(R_2) \neq \emptyset$.
- A third relation R_3 must match as conclusion, i.e., $\exists R_3$ such that $\text{DOMAIN}(R_3) \cap \text{DOMAIN}(R_1) \neq \emptyset$ and $\text{RANGE}(R_3) \cap \text{RANGE}(R_2) \neq \emptyset$. Furthermore, P_{R_3} must be consistent with $P_{R_1} \circ P_{R_2}$.

3.2 CSR: An Algorithm for Composing Semantic Relations

Consider any set of relations R defined using the extended definition. One can obtain inference axioms using the following algorithm:

For $(R_1, R_2) \in R \times R$:
  For $(R_i, R_j) \in [(R_1, R_2), (R_1{}^{-1}, R_2), (R_2, R_1), (R_2, R_1{}^{-1})]$:

1. **Domain and range compatibility**
   If $\text{RANGE}(R_i) \cap \text{DOMAIN}(R_j) = \emptyset$, **break**
2. **Conclusion match**
   Repeat for $R_3 \in possible\_conc(R, R_i, R_j)$:
   (a) **If** $\text{DOMAIN}(R_3) \cap \text{DOMAIN}(R_i) = \emptyset$ **or** $\text{RANGE}(R_3) \cap \text{RANGE}(R_j) = \emptyset$**, break**
   (b) **If** $consistent(P_{R_3}, P_{R_i} \circ P_{R_j})$**,**
   $axioms$ += $R_i(x, y) \circ R_j(y, z) \rightarrow R_3(x, z)$

Given R, R^{-1} can be automatically obtained (Section 2). $Possible\_conc(R, R_i, R_j)$ returns the set R unless R_i (R_j) is universal ($P^9 = +$), in which case it returns R_j (R_i). $Consistent(P_{R_1}, P_{R_2})$ is a simple procedure that compares the values assigned to each primitive; two values are consistent unless they have different opposite values or any of them is '$\times$' (i.e., the composition is prohibited).

3.3 An Example: Agent and Purpose

We present an example of applying the CSR algorithm by inspecting the potential axiom AGENT(*x*, *y*) $\circ$ PURPOSE^{-1}(*y*, *z*) $\rightarrow$ R_3(*x*, *z*), where *x* is the agent of *y*, and action *y* has as its purpose *z*. A statement instantiating the premises is *[Mary]$_x$ [came]$_y$ to [talk]$_z$ about the issue*. Knowing AGENT(*Mary*, *came*) and PURPOSE^{-1}(*came*, *talk*), our goal is to identify the links R_3(*Mary*, *talk*), if any.

We use the relations as defined in Table 4. First, we note that both AGENT and PURPOSE^{-1} are compatible (Step 1). Second, we must identify the possible conclusions R_3 that fit as conclusions (Step 2).

Given P_{AGENT} and $P_{\text{PURPOSE}^{-1}}$, we obtain $P_{\text{AGENT}} \circ P_{\text{PURPOSE}^{-1}}$ using the algebra:

$$P_{\text{AGENT}} = \{+,+,-,+,\ 0,-,-,+,-,0,\ 0\}$$
$$P_{\text{PURPOSE}^{-1}} = \{+,-,-,+,+,-,-,-,-,0,+\}$$
$$P_{\text{AGENT}} \circ P_{\text{PURPOSE}^{-1}} = \{+,+,-,+,+,-,-,+,-,0,+\}$$

Out of all relations (Section 4), AGENT and INTENT^{-1} fit the conclusion match. First, their domains and ranges are compatible with the composition (Step 2a). Second, both P_{AGENT} and $P_{\text{INTENT}^{-1}}$ are consistent with $P_{\text{AGENT}} \circ P_{\text{PURPOSE}^{-1}}$ (Step 2b). Thus, we obtain the following axioms: AGENT(*x*, *y*) $\circ$ PURPOSE^{-1}(*y*, *z*) $\rightarrow$ AGENT(*x*, *z*) and AGENT(*x*, *y*) $\circ$ PURPOSE^{-1}(*y*, *z*) $\rightarrow$ INTENT^{-1}(*x*, *z*).

Instantiating the axioms over *[Mary]$_x$ [came]$_y$ to [talk]$_z$ about the issue* yields AGENT(*Mary*, *talk*) and INTENT^{-1}(*Mary*, *talk*). Namely, the axioms

R_1 \ R_2	a	b	c	d	e	f	g	h
a	a	:	:		-	f	g	a
b				-		f	g	b
c	:	b	c	-	e	f	g	c
d	d	-	d		d	f	g	d
e	-	b	e		e	f	g	e
f								f
g								g
h	a	b	c	d	e	f	g	h

R_1 \ R_2	a	b	c	d	e	f	g	h
$\mathbf{a}^{-1}$	:	b	b		-	f	g	a^{-1}
$\mathbf{b}^{-1}$	b^{-1}	:	:		b^{-1},d^{-1}	f	g	b^{-1}
$\mathbf{c}^{-1}$	b^{-1}	:	:		e	f	g	c^{-1}
$\mathbf{d}^{-1}$				-		f	g	d^{-1}
$\mathbf{e}^{-1}$	-	b,d	e^{-1}		e,e^{-1}	f	g	e^{-1}
$\mathbf{f}^{-1}$	f^{-1}	f^{-1}	f^{-1}	f^{-1}	f^{-1}	-	-	f^{-1}
$\mathbf{g}^{-1}$	g^{-1}	g^{-1}	g^{-1}	g^{-1}	g^{-1}	-	-	g^{-1}
$\mathbf{h}^{-1}$	a	b	c	d	e	f	g	h,h^{-1}

R_1 \ R_2	a^{-1}	b^{-1}	c^{-1}	d^{-1}	e^{-1}	f^{-1}	g^{-1}	h^{-1}
a	:		:	d^{-1}	-			a
b		:	:					b
c	:	:	:	b,d^{-1}	e^{-1}			c
d	d		b^{-1},d	-	b,d			d
e	-		e	b^{-1},d^{-1}	e,e^{-1}			e
f						-		f
g							-	g
h	a^{-1}	b^{-1}	c^{-1}	d^{-1}	e^{-1}	f^{-1}	g^{-1}	h,h^{-1}

Table 5: Inference axioms automatically obtained using the relations from Table 4. A letter indicates an axiom $R_1 \circ R_2 \rightarrow R_3$ by indicating R_3. An empty cell indicates that R_1 and R_2 do not have compatible domains and ranges; ':' that the composition is prohibited; and '-' that a relation R_3 such that P_{R_3} is consistent with $P_{R_1} \circ P_{R_2}$ could not be found.

yield *Mary is the agent of talking*, and *she has the intention of talking*. These two relations are valid but most probably ignored by a role labeler since *Mary* is not an argument of *talk*.

4 Case Study

In this Section, we apply the CSR algorithm over a set of eight well-known relations. It is out of the scope of this paper to explain in detail the semantics of each relation or their detection. Our goal is to obtain inference axioms and, taking for granted that annotation is available, evaluate their accuracy.

The only requirement for the CSR algorithm is to define semantic relations using the extended definition (Table 4). To define domains and ranges, we use the ontology in Section 4.2. Values for the primitives are assigned manually. The meaning of each relations is as follows:

- CAU(*x*, *y*) encodes a relation between two situations, where the existence of *y* is due to the previous existence of *x*, e.g., *He [got]$_y$ a bad grade because he [didn't submit]$_x$ the project*.
- INT(*x*, *y*) links an animate concrete object and the situations he wants to become true, e.g., *[Mary]$_y$ would like to [grow]$_x$ bonsais*.
- PRP(*x*, *y*) holds between a concept *y* and its main goal *x*. Purposes can be defined for situations, e.g., *[pruning]$_y$ allows new [growth]$_x$*; concrete objects, e.g., *the [garage]$_y$ is used for [storage]$_x$*; or abstract objects, e.g., *[language]$_y$ is used to [communicate]$_x$*.
- AGT(*x*, *y*) links a situation *y* and its intentional doer *x*, e.g., *[Mary]$_x$ [went]$_y$ to Paris*. *x* is restricted to animate concrete objects.
- MNR(*x*, *y*) holds between the mode, way, style or fashion *x* in which a situation *y* happened. *x* can be a state, e.g., *[walking]$_y$ [holding]$_x$ hands*; abstract objects, e.g., *[die]$_y$ [with pain]$_x$*; or qualities, e.g. *[fast]$_x$ [delivery]$_y$*.
- AT-L(*x*, *y*) defines the spatial context *y* of an object or situation *x*, e.g., *He [went]$_x$ [to Cancun]$_y$*, *[The car]$_x$ is [in the garage]$_y$*.
- AT-T(*x*, *y*) links an object or situation *x*, with its temporal information *y*, e.g., *He [went]$_x$ [yesterday]$_y$*, *[20th century]$_y$ [sculptures]$_x$*.
- SYN(*x*, *y*) can be defined between any two entities and holds when both arguments are semantically equivalent, e.g., SYN(*dozen*, *twelve*).

4.1 Inference Axioms Automatically Obtained

After applying the CSR algorithm over the relations in Table 4, we obtain 78 unique inference axioms (Table 5). Each sub table must be indexed with the first and second premises as row and column respectively. The table on the left summarizes axioms $R_1 \circ R_2 \rightarrow R_3$ and $R_2 \circ R_1 \rightarrow R_3$, the one in the middle axiom ${R_1}^{-1} \circ R_2 \rightarrow R_3$ and the one on the right axiom $R_2 \circ {R_1}^{-1} \rightarrow R_3$.

The CSR algorithm identifies several correct axioms and accurately marks as prohibited several combinations that would lead to wrong inferences:

- For CAUSE, the inherent transitivity is detected ($a \circ a \rightarrow a$). Also, no relation is inferred between two different effects of the same cause ($a^{-1} \circ a \rightarrow :$) and between two causes of the same effect ($a \circ a^{-1} \rightarrow :$).
- The location and temporal information of concept *y* is inherited by its cause, intention, purpose, agent and manner (sub table on the left, *f* and *g* columns).

- As expected, axioms involving SYNONYMY as one of their premises yield the other premise as their conclusion (all sub tables).
- The AGENT of *y* is inherited by its causes, purposes and manners (*d* row, sub table on the right). In all examples below, AGT(*x*, *y*) holds, and we infer AGT(*x*, *z*) after composing it with R_2: (1) *[He]$_x$ [went]$_y$ after [reading]$_z$ a good review*, R_2: CAU^{-1}(*y*, *z*); (2) *[They]$_x$ [went]$_y$ to [talk]$_z$ about it*, R_2: PRP^{-1}(*y*, *z*); and (3) *[They]$_x$ [were walking]$_y$ [holding]$_z$ hands*, R_2: MNR^{-1}(*y*, *z*)
  An AGENT for a situation *y* is also inherited by its effects, and the situations that have *y* as their manner or purpose (*d* row, sub table on the left).
- A concept intends the effects of its intentions and purposes ($b^{-1} \circ a \rightarrow b^{-1}$, $c^{-1} \circ a \rightarrow b^{-1}$). For example, *[I]$_x$ printed the document to [read]$_y$ and [learn]$_z$ the contents*; INT^{-1}(*I, read*) $\circ$ CAU(*read, learn*) $\rightarrow$ INT^{-1}(*I, learn*).

It is important to note that domain and range restrictions are not sufficient to identify inference axioms; they only filter out pairs of not compatible relations. The algebra to compose primitives is used to detect prohibited combinations of relations based on semantic grounds and identify the conclusion of composing them. Without primitives, the cells in Table 5 would be either empty (marking the pair as not compatible) or would simply indicate that the pair has compatible domain and range (without identifying the conclusion).

Table 5 summarizes 136 unique pairs of premises (recall $R_i \circ R_j = ({R_j}^{-1} \circ {R_i}^{-1})^{-1}$). Domain and range restrictions mark 39 (28.7%) as not compatible. The algebra labels 12 pairs as prohibited (8.8%, [12.4% of the compatible pairs]) and is unable to find a conclusion 14 times (10.3%, [14.4%]). Finally, conclusions are found for 71 pairs (52.2%, [73.2%]). Since more than one conclusion might be detected for the same pair of premises, 78 inference axioms are ultimately identified.

4.2 Ontology

In order to define domains and ranges, we use a simplified version of the ontology presented in (Helbig, 2005). We find enough to contemplate only seven base classes: ev, st, co, aco, ao, loc and tmp. Entities (ent) refer to any concept and are divided into situations (si), objects (o) and descriptors (des).

- Situations are anything that happens at a time and place and are divided into events (ev) and states (st). Events imply a change in the status of other entities (e.g., *grow*, *conference*); states do not (e.g., *be standing*, *account for 10%*).
- Objects can be either concrete (co, palpable, tangible, e.g., *table*, *keyboard*) or abstract (ao, intangible, product of human reasoning, e.g., *disease*, *weight*). Concrete objects can be further classified as animate (aco) if they have life, vigor or spirit (e.g. *John*, *cat*).
- Descriptors state properties about the local (loc, e.g., *by the table*, *in the box*) or temporal (tmp, e.g., *yesterday*, *last month*) context of an entity.

This simplified ontology does not aim at defining domains and ranges for any relation set; it is a simplification to fit the eight relations we work with.

5 Evaluation

An evaluation was performed to estimate the validity of the 78 axioms. Because the number of axioms is large we have focused on a subset of them (Table 6). The 31 axioms having SYN as premise are intuitively correct: since synonymous concepts are interchangeable, given veracious annotation they perform valid inferences.

We use PropBank annotation (Palmer et al., 2005) to instantiate the premises of each axiom. First, all instantiations of axiom PRP $\circ$ $MNR^{-1} \rightarrow MNR^{-1}$ were manually checked. This axiom yields 237 new MANNER, 189 of which are valid (Accuracy 0.80).

Second, we evaluated axioms 1–7 (Table 6). Since PropBank is a large corpus, we restricted this phase to the first 1,000 sentences in which there is an instantiation of any axiom. These sentences contain 1,412 instantiations and are found in the first 31,450 sentences of PropBank.

Table 6 depicts the total number of instantiations for each axiom and its accuracy (columns 3 and 4). Accuracies range from 0.40 to 0.90, showing that the plausibility of an axiom depends on the axiom. The average accuracy for axioms involving CAU is 0.54 and for axioms involving PRP is 0.87.

Axiom CAU $\circ$ $AGT^{-1} \rightarrow AGT^{-1}$ adds 201 relations, which corresponds to 0.89% in relative terms. Its accuracy is low, 0.40. Other axioms are less productive but have a greater relative impact and accu-

No.	Axiom	no heuristic			with heuristic		
		No. Inst.	Acc.	Produc.	No. Inst.	Acc.	Produc.
1	CAU $\circ$ AGT^{-1} $\rightarrow$ AGT^{-1}	201	0.40	0.89%	75	0.67	0.33%
2	CAU $\circ$ AT-L $\rightarrow$ AT-L	17	0.82	0.84%	15	0.93	0.74%
3	CAU $\circ$ AT-T $\rightarrow$ AT-T	72	0.85	1.25%	69	0.87	1.20%
1–3	CAU $\circ$ R$_2$ $\rightarrow$ R$_3$	290	0.54	0.96%	159	0.78	0.52%
4	PRP $\circ$ AGT^{-1} $\rightarrow$ AGT^{-1}	375	0.89	1.66%	347	0.94	1.54%
5	PRP $\circ$ AT-L $\rightarrow$ AT-L	49	0.90	2.42%	48	0.92	2.37%
6	PRP $\circ$ AT-T $\rightarrow$ AT-T	138	0.84	2.40%	129	0.88	2.25%
7	PRP $\circ$ MNR^{-1} $\rightarrow$ MNR^{-1}	71	0.82	3.21%	70	0.83	3.16%
4–7	PRP $\circ$ R$_2$ $\rightarrow$ R$_3$	633	0.87	1.95%	594	0.91	1.83%
1–7	**All**	923	0.77	2.84%	**753**	**0.88**	**2.32%**

Table 6: Axioms used for evaluation, number of instances, accuracy and productivity (i.e., percentage of relations added on top the ones already present). Results are reported with and without the heuristic.

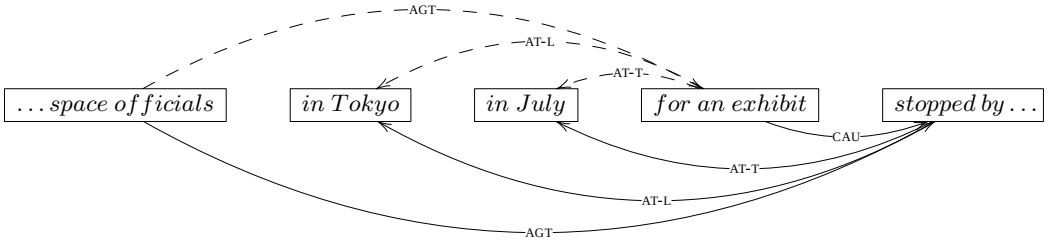


Figure 2: Basic (solid arrows) and inferred relations (discontinuous) from *A half-dozen Soviet space officials, in Tokyo in July for an exhibit, stopped by to see their counterparts at the National* ... (wsj_0405, 1).

racy. For example, axiom PRP $\circ$ MNR^{-1} $\rightarrow$ MNR^{-1}, only yields 71 new MNR, and yet it is adding 3.21% in relative terms with an accuracy of 0.82.

Overall, applying the seven axioms adds 923 relations on top of the ones already present (2.84% in relative terms) with an accuracy of 0.77. Figure 2 shows examples of inferences using axioms 1–3.

5.1 Error Analysis

Because of the low accuracy of axiom 1, an error analysis was performed. We found that unlike other axioms, this axiom often yield a relation type that is already present in the semantic representation. Specifically, it often yields R(*x, z*) when R(*x', z*) is already known. We use the following heuristic in order to improve accuracy: *do not instantiate an axiom* R$_1$*(x, y)* $\circ$ R$_2$*(y, z)* $\rightarrow$ R$_3$*(x, z) if a relation of the form* R$_3$*(x', z) is already known.*

This simple heuristic has increased the accuracy of the inferences at the cost of lowering their productivity. The last three columns in Table 6 show results when using the heuristic.

6 Comparison with Previous Work

There have been many proposals to detect semantic relations from text without composition. Researches have targeted particular relations (e.g., CAUSE (Chang and Choi, 2006; Bethard and Martin, 2008)), relations within noun phrases (Nulty, 2007), named entities (Hirano et al., 2007) or clauses (Szpakowicz et al., 1995). Competitions include (Litkowski, 2004; Carreras and Màrquez, 2005; Girju et al., 2007; Hendrickx et al., 2009).

Two recent efforts (Ruppenhofer et al., 2009; Gerber and Chai, 2010) are similar to CSR in their goal (i.e., extract meaning ignored by current semantic parsers), but completely differ in their means. Their merit relies on annotating and extracting semantic connections not originally contemplated (e.g., between concepts from two different sentences) using an already known and fixed relation set. Unlike CSR, they are dependent on the relation inventory, require annotation and do not reason or manipulate relations. In contrast to all the above references and the state of the art, the proposed framework obtains axioms that take as input semantic relations pro-

duced by others and output more relations: it adds an extra layer of semantics previously ignored.

Previous research has exploited the idea of using semantic primitives to define and classify semantic relations under the names of *relation elements*, *deep structure*, *aspects* and *primitives*. The first attempt on describing semantic relations using primitives was made by Chaffin and Herrmann (1987); they differentiate 31 relations using 30 *relation elements* clustered into five groups (intensional force, dimension, agreement, propositional and part-whole inclusion). Winston et al. (1987) introduce 3 *relation elements* (functional, homeomerous and separable) to distinguish six subtypes of PART-WHOLE. Cohen and Losielle (1988) use the notion of *deep structure* in contrast to the *surface relation* and utilizes two *aspects* (hierarchical and temporal). Huhns and Stephens (1989) consider a set of 10 *primitives*.

In theoretical linguistics, Wierzbicka (1996) introduced the notion of semantic primes to perform linguistic analysis. Dowty (2006) studies compositionality and identifies entailments associated with certain predicates and arguments (Dowty, 2001).

There has not been much work on composing relations in the field of computational linguistics. The term *compositional semantics* is used in conjunction with the principle of compositionality, i.e., the meaning of a complex expression is determined from the meanings of its parts, and the way in which those parts are combined. These approaches are usually formal and use a potentially infinite set of predicates to represent semantics. Ge and Mooney (2009) extracts semantic representations using syntactic structures while Copestake et al. (2001) develops algebras for semantic construction within grammars. Logic approaches include (Lakoff, 1970; Sánchez Valencia, 1991; MacCartney and Manning, 2009). Composition of Semantic Relations is complimentary to Compositional Semantics.

Previous research has manually extracted plausible inference axioms for WordNet relations (Harabagiu and Moldovan, 1998) and transformed chains of relations into theoretical axioms (Helbig, 2005). The CSR algorithm proposed here automatically obtains inference axioms.

Composing relations has been proposed before within knowledge bases. Cohen and Losielle (1988) combines a set of nine fairly specific relations (e.g., FOCUS-OF, PRODUCT-OF, SETTING-OF). The key to determine plausibility is the *transitivity characteristic* of the aspects: two relations shall not combine if they have contradictory values for any aspect. The first algebra to compose semantic primitives was proposed by Huhns and Stephens (1989). Their relations are not linguistically motivated and ten of them map to some sort of PART-WHOLE (e.g. PIECE-OF, SUBREGION-OF). Unlike (Cohen and Losielle, 1988; Huhns and Stephens, 1989), we use typical relations that encode the semantics of natural language, propose a method to automatically obtain the inverse of a relation and empirically test the validity of the axioms obtained.

7 Conclusions

Going beyond current research, in this paper we investigate the composition of semantic relations. The proposed CSR algorithm obtains inference axioms that take as their input semantic relations and output a relation previously ignored. Regardless of the set of relations and annotation scheme, an additional layer of semantics is created on top of the already existing relations.

An extended definition for semantic relations is proposed, including restrictions on their domains and ranges as well as values for semantic primitives. Primitives indicate if a certain property holds between the arguments of a relation. An algebra for composing semantic primitives is defined, allowing to automatically determine the primitives values for the composition of any two relations.

The CSR algorithm makes use of the extended definition and algebra to discover inference axioms in an unsupervised manner. Its usefulness is shown using a set of eight common relations, obtaining 78 axioms. Empirical evaluation shows the axioms add 2.32% of relations in relative terms with an overall accuracy of 0.88, more than what state-of-the-art semantic parsers achieve.

The framework presented is completely independent of any particular set of relations. Even though different sets may call for different ontologies and primitives, we believe the model is generally applicable; the only requirement is to use the extended definition. This is a novel way of retrieving semantic relations in the field of computational linguistics.

References

Steven Bethard and James H. Martin. 2008. Learning Semantic Links from a Corpus of Parallel Temporal and Causal Relations. In *Proceedings of ACL-08: HLT, Short Papers*, pages 177–180, Columbus, Ohio.

Eduardo Blanco and Dan Moldovan. 2011. A Model for Composing Semantic Relations. In *Proceedings of the 9th International Conference on Computational Semantics (IWCS 2011)*, Oxford, UK.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: semantic role labeling. In *CONLL '05: Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 152–164, Morristown, NJ, USA.

Roger Chaffin and Douglass J. Herrmann, 1987. *Relation Element Theory: A New Account of the Representation and Processing of Semantic Relations*.

Du S. Chang and Key S. Choi. 2006. Incremental cue phrase learning and bootstrapping method for causality extraction using cue phrase and word pair probabilities. *Information Processing & Management*, 42(3):662–678.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *Proceedings of EMNLP 2004*, pages 33–40, Barcelona, Spain.

Paul R. Cohen and Cynthia L. Losielle. 1988. Beyond ISA: Structures for Plausible Inference in Semantic Networks. In *Proceedings of the Seventh National conference on Artificial Intelligence*, St. Paul, Minnesota.

Ann Copestake, Alex Lascarides, and Dan Flickinger. 2001. An Algebra for Semantic Construction in Constraint-based Grammars. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 140–147, Toulouse, France.

David D. Dowty. 2001. The Semantic Asymmetry of 'Argument Alternations' (and Why it Matters). In Geart van der Meer and Alice G. B. ter Meulen, editors, *Making Sense: From Lexeme to Discourse*, volume 44.

David Dowty. 2006. Compositionality as an Empirical Problem. In Chris Barker and Polly Jacobson, editors, *Papers from the Brown University Conference on Direct Compositionality*. Oxford University Press.

Ruifang Ge and Raymond Mooney. 2009. Learning a Compositional Semantic Parser using an Existing Syntactic Parser. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 611–619, Suntec, Singapore.

Matthew Gerber and Joyce Chai. 2010. Beyond NomBank: A Study of Implicit Arguments for Nominal Predicates. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1583–1592, Uppsala, Sweden.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. SemEval-2007 Task 04: Classification of Semantic Relations between Nominals. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 13–18, Prague, Czech Republic.

Sanda Harabagiu and Dan Moldovan. 1998. Knowledge Processing on an Extended WordNet. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database and Some of its Applications.*, chapter 17, pages 684–714. The MIT Press.

Hermann Helbig. 2005. *Knowledge Representation and the Semantics of Natural Language*. Springer, 1st edition.

Iris Hendrickx, Su N. Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations Between Pairs of Nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 94–99, Boulder, Colorado.

Toru Hirano, Yoshihiro Matsuo, and Genichiro Kikui. 2007. Detecting Semantic Relations between Named Entities in Text Using Contextual Features. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Demo and Poster Sessions*, pages 157–160, Prague, Czech Republic.

Michael N. Huhns and Larry M. Stephens. 1989. Plausible Inferencing Using Extended Composition. In *IJCAI'89: Proceedings of the 11th international joint conference on Artificial intelligence*, pages 1420–1425, San Francisco, CA, USA.

George Lakoff. 1970. Linguistics and Natural Logic. 22(1):151–271, December.

Ken Litkowski. 2004. Senseval-3 task: Automatic labeling of semantic roles. In *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 9–12, Barcelona, Spain.

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *Proceedings of the Eight International Conference on Computational Semantics*, pages 140–156, Tilburg, The Netherlands.

Paul Nulty. 2007. Semantic Classification of Noun Phrases Using Web Counts and Learning Algorithms. In *Proceedings of the ACL 2007 Student Research Workshop*, pages 79–84, Prague, Czech Republic.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

Josef Ruppenhofer, Caroline Sporleder, Roser Morante, Collin Baker, and Martha Palmer. 2009. SemEval-2010 Task 10: Linking Events and Their Participants in Discourse. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 106–111, Boulder, Colorado.

Victor Sánchez Valencia. 1991. *Studies on Natural Logic and Categorial Grammar*. Ph.D. thesis, University of Amsterdam.

Barker Szpakowicz, Ken Barker, and Stan Szpakowicz. 1995. Interactive semantic analysis of Clause-Level Relationships. In *Proceedings of the Second Conference of the Pacific Association for Computational Linguistics*, pages 22–30.

Anna Wierzbicka. 1996. *Semantics: Primes and Universals*. Oxford University Press, USA.

Morton E. Winston, Roger Chaffin, and Douglas Herrmann. 1987. A Taxonomy of Part-Whole Relations. *Cognitive Science*, 11(4):417–444.

Unsupervised Discovery of Domain-Specific Knowledge from Text

Dirk Hovy, Chunliang Zhang, Eduard Hovy
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
{dirkh, czheng, hovy}@isi.edu

Anselmo Peñas
UNED NLP and IR Group
Juan del Rosal 16
28040 Madrid, Spain
anselmo@lsi.uned.es

Abstract

Learning by Reading (LbR) aims at enabling machines to acquire knowledge from and reason about textual input. This requires knowledge about the domain structure (such as entities, classes, and actions) in order to do inference. We present a method to infer this implicit knowledge from unlabeled text. Unlike previous approaches, we use automatically extracted classes with a probability distribution over entities to allow for context-sensitive labeling. From a corpus of 1.4m sentences, we learn about 250k simple propositions about American football in the form of predicate-argument structures like "quarterbacks throw passes to receivers". Using several statistical measures, we show that our model is able to generalize and explain the data statistically significantly better than various baseline approaches. Human subjects judged up to 96.6% of the resulting propositions to be sensible. The classes and probabilistic model can be used in textual enrichment to improve the performance of LbR end-to-end systems.

1 Introduction

The goal of Learning by Reading (LbR) is to enable a computer to learn about a new domain and then to reason about it in order to perform such tasks as question answering, threat assessment, and explanation (Strassel et al., 2010). This requires joint efforts from Information Extraction, Knowledge Representation, and logical inference. All these steps depend on the system having access to basic, often unstated, foundational knowledge about the domain.

Most documents, however, do not explicitly mention this information in the text, but assume basic background knowledge about the domain, such as positions ("quarterback"), titles ("winner"), or actions ("throw") for sports game reports. Without this knowledge, the text will not make sense to the reader, despite being well-formed English. Luckily, the information is often implicitly contained in the document or can be inferred from similar texts.

Our system automatically acquires domain-specific knowledge (classes and actions) from large amounts of unlabeled data, and trains a probabilistic model to determine and apply the most informative classes (*quarterback*, etc.) at appropriate levels of generality for unseen data. E.g., from sentences such as "Steve Young threw a pass to Michael Holt", "Quarterback Steve Young finished strong", and "Michael Holt, the receiver, left early" we can learn the classes *quarterback* and *receiver*, and the proposition "*quarterbacks throw passes to receivers*".

We will thus assume that the implicit knowledge comes in two forms: actions in the form of predicate-argument structures, and classes as part of the source data. Our task is to identify and extract both. Since LbR systems must quickly adapt and scale well to new domains, we need to be able to work with large amounts of data and minimal supervision. Our approach produces simple propositions about the domain (see Figure 1 for examples of actual propositions learned by our system).

American football was the first official evaluation domain in the DARPA-sponsored Machine Reading program, and provides the background for a number

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1466–1475,
Portland, Oregon, June 19-24, 2011.

of LbR systems (Mulkar-Mehta et al., 2010). Sports is particularly amenable, since it usually follows a finite, explicit set of rules. Due to its popularity, results are easy to evaluate with lay subjects, and game reports, databases, etc. provide a large amount of data. The same need for basic knowledge appears in all domains, though. In music, *musicians* play *instruments*, in electronics, *components* constitute *circuits*, *circuits* use *electricity*, etc.

Teams beat teams
Teams play teams
Quarterbacks throw passes
Teams win games
Teams defeat teams
Receivers catch passes
Quarterbacks complete passes
Quarterbacks throw passes to receivers
Teams play games
Teams lose games

Figure 1: The ten most frequent propositions discovered by our system for the American football domain

Our approach differs from verb-argument identification or Named Entity (NE) tagging in several respects. While previous work on verb-argument selection (Pardo et al., 2006; Fan et al., 2010) uses fixed sets of classes, we cannot know a priori how many and which classes we will encounter. We therefore provide a way to derive the appropriate classes automatically and include a probability distribution for each of them. Our approach is thus less restricted and can learn context-dependent, fine-grained, domain-specific propositions. While a NE-tagged corpus could produce a general proposition like "*PERSON* throws to *PERSON*", our method enables us to distinguish the arguments and learn "*quarterback* throws to *receiver*" for American football and "*outfielder* throws to *third base*" for baseball. While in NE tagging each word has only one correct tag in a given context, we have hierarchical classes: an entity can be correctly labeled as a *player* or a *quarterback* (and possibly many more classes), depending on the context. By taking context into account, we are also able to label each sentence individually and account for unseen entities without using external resources.

Our contributions are:

- we use unsupervised learning to train a model that makes use of automatically extracted classes to uncover implicit knowledge in the form of predicate-argument propositions
- we evaluate the explanatory power, generalization capability, and sensibility of the propositions using both statistical measures and human judges, and compare them to several baselines
- we provide a model and a set of propositions that can be used to improve the performance of end-to-end LbR systems via textual enrichment.

2 Methods

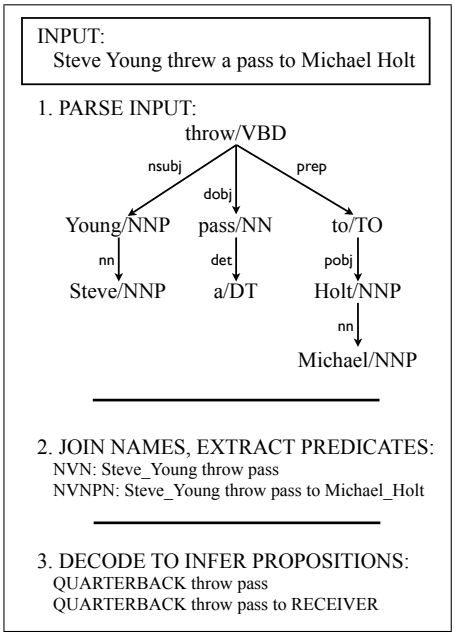


Figure 2: Illustrated example of different processing steps

Our running example will be "Steve Young threw a pass to Michael Holt". This is an instance of the underlying proposition "*quarterbacks throw passes to receivers*", which is not explicitly stated in the data. A proposition is thus a more general statement about the domain than the sentences it derives. It contains domain-specific classes (*quarterback*, *receiver*), as well as lexical items ("throw", "pass"). In order to reproduce the proposition, given the input sentences, our system has to not only identify the classes, but also learn when to

abstract away from the lexical form to the appropriate class and when to keep it (cf. Figure 2, step 3). To facilitate extraction, we focus on propositions with the following predicate-argument structures: NOUN-VERB-NOUN (e.g., "*quarterbacks throw passes*"), or NOUN-VERB-NOUN-PREPOSITION-NOUN (e.g., "*quarterbacks throw passes to receivers*". There is nothing, though, that prevents the use of other types of structures as well. We do not restrict the verbs we consider (Pardo et al., 2006; Ritter et al., 2010)), which extracts a high number of hapax structures.

Given a sentence, we want to find the most likely class for each word and thereby derive the most likely proposition. Similar to Pardo et al. (2006), we assume the observed data was produced by a process that generates the proposition and then transforms the classes into a sentence, possibly adding additional words. We model this as a Hidden Markov Model (HMM) with bigram transitions (see Section 2.3) and use the EM algorithm (Dempster et al., 1977) to train it on the observed data, with smoothing to prevent overfitting.

2.1 Data

We use a corpus of about 33k texts on American football, extracted from the New York Times (Sandhaus, 2008). To identify the articles, we rely on the provided "football" keyword classifier. The resulting corpus comprises $1,359,709$ sentences from game reports, background stories, and opinion pieces. In a first step, we parse all documents with the Stanford dependency parser (De Marneffe et al., 2006) (see Figure 2, step 1). The output is lemmatized (collapsing "*throws*", "*threw*", etc., into "*throw*"), and marked for various dependencies (*nsubj*, *amod*, etc.). This enables us to extract the predicate argument structure, like subject-verb-object, or additional prepositional phrases (see Figure 2, step 2). These structures help to simplify the model by discarding additional words like modifiers, determiners, etc., which are not essential to the proposition. The same approach is used by (Brody, 2007). We also concatenate multi-word names (identified by sequences of NNPs) with an underscore to form a single token ("Steve/NNP Young/NNP" → "Steve_Young").

2.2 Deriving Classes

To derive the classes used for entities, we do not restrict ourselves to a fixed set, but derive a domain-specific set directly from the data. This step is performed simultaneously with the corpus generation described above. We utilize three syntactic constructions to identify classes, namely *nominal modifiers*, *copula verbs*, and *appositions*, see below. This is similar in nature to Hearst's lexico-syntactic patterns (Hearst, 1992) and other approaches that derive *IS-A* relations from text. While we find it straightforward to collect classes for entities in this way, we did not find similar patterns for verbs. Given a suitable mechanism, however, these could be incorporated into our framework as well.

Nominal modifier are common nouns (labeled NN) that precede proper nouns (labeled NNP), as in "***quarterback/NN*** *Steve/NNP Young/NNP*", where "quarterback" is the nominal modifier of "Steve Young". Similar information can be gained from appositions (e.g., "*Steve Young,* ***the quarterback of his team****, said...*"), and copula verbs ("*Steve Young* ***is the quarterback*** *of the 49ers*"). We extract those co-occurrences and store the proper nouns as entities and the common nouns as their possible classes. For each pair of class and entity, we collect counts over the corpus to derive probability distributions.

Entities for which we do not find any of the above patterns in our corpus are marked "*UNK*". These entities are instantiated with the 20 most frequent classes. All other (non-entity) words (including verbs) have only their identity as class (i.e., "*pass*" remains "*pass*").

The average number of classes per entity is 6.87. The total number of distinct classes for entities is $63,942$. This is a huge number to model in our state space.[1] Instead of manually choosing a subset of the classes we extracted, we defer the task of finding the best set to the model.

We note, however, that the distribution of classes for each entity is highly skewed. Due to the unsupervised nature of the extraction process, many of the extracted classes are hapaxes and/or random noise. Most entities have only a small number of applicable classes (a football player usually has one main posi-

[1] NE taggers usually use a set of only a few dozen classes at most.

tion, and a few additional roles, such as *star*, *teammate*, etc.). We handle this by limiting the number of classes considered to 3 per entity. This constraint reduces the total number of distinct classes to $26,165$, and the average number of classes per entity to 2.53. The reduction makes for a more tractable model size without losing too much information. The class alphabet is still several magnitudes larger than that for NE or POS tagging. Alternatively, one could use external resources such as Wikipedia, Yago (Suchanek et al., 2007), or WordNet++ (Ponzetto and Navigli, 2010) to select the most appropriate classes for each entity. This is likely to have a positive effect on the quality of the applicable classes and merits further research. Here, we focus on the possibilities of a self-contained system without recurrence to outside resources.

The number of classes we consider for each entity also influences the number of possible propositions: if we consider exactly one class per entity, there will be little overlap between sentences, and thus no generalization possible—the model will produce many distinct propositions. If, on the other hand, we used only one class for all entities, there will be similarities between many sentences—the model will produce very few distinct propositions.

2.3 Probabilistic Model

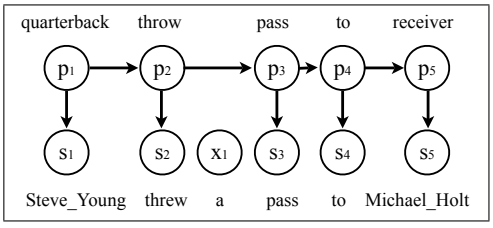


Figure 3: Graphical model for the running example

We use a generative noisy-channel model to capture the joint probability of input sentences and their underlying proposition. Our generative story of how a sentence $\mathbf{s}$ (with words $s_1, ..., s_n$) was generated assumes that a proposition $\mathbf{p}$ is generated as a sequence of classes $p_1, ..., p_n$, with transition probabilities $P(p_i|p_{i-1})$. Each class p_i generates a word s_i with probability $P(s_i|p_i)$. We allow additional words x in the sentence which do not depend on any class in the proposition and are thus generated independently with $P(x)$ (cf. model in Figure 3).

Since we observe the co-occurrence counts of classes and entities in the data, we can fix the emission parameter $P(s|p)$ in our HMM. Further, we do not want to generate sentences from propositions, so we can omit the step that adds the additional words x in our model. The removal of these words is reflected by the preprocessing step that extracts the structure (cf. Section 2.1).

Our model is thus defined as

$$P(\mathbf{s}, \mathbf{p}) = P(p_1) \cdot \prod_{i=1}^{n} \Big(P(p_i|p_{i-1}) \cdot P(s_i|p_i) \Big) \quad (1)$$

where s_i, p_i denote the i^{th} word of sentence $\mathbf{s}$ and proposition $\mathbf{p}$, respectively.

3 Evaluation

We want to evaluate how well our model predicts the data, and how sensible the resulting propositions are. We define a good model as one that generalizes well and produces semantically useful propositions.

We encounter two problems. First, since we derive the classes in a data-driven way, we have no gold standard data available for comparison. Second, there is no accepted evaluation measure for this kind of task. Ultimately, we would like to evaluate our model externally, such as measuring its impact on performance of a LbR system. In the absence thereof, we resort to several complementary measures, as well as performing an annotation task. We derive evaluation criteria as follows. A model generalizes well if it can cover ('explain') all the sentences in the corpus with a few propositions. This requires a measure of generality. However, while a proposition such as "*PERSON* does *THING*", has excellent generality, it possesses no discriminating power. We also need the propositions to partition the sentences into clusters of semantic similarity, to support effective inference. This requires a measure of distribution. Maximal distribution, achieved by assigning every sentence to a different proposition, however, is not useful either. We need to find an appropriate level of generality within which the sentences are clustered into propositions for the best overall groupings to support inference.

To assess the learned model, we apply the measures of *generalization*, *entropy*, and *perplexity* (see

Sections 3.2, 3.3, and 3.4). These measures can be used to compare different systems. We do not attempt to weight or combine the different measures, but present each in its own right.

Further, to assess label accuracy, we use Amazon's Mechanical Turk annotators to judge the sensibility of the propositions produced by each system (Section 3.5). We reason that if our system learned to infer the correct classes, then the resulting propositions should constitute true, general statements about that domain, and thus be judged as sensible.[2] This approach allows the effective annotation of sufficient amounts of data for an evaluation (first described for NLP in (Snow et al., 2008)).

3.1 Evaluation Data

With the trained model, we use Viterbi decoding to extract the best class sequence for each example in the data. This translates the original corpus sentences into propositions. See steps 2 and 3 in Figure 2.

We create two baseline systems from the same corpus, one which uses the most frequent class (MFC) for each entity, and another one which uses a class picked at random from the applicable classes of each entity.

Ultimately, we are interested in labeling unseen data from the same domain with the correct class, so we evaluate separately on the full corpus and the subset of sentences that contain unknown entities (i.e., entities for which no class information was available in the corpus, cf. Section 2.2).

For the latter case, we select all examples containing at least one unknown entity (labeled UNK), resulting in a subset of $41,897$ sentences, and repeat the evaluation steps described above. Here, we have to consider a much larger set of possible classes per entity (the 20 overall most frequent classes). The MFC baseline for these cases is the most frequent of the 20 classes for UNK tokens, while the random baseline chooses randomly from that set.

3.2 Generalization

Generalization measures how widely applicable the produced propositions are. A completely lexical approach, at one extreme, would turn each sentence into a separate proposition, thus achieving a generalization of 0%. At the other extreme, a model that produces only one proposition would generalize extremely well (but would fail to explain the data in any meaningful way). Both are of course not desirable.

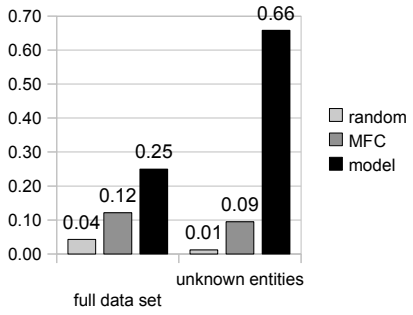


Figure 4: Generalization of models on the data sets

We define generalization as

$$g = 1 - \frac{|propositions|}{|sentences|} \quad (2)$$

The results in Figure 4 show that our model is capable of abstracting away from the lexical form, achieving a generalization rate of 25% for the full data set. The baseline approaches do significantly worse, since they are unable to detect similarities between lexically different examples, and thus create more propositions. Using a two-tailed t-test, the difference between our model and each baseline is statistically significant at $p < .001$.

Generalization on the unknown entity data set is even higher (65.84%). The difference between the model and the baselines is again statistically significant at $p < .001$. MFC always chooses the same class for UNK, regardless of context, and performs much worse. The random baseline chooses between 20 classes per entity instead of 3, and is thus even less general.

3.3 Normalized Entropy

Entropy is used in information theory to measure how predictable data is. 0 means the data is completely predictable. The higher the entropy of our propositions, the less well they explain the data. We are looking for models with low entropy. The extreme case of only one proposition has 0 entropy:

[2] Unfortunately, if judged insensible, we can not infer whether our model used the wrong class despite better options, or whether we simply have not learned the correct label.

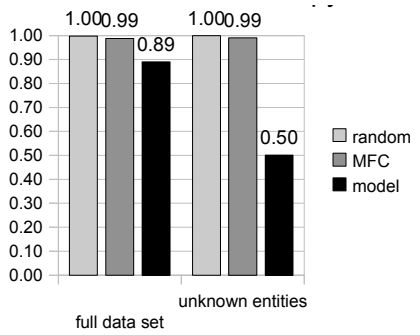


Figure 5: Entropy of models on the data sets

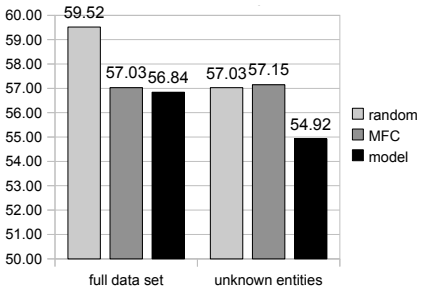


Figure 6: Perplexity of models on the data sets

we know exactly which sentences are produced by the proposition.

Entropy is directly influenced by the number of propositions used by a system.[3] In order to compare different models, we thus define normalized entropy as

$$H_N = \frac{-\sum_{i=0}^{n} P_i \cdot \log P_i}{\log n} \quad (3)$$

where P_i is the coverage of the proposition, or the percentage of sentences explained by it, and n is the number of distinct propositions.

The entropy of our model on the full data set is relatively high with 0.89, see Figure 5. The best entropy we can hope to achieve given the number of propositions and sentences is actually 0.80 (by concentrating the maximum probability mass in one proposition). The model thus does not perform as badly as the number might suggest. The entropy of our model on unseen data is better, with 0.50 (best possible: 0.41). This might be due to the fact that we considered more classes for UNK than for regular entities.

3.4 Perplexity

Since we assume that propositions are valid sentences in our domain, good propositions should have a higher probability than bad propositions in a language model. We can compute this using perplexity:[4]

$$perplexity(data) = 2^{\frac{-\log P(data)}{n}} \quad (4)$$

where $P(data)$ is the product of the proposition probabilities, and n is the number of propositions. We use the uni-, bi-, and trigram counts of the GoogleGrams corpus (Brants and Franz, 2006) with simple interpolation to compute the probability of each proposition.

The results in Figure 6 indicate that the propositions found by the model are preferable to the ones found by the baselines. As would be expected, the sensibility judgements for MFC and model[5] (Tables 1 and 2, Section 3.5) are perfectly anti-correlated (correlation coefficient -1) with the perplexity for these systems in each data set. However, due to the small sample size, this should be interpreted cautiously.

3.5 Sensibility and Label Accuracy

In unsupervised training, the model with the best data likelihood does not necessarily produce the best label accuracy. We evaluate label accuracy by presenting subjects with the propositions we obtained from the Viterbi decoding of the corpus, and ask them to rate their sensibility. We compare the different systems by computing sensibility as the percentage of propositions judged sensible for each system. Since the underlying probability distributions are quite different, we weight the sensibility judgement for each proposition by the likelihood of that proposition. We report results for both aggregate

[3] Note that how many classes we consider per entity influences how many propositions are produced (cf. Section 2.2), and thus indirectly puts a bound on entropy.

[4] Perplexity also quantifies the uncertainty of the resulting propositions, where 0 perplexity means no uncertainty.

[5] We did not collect sensibility judgements for the random baseline.

		100 most frequent		random		combined	
Data set	**System**	**agg**	**maj**	**agg**	**maj**	**agg**	**maj**
full	baseline	90.16	92.13	69.35	**70.57**	88.84	90.37
	model	**94.28**	**96.55**	**70.93**	70.45	**93.06**	**95.16**

Table 1: Percentage of propositions derived from labeling the full data set that were judged sensible

		100 most frequent		random		combined	
Data set	**System**	**agg**	**maj**	**agg**	**maj**	**agg**	**maj**
unknown	baseline	51.92	51.51	32.39	28.21	50.39	49.66
	model	**66.00**	**69.57**	**48.14**	**41.74**	**64.83**	**67.76**

Table 2: Percentage of propositions derived from labeling unknown entities that were judged sensible

sensibility (using the total number of individual answers), and majority sensibility, where each proposition is scored according to the majority of annotators' decisions.

The model and baseline propositions for the full data set are both judged highly sensible, achieving accuracies of 96.6% and 92.1% (cf. Table 1). While our model did slightly better, the differences are not statistically significant when using a two-tailed test. The propositions produced by the model from unknown entities are less sensible (67.8%), albeit still significantly above chance level, and the baseline propositions for the same data set ($p < 0.01$). Only 49.7% propositions of the baseline were judged sensible (cf. Table 2).

3.5.1 Annotation Task

Our model finds $250,169$ distinct propositions, the MFC baseline $293,028$. We thus have to restrict ourselves to a subset in order to judge their sensibility. For each system, we sample the 100 most frequent propositions and 100 random propositions found for both the full data set and the unknown entities[6] and have 10 annotators rate each proposition as sensible or insensible. To identify and omit bad annotators ('spammers'), we use the method described in Section 3.5.2, and measure inter-annotator agreement as described in Section 3.5.3. The details of this evaluation are given below, the results can be found in Tables 1 and 2.

The 200 propositions from each of the four systems (model and baseline on both full and unknown data set), contain 696 distinct propositions. We break these up into 70 batches (Amazon Turk annotation HIT pages) of ten propositions each. For each proposition, we request 10 annotators. Overall, 148 different annotators participated in our annotation. The annotators are asked to state whether each proposition represents a sensible statement about American Football or not. A proposition like "*Quarterbacks can throw passes to receivers*" should make sense, while "*Coaches can intercept teams*" does not. To ensure that annotators judge sensibility and not grammaticality, we format each proposition the same way, namely pluralizing the nouns and adding "can" before the verb. In addition, annotators can state whether a proposition sounds odd, seems ungrammatical, is a valid sentence, but against the rules (e.g., "*Coaches can hit players*") or whether they do not understand it.

3.5.2 Spammers

Some (albeit few) annotators on Mechanical Turk try to complete tasks as quickly as possible without paying attention to the actual requirements, introducing noise into the data. We have to identify these spammers before the evaluation. One way is to include tests. Annotators that fail these tests will be excluded. We use a repetition (first and last question are the same), and a truism (annotators answering "no" either do not know about football or just answered randomly). Alternatively, we can assume that good annotators, who are the majority, will exhibit similar behavior to one another, while spam-

[6] We omit the random baseline here due to size issues, and because it is not likely to produce any informative comparison.

mers exhibit a deviant answer pattern. To identify those outliers, we compare each annotator's agreement to the others and exclude those whose agreement falls more than one standard deviation below the average overall agreement.

We find that both methods produce similar results. The first method requires more careful planning, and the resulting set of annotators still has to be checked for outliers. The second method has the advantage that it requires no additional questions. It includes the risk, though, that one selects a set of bad annotators solely because they agree with one another.

3.5.3 Agreement

measure	100 most frequent	random	combined
agreement	0.88	0.76	0.82
κ	0.45	0.50	0.48
G-index	0.66	0.53	0.58

Table 3: Agreement measures for different samples

We use inter-annotator agreement to quantify the reliability of the judgments. Apart from the simple agreement measure, which records how often annotators choose the same value for an item, there are several statistics that qualify this measure by adjusting for other factors. One frequently used measure, Cohen's κ, has the disadvantage that if there is prevalence of one answer, κ will be low (or even negative), despite high agreement (Feinstein and Cicchetti, 1990). This phenomenon, known as the κ paradox, is a result of the formula's adjustment for chance agreement. As shown by Gwet (2008), the true level of actual chance agreement is realistically not as high as computed, resulting in the counterintuitive results. We include it for comparative reasons. Another statistic, the G-index (Holley and Guilford, 1964), avoids the paradox. It assumes that expected agreement is a function of the number of choices rather than chance. It uses the same general formula as κ,

$$\frac{(P_a - P_e)}{(1 - P_e)} \tag{5}$$

where P_a is the actual raw agreement measured, and P_e is the expected agreement. The difference with κ is that P_e for the G-index is defined as $P_e = 1/q$, where q is the number of available categories, instead of expected chance agreement. Under most conditions, G and κ are equivalent, but in the case of high raw agreement and few categories, G gives a more accurate estimation of the agreement. We thus report raw agreement, κ, and G-index.

Despite early spammer detection, there are still outliers in the final data, which have to be accounted for when calculating agreement. We take the same approach as in the statistical spammer detection and delete outliers that are more than one standard deviation below the rest of the annotators' average.

The raw agreement for both samples combined is 0.82, $G = 0.58$, and $\kappa = 0.48$. The numbers show that there is reasonably high agreement on the label accuracy.

4 Related Research

The approach we describe is similar in nature to unsupervised verb argument selection/selectional preferences and semantic role labeling, yet goes beyond it in several ways. For semantic role labeling (Gildea and Jurafsky, 2002; Fleischman et al., 2003), classes have been derived from FrameNet (Baker et al., 1998). For verb argument detection, classes are either semi-manually derived from a repository like WordNet, or from NE taggers (Pardo et al., 2006; Fan et al., 2010). This allows for domain-independent systems, but limits the approach to a fixed set of oftentimes rather inappropriate classes. In contrast, we derive the level of granularity directly from the data.

Pre-tagging the data with NE classes before training comes at a cost. It lumps entities together which can have very different classes (i.e., all people become labeled as PERSON), effectively allowing only one class per entity. Etzioni et al. (2005) resolve the problem with a web-based approach that learns hierarchies of the NE classes in an unsupervised manner. We do not enforce a taxonomy, but include statistical knowledge about the distribution of possible classes over each entity by incorporating a prior distribution $P(class, entity)$. This enables us to generalize from the lexical form without restricting ourselves to one class per entity, which helps to better fit the data. In addition, we can distinguish several classes for each entity, depending on the context

(e.g., *winner* vs. *quarterback*). Ritter et al. (2010) also use an unsupervised model to derive selectional predicates from unlabeled text. They do not assign classes altogether, but group similar predicates and arguments into unlabeled clusters using LDA. Brody (2007) uses a very similar methodology to establish relations between clauses and sentences, by clustering simplified propositions.

Peñas and Hovy (2010) employ syntactic patterns to derive classes from unlabeled data in the context of LbR. They consider a wider range of syntactic structures, but do not include a probabilistic model to label new data.

5 Conclusion

We use an unsupervised model to infer domain-specific classes from a corpus of 1.4m unlabeled sentences, and applied them to learn 250k propositions about American football. Unlike previous approaches, we use automatically extracted classes with a probability distribution over entities to allow for context-sensitive selection of appropriate classes.

We evaluate both the model qualities and sensibility of the resulting propositions. Several measures show that the model has good explanatory power and generalizes well, significantly outperforming two baseline approaches, especially where the possible classes of an entity can only be inferred from the context.

Human subjects on Amazon's Mechanical Turk judged up to 96.6% of the propositions for the full data set, and 67.8% for data containing unseen entities as sensible. Inter-annotator agreement was reasonably high ($agreement = 0.82$, $G = 0.58$, $\kappa = 0.48$).

The probabilistic model and the extracted propositions can be used to enrich texts and support post-parsing inference for question answering. We are currently applying our method to other domains.

Acknowledgements

We would like to thank David Chiang, Victoria Fossum, Daniel Marcu, and Stephen Tratz, as well as the anonymous ACL reviewers for comments and suggestions to improve the paper. Research supported in part by Air Force Contract FA8750-09-C-0172 under the DARPA Machine Reading Program.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics Morristown, NJ, USA.

Thorsten Brants and Alex Franz, editors. 2006. *The Google Web 1T 5-gram Corpus Version 1.1*. Number LDC2006T13. Linguistic Data Consortium, Philadelphia.

Samuel Brody. 2007. Clustering Clauses for High-Level Relation Detection: An Information-theoretic Approach. In *Annual Meeting-Association for Computational Linguistics*, volume 45, page 448.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC 2006*. Citeseer.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Oren Etzioni, Michael Cafarella, Doug. Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134.

James Fan, David Ferrucci, David Gondek, and Aditya Kalyanpur. 2010. Prismatic: Inducing knowledge from a large scale lexicalized relation resource. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 122–127, Los Angeles, California, June. Association for Computational Linguistics.

Alvan R. Feinstein and Domenic V. Cicchetti. 1990. High agreement but low kappa: I. the problems of two paradoxes. *Journal of Clinical Epidemiology*, 43(6):543–549.

Michael Fleischman, Namhee Kwon, and Eduard Hovy. 2003. Maximum entropy models for FrameNet classification. In *Proceedings of EMNLP*, volume 3.

Danies Gildea and Dan Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Kilem Li Gwet. 2008. Computing inter-rater reliability and its variance in the presence of high agreement. *British Journal of Mathematical and Statistical Psychology*, 61(1):29–48.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.

Jasper Wilson Holley and Joy Paul Guilford. 1964. A Note on the G-Index of Agreement. *Educational and Psychological Measurement*, 24(4):749.

Rutu Mulkar-Mehta, James Allen, Jerry Hobbs, Eduard Hovy, Bernardo Magnini, and Christopher Manning, editors. 2010. *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*. Association for Computational Linguistics, Los Angeles, California, June.

Thiago Pardo, Daniel Marcu, and Maria Nunes. 2006. Unsupervised Learning of Verb Argument Structures. *Computational Linguistics and Intelligent Text Processing*, pages 59–70.

Anselmo Peñas and Eduard Hovy. 2010. Semantic enrichment of text with background knowledge. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 15–23, Los Angeles, California, June. Association for Computational Linguistics.

Simone Paolo Ponzetto and Roberto Navigli. 2010. Knowledge-rich Word Sense Disambiguation rivaling supervised systems. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1522–1531. Association for Computational Linguistics.

Alan Ritter, Mausam, and Oren Etzioni. 2010. A latent dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 424–434, Uppsala, Sweden, July. Association for Computational Linguistics.

Evan Sandhaus, editor. 2008. *The New York Times Annotated Corpus*. Number LDC2008T19. Linguistic Data Consortium, Philadelphia.

Rion Snow, Brendan O'Connor, Dan Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast—but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263. Association for Computational Linguistics.

Stephanie Strassel, Dan Adams, Henry Goldberg, Jonathan Herr, Ron Keesing, Daniel Oblinger, Heather Simpson, Robert Schrag, and Jonathan Wright. 2010. The DARPA Machine Reading Program-Encouraging Linguistic and Reasoning Research with a Series of Reading Tasks. In *Proceedings of LREC 2010*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.

Latent Semantic Word Sense Induction and Disambiguation

Tim Van de Cruys
RCEAL
University of Cambridge
United Kingdom
tv234@cam.ac.uk

Marianna Apidianaki
Alpage, INRIA & Univ Paris Diderot
Sorbonne Paris Cité, UMRI-001
75013 Paris, France
marianna.apidianaki@inria.fr

Abstract

In this paper, we present a unified model for the automatic induction of word senses from text, and the subsequent disambiguation of particular word instances using the automatically extracted sense inventory. The induction step and the disambiguation step are based on the same principle: words and contexts are mapped to a limited number of topical dimensions in a latent semantic word space. The intuition is that a particular sense is associated with a particular topic, so that different senses can be discriminated through their association with particular topical dimensions; in a similar vein, a particular instance of a word can be disambiguated by determining its most important topical dimensions. The model is evaluated on the SEMEVAL-2010 word sense induction and disambiguation task, on which it reaches state-of-the-art results.

1 Introduction

Word sense induction (WSI) is the task of automatically identifying the senses of words in texts, without the need for handcrafted resources or manually annotated data. The manual construction of a sense inventory is a tedious and time-consuming job, and the result is highly dependent on the annotators and the domain at hand. By applying an automatic procedure, we are able to only extract the senses that are objectively present in a particular corpus, and it allows for the sense inventory to be straightforwardly adapted to a new domain.

Word sense disambiguation (WSD), on the other hand, is the closely related task of assigning a sense label to a particular instance of a word in context, using an existing sense inventory. The bulk of WSD algorithms up till now use pre-defined sense inventories (such as WordNet) that often contain fine-grained sense distinctions, which poses serious problems for computational semantic processing (Ide and Wilks, 2007). Moreover, most WSD algorithms take a supervised approach, which requires a significant amount of manually annotated training data.

The model presented here induces the senses of words in a fully unsupervised way, and subsequently uses the induced sense inventory for the unsupervised disambiguation of particular occurrences of words. The induction step and the disambiguation step are based on the same principle: words and contexts are mapped to a limited number of topical dimensions in a latent semantic word space. The key idea is that the model combines tight, synonym-like similarity (based on dependency relations) with broad, topical similarity (based on a large 'bag of words' context window). The intuition in this is that the dependency features can be disambiguated by the topical dimensions identified by the broad contextual features; in a similar vein, a particular instance of a word can be disambiguated by determining its most important topical dimensions (based on the instance's context words).

The paper is organized as follows. Section 2 presents some previous research on distributional similarity and word sense induction. Section 3 gives an overview of our method for word sense induction and disambiguation. Section 4 provides a quantitative evaluation and comparison to other algorithms in the framework of the SEMEVAL-2010 word sense

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1476–1485,
Portland, Oregon, June 19-24, 2011.

induction and disambiguation (WSI/WSD) task. The last section draws conclusions, and lays out a number of future research directions.

2 Previous Work

2.1 Distributional similarity

According to the distributional hypothesis of meaning (Harris, 1954), words that occur in similar contexts tend to be semantically similar. In the spirit of this by now well-known adage, numerous algorithms have sprouted up that try to capture the semantics of words by looking at their distribution in texts, and comparing those distributions in a vector space model.

One of the best known models in this respect is latent semantic analysis — LSA (Landauer and Dumais, 1997; Landauer et al., 1998). In LSA, a term-document matrix is created, that contains the frequency of each word in a particular document. This matrix is then decomposed into three other matrices with a mathematical factorization technique called singular value decomposition (SVD). The most important dimensions that come out of the SVD are said to represent latent semantic dimensions, according to which nouns and documents can be represented more efficiently. Our model also applies a factorization technique (albeit a different one) in order to find a reduced semantic space.

Context is a determining factor in the nature of the semantic similarity that is induced. A broad context window (e.g. a paragraph or document) yields broad, topical similarity, whereas a small context yields tight, synonym-like similarity. This has lead a number of researchers to use the dependency relations that a particular word takes part in as contextual features. One of the most important approaches is Lin (1998). An overview of dependency-based semantic space models is given in Padó and Lapata (2007).

2.2 Word sense induction

The following paragraphs provide a succinct overview of word sense induction research. A thorough survey on word sense disambiguation (including unsupervised induction algorithms) is presented in Navigli (2009).

Algorithms for word sense induction can roughly be divided into *local* and *global* ones. Local WSI algorithms extract the different senses of a word on a per-word basis, i.e. the different senses for each word are determined separately. They can be further subdivided into *context-clustering* algorithms and *graph-based* algorithms. In the context-clustering approach, context vectors are created for the different instances of a particular word, and those contexts are grouped into a number of clusters, representing the different senses of the word. The context vectors may be represented as first or second-order co-occurrences (i.e. the contexts of the target word are similar if the words they in turn co-occur with are similar). The first one to propose this idea of context-group discrimination was Schütze (1998), and many researchers followed a similar approach to sense induction (Purandare and Pedersen, 2004). In the graph-based approach, on the other hand, a co-occurrence graph is created, in which nodes represent words, and edges connect words that appear in the same context (dependency relation or context window). The senses of a word may then be discovered using graph clustering techniques (Widdows and Dorow, 2002), or algorithms such as HyperLex (Véronis, 2004) or Pagerank (Agirre et al., 2006). Finally, Bordag (2006) recently proposed an approach that uses word triplets to perform word sense induction. The underlying idea is the 'one sense per collocation' assumption, and co-occurrence triplets are clustered based on the words they have in common.

Global algorithms take an approach in which the different senses of a particular word are determined by comparing them to, and demarcating them from, the senses of other words in a full-blown word space model. The best known global approach is the one by Pantel and Lin (2002). They present a global clustering algorithm – coined clustering by committee (CBC) – that automatically discovers word senses from text. The key idea is to first discover a set of tight, unambiguous clusters, to which possibly ambiguous words can be assigned. Once a word has been assigned to a cluster, the features associated with that particular cluster are stripped off the word's vector. This way, less frequent senses of the word may be discovered.

Van de Cruys (2008) proposes a model for sense induction based on latent semantic dimensions. Using an extension of non-negative matrix factoriza-

tion, the model induces a latent semantic space according to which both dependency features and broad contextual features are classified. Using the latent space, the model is able to discriminate between different word senses. The model presented below is an extension of this approach: whereas the model described in Van de Cruys (2008) is only able to perform word sense induction, our model is capable of performing both word sense induction and disambiguation.

3 Methodology

3.1 Non-negative Matrix Factorization

Our model uses non-negative matrix factorization – NMF (Lee and Seung, 2000) in order to find latent dimensions. There are a number of reasons to prefer NMF over the better known singular value decomposition used in LSA. First of all, NMF allows us to minimize the Kullback-Leibler divergence as an objective function, whereas SVD minimizes the Euclidean distance. The Kullback-Leibler divergence is better suited for language phenomena. Minimizing the Euclidean distance requires normally distributed data, and language phenomena are typically not normally distributed. Secondly, the non-negative nature of the factorization ensures that only additive and no subtractive relations are allowed. This proves particularly useful for the extraction of semantic dimensions, so that the NMF model is able to extract much more clear-cut dimensions than an SVD model. And thirdly, the non-negative property allows the resulting model to be interpreted probabilistically, which is not straightforward with an SVD factorization.

The key idea is that a non-negative matrix $\mathbf{A}$ is factorized into two other non-negative matrices, $\mathbf{W}$ and $\mathbf{H}$

$$\mathbf{A}_{i\times j} \approx \mathbf{W}_{i\times k}\mathbf{H}_{k\times j} \tag{1}$$

where k is much smaller than i, j so that both instances and features are expressed in terms of a few components. Non-negative matrix factorization enforces the constraint that all three matrices must be non-negative, so all elements must be greater than or equal to zero.

Using the minimization of the Kullback-Leibler divergence as an objective function, we want to find the matrices $\mathbf{W}$ and $\mathbf{H}$ for which the Kullback-Leibler divergence between $\mathbf{A}$ and $\mathbf{WH}$ (the multiplication of $\mathbf{W}$ and $\mathbf{H}$) is the smallest. This factorization is carried out through the iterative application of update rules. Matrices $\mathbf{W}$ and $\mathbf{H}$ are randomly initialized, and the rules in 2 and 3 are iteratively applied – alternating between them. In each iteration, each vector is adequately normalized, so that all dimension values sum to 1.

$$\mathbf{H}_{a\mu} \leftarrow \mathbf{H}_{a\mu}\frac{\sum_i \mathbf{W}_{ia}\frac{\mathbf{A}_{i\mu}}{(\mathbf{WH})_{i\mu}}}{\sum_k \mathbf{W}_{ka}} \tag{2}$$

$$\mathbf{W}_{ia} \leftarrow \mathbf{W}_{ia}\frac{\sum_\mu \mathbf{H}_{a\mu}\frac{\mathbf{A}_{i\mu}}{(\mathbf{WH})_{i\mu}}}{\sum_v \mathbf{H}_{av}} \tag{3}$$

3.2 Word sense induction

Using an extension of non-negative matrix factorization, we are able to jointly induce latent factors for three different modes: words, their window-based ('bag of words') context words, and their dependency relations. Three matrices are constructed that capture the pairwise co-occurrence frequencies for the different modes. The first matrix contains co-occurrence frequencies of words cross-classified by dependency relations, the second matrix contains co-occurrence frequencies of words cross-classified by words that appear in the noun's context window, and the third matrix contains co-occurrence frequencies of dependency relations cross-classified by co-occurring context words. NMF is then applied to the three matrices and the separate factorizations are interleaved (i.e. the results of the former factorization are used to initialize the factorization of the next matrix). A graphical representation of the interleaved factorization algorithm is given in figure 1.

The procedure of the algorithm goes as follows. First, matrices $\mathbf{W}$, $\mathbf{H}$, $\mathbf{G}$, and $\mathbf{F}$ are randomly initialized. We then start our first iteration, and compute the update of matrix $\mathbf{W}$ (using equation 3). Matrix $\mathbf{W}$ is then copied to matrix $\mathbf{V}$, and the update of matrix $\mathbf{G}$ is computed (using equation 2). The transpose of matrix $\mathbf{G}$ is again copied to matrix $\mathbf{U}$, and the update of $\mathbf{F}$ is computed (again using equation 2). As a last step, matrix $\mathbf{F}$ is copied to matrix $\mathbf{H}$, and we restart the iteration loop until a stopping criterion (e.g. a maximum number of iterations, or no more significant change in objective function; we used the

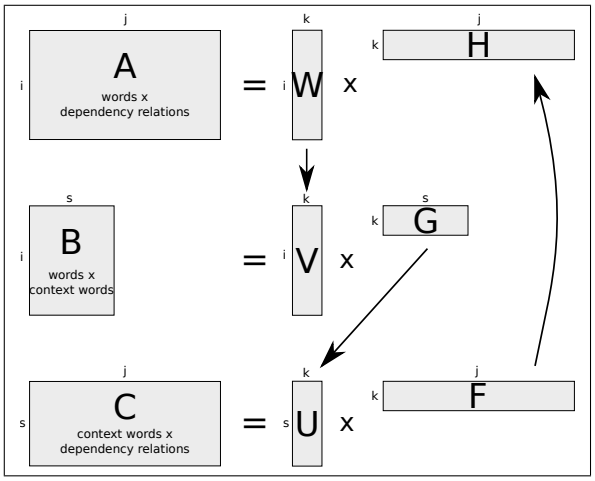


Figure 1: A graphical representation of the interleaved NMF algorithm

former one) is reached.[1] When the factorization is finished, the three different modes (words, window-based context words and dependency relations) are all represented according to a limited number of latent factors.

Next, the factorization that is thus created is used for word sense induction. The intuition is that a particular, dominant dimension of an ambiguous word is 'switched off', in order to reveal other possible senses of the word. Formally, we proceed as follows. Matrix $\mathbf{H}$ indicates the importance of each dependency relation given a topical dimension. With this knowledge, the dependency relations that are responsible for a certain dimension can be subtracted from the original noun vector. This is done by scaling down each feature of the original vector according to the load of the feature on the subtracted dimension, using equation 4.

$$\mathbf{t} = \mathbf{v}(\mathbf{u_1} - \mathbf{h_k}) \tag{4}$$

Equation 4 multiplies each dependency feature of the original noun vector $\mathbf{v}$ with a scaling factor, according to the load of the feature on the subtracted dimension ($\mathbf{h_k}$ – the vector of matrix $\mathbf{H}$ that corresponds to the dimension we want to subtract). $\mathbf{u_1}$ is a vector of ones with the same length as $\mathbf{h_k}$. The result is vector $\mathbf{t}$, in which the dependency features relevant to the particular topical dimension have been scaled down.

In order to determine which dimension(s) are responsible for a particular sense of the word, the method is embedded in a clustering approach. First, a specific word is assigned to its predominant sense (i.e. the most similar cluster). Next, the dominant semantic dimension(s) for this cluster are subtracted from the word vector, and the resulting vector is fed to the clustering algorithm again, to see if other word senses emerge. The dominant semantic dimension(s) can be identified by folding vector $\mathbf{c}$ – representing the cluster centroid – into the factorization (equation 5). This yields a probability vector $\mathbf{b}$ over latent factors for the particular centroid.

$$\mathbf{b} = \mathbf{c}\mathbf{H}^T \tag{5}$$

A simple k-means algorithm is used to compute the initial clustering, using the non-factorized dependency-based feature vectors (matrix $\mathbf{A}$). k-means yields a hard clustering, in which each noun is assigned to exactly one (dominant) cluster. In the second step, we determine for each noun whether it can be assigned to other, less dominant clusters. First, the salient dimension(s) of the centroid to which the noun is assigned are determined. The centroid of the cluster is computed by averaging the frequencies of all cluster elements except for the target word we want to reassign. After subtracting the salient dimensions from the noun vector, we check whether the vector is reassigned to another cluster centroid. If this is the case, (another instance of) the noun is assigned to the cluster, and the second step is repeated. If there is no reassignment, we continue with the next word. The target element is removed from the centroid to make sure that only the dimensions associated with the sense of the cluster are subtracted. When the algorithm is finished, each noun is assigned to a number of clusters, representing its different senses.

We use two different methods for selecting the final number of candidate senses. The first method, NMF$_{con}$, takes a conservative approach, and only selects candidate senses if – after the subtraction of salient dimensions – another sense is found that is more similar[2] to the adapted noun vector than the

[1] Note that this is not the only possibly way of interleaving the different factorizations, but in our experiments we found that different constellations lead to similar results.

[2] We use the cosine measure for our similarity calculations.

dominant sense. The second method, NMF_{lib}, is more liberal, and also selects the next best cluster centroid as candidate sense until a certain similarity threshold ϕ is reached.[3]

3.3 Word sense disambiguation

The sense inventory that results from the induction step can now be used for the disambiguation of individual instances as follows. For each instance of the target noun, we extract its context words, i.e. the words that co-occur in the same paragraph, and represent them as a probability vector $\mathbf{f}$. Using matrix $\mathbf{G}$ from our factorization model (which represents context words by semantic dimensions), this vector can be folded into the semantic space, thus representing a probability vector over latent factors for the particular instance of the target noun (equation 6).

$$\mathbf{d} = \mathbf{f}\mathbf{G}^T \tag{6}$$

Likewise, the candidate senses of the noun (represented as centroids) can be folded into our semantic space using matrix $\mathbf{H}$ (equation 5). This yields a probability distribution over the semantic dimensions for each centroid. As a last step, we compute the Kullback-Leibler divergence between the context vector and the candidate centroids, and select the candidate centroid that yields the lowest divergence as the correct sense. The disambiguation process is represented graphically in figure 2.

3.4 Example

Let us clarify the process with an example for the noun *chip*. The sense induction algorithm finds the following candidate senses:[4]

1. *cache*, *CPU*, *memory*, *microprocessor*, *processor*, *RAM*, *register*

2. *bread*, *cake*, *chocolate*, *cookie*, *recipe*, *sandwich*

3. *accessory*, *equipment*, *goods*, *item*, *machinery*, *material*, *product*, *supplies*

[3]Experimentally (examining the cluster output), we set $\phi = 0.2$

[4]Note that we do not use the word *sense* to hint at a lexicographic meaning distinction; rather, *sense* in this case should be regarded as a more coarse-grained and topic-related entity.

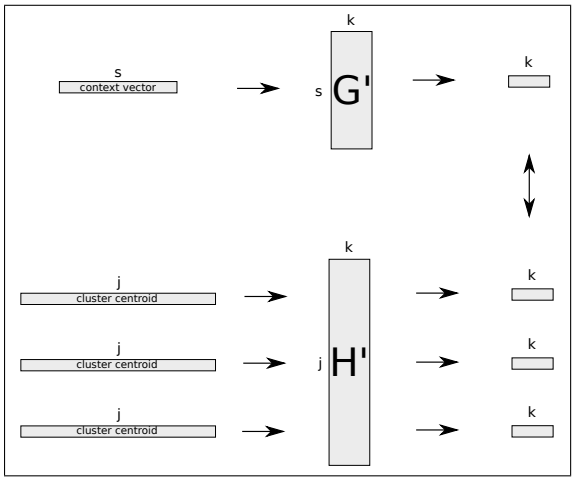


Figure 2: Graphical representation of the disambiguation process

Each candidate sense is associated with a centroid (the average frequency vector of the cluster's members), that is folded into the semantic space, which yields a 'semantic fingerprint', i.e. a distribution over the semantic dimensions. For the first sense, the 'computer' dimension will be the most important. Likewise, for the second and the third sense the 'food' dimension and the 'manufacturing' dimension will be the most important.[5]

Let us now take a particular instance of the noun *chip*, such as the one in (1).

(1) An N.V. Philips **unit** has **created** a **computer system** that **processes video images** 3,000 times faster than conventional **systems**. Using **reduced instruction** - **set computing**, or RISC, chips made by Intergraph of Huntsville, Ala., the **system** splits the **image** it '**sees**' into 20 **digital representations**, each **processed** by one *chip*.

Looking at the context of the particular instance of *chip*, a context vector is created which represents the semantic content words that appear in the same paragraph (the extracted content words are printed in boldface). This context vector is again folded into the semantic space, yielding a distribution over the semantic dimensions. By selecting the lowest

[5]In the majority of cases, the induced dimensions indeed contain such clear-cut semantics, so that the dimensions can be rightfully labeled as above.

Kullback-Leibler divergence between the semantic probability distribution of the target instance and the semantic probability distributions of the candidate senses, the algorithm is able to assign the 'computer' sense of the target noun *chip*.

4 Evaluation

4.1 Dataset

Our word sense induction and disambiguation model is trained and tested on the dataset of the SEMEVAL-2010 WSI/WSD task (Manandhar et al., 2010). The SEMEVAL-2010 WSI/WSD task is based on a dataset of 100 target words, 50 nouns and 50 verbs. For each target word, a training set is provided from which the senses of the word have to be induced without using any other resources. The training set for a target word consists of a set of target word instances in context (sentences or paragraphs). The complete training set contains 879,807 instances, viz. 716,945 noun and 162,862 verb instances.

The senses induced during training are used for disambiguation in the testing phase. In this phase, the system is provided with a test set that consists of unseen instances of the target words. The test set contains 8,915 instances in total, of which 5,285 nouns and 3,630 verbs. The instances in the test set are tagged with OntoNotes senses (Hovy et al., 2006). The system needs to disambiguate these instances using the senses acquired during training.

4.2 Implementational details

The SEMEVAL training set has been part of speech tagged and lemmatized with the Stanford Part-Of-Speech Tagger (Toutanova and Manning, 2000; Toutanova et al., 2003) and parsed with MaltParser (Nivre et al., 2006), trained on sections 2-21 of the Wall Street Journal section of the Penn Treebank extended with about 4000 questions from the QuestionBank[6] in order to extract dependency triples. The SEMEVAL test set has only been tagged and lemmatized, as our disambiguation model does not use dependency triples as features (contrary to the induction model).

[6] http://maltparser.org/mco/english_parser/engmalt.html

We constructed two different models – one for nouns and one for verbs. For each model, the matrices needed for our interleaved NMF factorization are extracted from the corpus. The noun model was built using 5K nouns, 80K dependency relations, and 2K context words (excluding stop words) with highest frequency in the training set, which yields matrices of 5K nouns $\times$ 80K dependency relations, 5K nouns $\times$ 2K context words, and 80K dependency relations $\times$ 2K context words. The model for verbs was constructed analogously, using 3K verbs, and the same number of dependency relations and context words. For our initial k-means clustering, we set $k = 600$ for nouns, and $k = 400$ for verbs. For the underlying interleaved NMF model, we used 50 iterations, and factored the model to 50 dimensions.

4.3 Evaluation measures

The results of the systems participating in the SEMEVAL-2010 WSI/WSD task are evaluated both in a supervised and in an unsupervised manner.

The supervised evaluation in the SEMEVAL-2010 WSI/WSD task follows the scheme of the SEMEVAL-2007 WSI task (Agirre and Soroa, 2007), with some modifications. One part of the test set is used as a mapping corpus, which maps the automatically induced clusters to gold standard senses; the other part acts as an evaluation corpus. The mapping between clusters and gold standard senses is used to tag the evaluation corpus with gold standard tags. The systems are then evaluated as in a standard WSD task, using recall.

In the unsupervised evaluation, the induced senses are evaluated as clusters of instances which are compared to the sets of instances tagged with the gold standard senses (corresponding to classes). Two partitions are thus created over the test set of a target word: a set of automatically generated clusters and a set of gold standard classes. A number of these instances will be members of both one gold standard class and one cluster. Consequently, the quality of the proposed clustering solution is evaluated by comparing the two groupings and measuring their similarity.

Two evaluation metrics are used during the unsupervised evaluation in order to estimate the quality of the clustering solutions, the *V-Measure* (Rosenberg and Hirschberg, 2007) and the *paired F-*

Score (Artiles et al., 2009). *V-Measure* assesses the quality of a clustering by measuring its *homogeneity* (h) and its *completeness* (c). Homogeneity refers to the degree that each cluster consists of data points primarily belonging to a single gold standard class, while completeness refers to the degree that each gold standard class consists of data points primarily assigned to a single cluster. V-Measure is the harmonic mean of h and c.

$$VM = \frac{2 \cdot h \cdot c}{h + c} \tag{7}$$

In the *paired F-Score* (Artiles et al., 2009) evaluation, the clustering problem is transformed into a classification problem (Manandhar et al., 2010). A set of instance pairs is generated from the automatically induced clusters, which comprises pairs of the instances found in each cluster. Similarly, a set of instance pairs is created from the gold standard classes, containing pairs of the instances found in each class. *Precision* is then defined as the number of common instance pairs between the two sets to the total number of pairs in the clustering solution (cf. formula 8). *Recall* is defined as the number of common instance pairs between the two sets to the total number of pairs in the gold standard (cf. formula 9). Precision and recall are finally combined to produce the harmonic mean (cf. formula 10).

$$P = \frac{|F(K) \cap F(S)|}{|F(K)|} \tag{8}$$

$$R = \frac{|F(K) \cap F(S)|}{|F(S)|} \tag{9}$$

$$FS = \frac{2 \cdot P \cdot R}{P + R} \tag{10}$$

The obtained results are also compared to two baselines. The most frequent sense (MFS) baseline groups all testing instances of a target word into one cluster. The *Random* baseline randomly assigns an instance to one of the clusters.[7] This baseline is executed five times and the results are averaged.

[7]The number of clusters in *Random* was chosen to be roughly equal to the average number of senses in the gold standard.

4.4 Results

4.4.1 Unsupervised evaluation

In table 1, we present the performance of a number of algorithms on the V-measure. We compare our V-measure scores with the scores of the best-ranked systems in the SEMEVAL 2010 WSI/WSD task, both for the complete data set and for nouns and verbs separately. The fourth column shows the average number of clusters induced in the test set by each algorithm. The MFS baseline has a V-Measure equal to 0, since by definition its completeness is 1 and its homogeneity is 0.

NMF$_{con}$ – our model that takes a conservative approach in the induction of candidate senses – does not beat the random baseline. NMF$_{lib}$ – our model that is more liberal in inducing senses – reaches better results. With 11.8%, it scores similar to other algorithms that induce a similar average number of clusters, such as Duluth-WSI (Pedersen, 2010).

Pedersen (2010) has shown that the V-Measure tends to favour systems producing a higher number of clusters than the number of gold standard senses. This is reflected in the scores of our models as well.

VM (%)	**all**	**noun**	**verb**	**#cl**
Hermit	16.2	16.7	15.6	10.78
UoY	15.7	20.6	8.5	11.54
KSU KDD	15.7	18.0	12.4	17.50
NMF$_{lib}$	11.8	13.5	9.4	4.80
Duluth-WSI	9.0	11.4	5.7	4.15
Random	4.4	4.2	4.6	4.00
NMF$_{con}$	3.9	3.9	3.9	1.58
MFS	0.0	0.0	0.0	1.00

Table 1: Unsupervised V-measure evaluation on SEMEVAL test set

Motivated by the large divergences in the system rankings on the different metrics used in the SEMEVAL-2010 WSI/WSD task, Pedersen evaluated the metrics themselves. His evaluation relied on the assumption that a good measure should assign low scores to random baselines. Pedersen showed that the V-Measure continued to improve as randomness increased. We agree with Pedersen's conclusion that the V-Measure results should be interpreted with caution, but we still report the results in order

to perform a global comparison, on all metrics, of our system's performance to the systems that participated to the SEMEVAL task.

Contrary to V-Measure, paired F-score is a fairly reliable measure and the only one that managed to identify and expose random baselines in the above mentioned metric evaluation. This means that the random systems used for testing were ranked low when a high number of random senses was used.

In table 2, the paired F-Score of a number of algorithms is given. The paired F-Score penalizes systems when they produce a higher number of clusters (low recall) or a lower number of clusters (low precision) than the gold standard number of senses. We again compare our results with the scores of the best-ranked systems in the SEMEVAL-2010 WSI/WSD TASK.

FS (%)	all	noun	verb	#cl
MFS	63.5	57.0	72.7	1.00
Duluth-WSI-SVD-Gap	63.3	57.0	72.4	1.02
NMF$_{con}$	60.2	54.6	68.4	1.58
NMF$_{lib}$	45.3	42.2	49.8	5.42
Duluth-WSI	41.1	37.1	46.7	4.15
Random	31.9	30.4	34.1	4.00

Table 2: Unsupervised paired F-score evaluation on SEMEVAL testset

NMF$_{con}$ reaches a score of 60.2%, which is again similar to other algorithms that induce the same average number of clusters. NMF$_{lib}$ scores 45.3%, indicating that the algorithm is able to retain a reasonable F-Score while at the same time inducing a significant number of clusters. This especially becomes clear when comparing its score to the other algorithms.

4.4.2 Supervised evaluation

In the supervised evaluation, the automatically induced clusters are mapped to gold standard senses, using the mapping corpus (i.e. one part of the test set). The obtained mapping is used to tag the evaluation corpus (i.e. the other part of the test set) with gold standard tags, which means that the methods are evaluated in a standard WSD task.

Table 3 shows the recall of our algorithms in the supervised evaluation, again compared to other algorithms evaluated in the SEMEVAL-2010 WSI/WSD task.

SR (%)	all	noun	verb	#S
NMF$_{lib}$	62.6	57.3	70.2	1.82
UoY	62.4	59.4	66.8	1.51
Duluth-WSI	60.5	54.7	68.9	1.66
NMF$_{con}$	60.3	54.5	68.8	1.21
MFS	58.7	53.2	66.6	1.00
Random	57.3	51.5	65.7	1.53

Table 3: Supervised recall for SEMEVAL testset, 80% mapping, 20% evaluation

NMF$_{lib}$ gets 62.6%, which makes it the best scoring algorithm on the supervised evaluation. NMF$_{con}$ reaches 60.3%, which again indicates that it is in the same ballpark as other algorithms that induce a similar average number of senses.

Some doubts have been cast on the representativeness of the supervised recall results as well. According to Pedersen (2010), the supervised learning algorithm that underlies this evaluation method tends to converge to the Most Frequent Sense (MFS) baseline, because the number of senses that the classifier assigns to the test instances is rather low. We think these shortcomings indicate the need for the development of new evaluation metrics, capable of providing a more accurate evaluation of the performance of WSI systems. Nevertheless, these metrics still constitute a useful testbed for comparing the performance of different systems.

5 Conclusion and future work

In this paper, we presented a model based on latent semantics that is able to perform word sense induction as well as disambiguation. Using latent topical dimensions, the model is able to discriminate between different senses of a word, and subsequently disambiguate particular instances of a word. The evaluation results indicate that our model reaches state-of-the-art performance compared to other systems that participated in the SEMEVAL-2010 word sense induction and disambiguation task. Moreover, our global approach is able to reach similar performance on an evaluation set that is tuned to fit the needs of local approaches. The evaluation set con-

tains an enormous amount of contexts for only a small number of target words, favouring methods that induce senses on a per-word basis. A global approach like ours is likely to induce a more balanced sense inventory using an unbiased corpus, and is likely to outperform local methods when such an unbiased corpus is used as input. We therefore think that the global, unified approach to word sense induction and disambiguation presented here provides a genuine and powerful solution to the problem at hand.

We conclude with some issues for future work. First of all, we would like to evaluate the approach presented here using a more balanced and unbiased corpus, and compare its performance on such a corpus to local approaches. Secondly, we would also like to include grammatical dependency information in the disambiguation step of the algorithm. For now, the disambiguation step only uses a word's context words; enriching the feature set with dependency information is likely to improve the performance of the disambiguation.

Acknowledgments

This work is supported by the Scribo project, funded by the French 'pôle de compétitivité' System@tic, and by the French national grant EDyLex (ANR-09-CORD-008).

References

Eneko Agirre and Aitor Soroa. 2007. SemEval-2007 Task 02: Evaluating word sense induction and discrimination systems. In *Proceedings of the fourth International Workshop on Semantic Evaluations (SemEval), ACL*, pages 7–12, Prague, Czech Republic.

Eneko Agirre, David Martínez, Ojer López de Lacalle, and Aitor Soroa. 2006. Two graph-based algorithms for state-of-the-art WSD. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-06)*, pages 585–593, Sydney, Australia.

Marianna Apidianaki and Tim Van de Cruys. 2011. A Quantitative Evaluation of Global Word Sense Induction. In *Proceedings of the 12th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing), published in Springer Lecture Notes in Computer Science (LNCS)*, volume 6608, pages 253–264, Tokyo, Japan.

Javier Artiles, Enrique Amigó, and Julio Gonzalo. 2009. The role of named entities in web people search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pages 534–542, Singapore.

Stefan Bordag. 2006. Word sense induction: Triplet-based clustering and automatic evaluation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, pages 137–144, Trento, Italy.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the Human Language Technology / North American Association of Computational Linguistics conference (HLT-NAACL-06)*, pages 57–60, New York, NY.

Nancy Ide and Yorick Wilks. 2007. Making Sense About Sense. In Eneko Agirre and Philip Edmonds, editors, *Word Sense Disambiguation, Algorithms and Applications*, pages 47–73. Springer.

Thomas Landauer and Susan Dumais. 1997. A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychology Review*, 104:211–240.

Thomas Landauer, Peter Foltz, and Darrell Laham. 1998. An Introduction to Latent Semantic Analysis. *Discourse Processes*, 25:295–284.

Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, volume 13, pages 556–562.

Dekang Lin. 1998. Automatic Retrieval and Clustering of Similar Words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL98)*, volume 2, pages 768–774, Montreal, Quebec, Canada.

Suresh Manandhar, Ioannis P. Klapaftis, Dmitriy Dligach, and Sameer S. Pradhan. 2010. SemEval-2010 Task 14: Word Sense Induction & Disambiguation. In *Proceedings of the fifth International Workshop on Semantic Evaluation (SemEval), ACL-10*, pages 63–68, Uppsala, Sweden.

Roberto Navigli. 2009. Word Sense Disambiguation: a Survey. *ACM Computing Surveys*, 41(2):1–69.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the fifth International Conference on Language Resources and Evaluation (LREC-06)*, pages 2216–2219, Genoa, Italy.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 613–619, Edmonton, Alberta, Canada.

Ted Pedersen. 2010. Duluth-WSI: SenseClusters Applied to the Sense Induction Task of SemEval-2. In *Proceedings of the fifth International Workshop on Semantic Evaluations (SemEval-2010)*, pages 363–366, Uppsala, Sweden.

Amruta Purandare and Ted Pedersen. 2004. Word Sense Discrimination by Clustering Contexts in Vector and Similarity Spaces. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, pages 41–48, Boston, MA.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the Joint 2007 Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic.

Hinrich Schütze. 1998. Automatic Word Sense Discrimination. *Computational Linguistics*, 24(1):97–123.

Kristina Toutanova and Christopher D. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, pages 63–70.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of the Human Language Technology / North American Association of Computational Linguistics conference (HLT-NAACL-03*, pages 252–259, Edmonton, Canada.

Tim Van de Cruys. 2008. Using Three Way Data for Word Sense Discrimination. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-08)*, pages 929–936, Manchester, UK.

Jean Véronis. 2004. Hyperlex: lexical cartography for information retrieval. *Computer Speech & Language*, 18(3):223–252.

Dominic Widdows and Beate Dorow. 2002. A Graph Model for Unsupervised Lexical Acquisition. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING-02)*, pages 1093–1099, Taipei, Taiwan.

Confidence Driven Unsupervised Semantic Parsing

Dan Goldwasser * **Roi Reichart** † **James Clarke** * **Dan Roth** *
* Department of Computer Science, University of Illinois at Urbana-Champaign
{goldwas1,clarkeje,danr}@illinois.edu
† Computer Science and Artificial Intelligence Laboratory, MIT
roiri@csail.mit.edu

Abstract

Current approaches for semantic parsing take a supervised approach requiring a considerable amount of training data which is expensive and difficult to obtain. This supervision bottleneck is one of the major difficulties in scaling up semantic parsing.

We argue that a semantic parser can be trained effectively without annotated data, and introduce an unsupervised learning algorithm. The algorithm takes a self training approach driven by confidence estimation. Evaluated over Geoquery, a standard dataset for this task, our system achieved 66% accuracy, compared to 80% of its fully supervised counterpart, demonstrating the promise of unsupervised approaches for this task.

1 Introduction

Semantic parsing, the ability to transform Natural Language (NL) input into a formal Meaning Representation (MR), is one of the longest standing goals of natural language processing. The importance of the problem stems from both theoretical and practical reasons, as the ability to convert NL into a formal MR has countless applications.

The term *semantic parsing* has been used ambiguously to refer to several semantic tasks (e.g., semantic role labeling). We follow the most common definition of this task: finding a mapping between NL input and its interpretation expressed in a well-defined formal MR language. Unlike shallow semantic analysis tasks, the output of a semantic parser is complete and unambiguous to the extent it can be understood or even executed by a computer system.

Current approaches for this task take a data driven approach (Zettlemoyer and Collins, 2007; Wong and Mooney, 2007), in which the learning algorithm is given a set of NL sentences as input and their corresponding MR, and learns a statistical semantic parser — a set of parameterized rules mapping lexical items and syntactic patterns to their MR. Given a sentence, these rules are applied recursively to derive the most probable interpretation.

Since semantic interpretation is limited to the syntactic patterns observed in the training data, in order to work well these approaches require considerable amounts of annotated data. Unfortunately annotating sentences with their MR is a time consuming task which requires specialized domain knowledge and therefore minimizing the supervision effort is one of the key challenges in scaling semantic parsers.

In this work we present the first unsupervised approach for this task. Our model compensates for the lack of training data by employing a self training protocol based on identifying high confidence self labeled examples and using them to retrain the model. We base our approach on a simple observation: semantic parsing is a difficult structured prediction task, which requires learning a complex model, however identifying good predictions can be done with a far simpler model capturing repeating patterns in the predicted data. We present several simple, yet highly effective confidence measures capturing such patterns, and show how to use them to train a semantic parser without manually annotated sentences.

Our basic premise, that predictions with high confidence score are of high quality, is further used to improve the performance of the unsupervised train-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1486–1495,
Portland, Oregon, June 19-24, 2011.

ing procedure. Our learning algorithm takes an EM-like iterative approach, in which the predictions of the previous stage are used to bias the model. While this basic scheme was successfully applied to many unsupervised tasks, it is known to converge to a sub optimal point. We show that by using confidence estimation as a proxy for the model's prediction quality, the learning algorithm can identify a better model compared to the default convergence criterion.

We evaluate our learning approach and model on the well studied Geoquery domain (Zelle and Mooney, 1996; Tang and Mooney, 2001), consisting of natural language questions and their prolog interpretations used to query a database consisting of U.S. geographical information. Our experimental results show that using our approach we are able to train a good semantic parser without annotated data, and that using a confidence score to identify good models results in a significant performance improvement.

2 Semantic Parsing

We formulate semantic parsing as a structured prediction problem, mapping a NL input sentence (denoted $\mathbf{x}$), to its highest ranking MR (denoted $\mathbf{z}$). In order to correctly parametrize and weight the possible outputs, the decision relies on an intermediate representation: an alignment between textual fragments and their meaning representation (denoted $\mathbf{y}$). Fig. 1 describes a concrete example of this terminology. In our experiments the input sentences $\mathbf{x}$ are natural language queries about U.S. geography taken from the Geoquery dataset. The meaning representation $\mathbf{z}$ is a formal language database query, this output representation language is described in Sec. 2.1.

The prediction function, mapping a sentence to its corresponding MR, is formalized as follows:

$$\hat{\mathbf{z}} = F_{\mathbf{w}}(\mathbf{x}) = \arg\max_{\mathbf{y}\in\mathcal{Y},\mathbf{z}\in\mathcal{Z}} \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) \qquad (1)$$

Where Φ is a feature function defined over an input sentence $\mathbf{x}$, alignment $\mathbf{y}$ and output $\mathbf{z}$. The weight vector $\mathbf{w}$ contains the model's parameters, whose values are determined by the learning process.

We refer to the $\arg\max$ above as the inference problem. Given an input sentence, solving this inference problem based on Φ and $\mathbf{w}$ is what compromises our semantic parser. In practice the parsing decision is decomposed into smaller decisions (Sec. 2.2). Sec. 4 provides more details about the feature representation and inference procedure used.

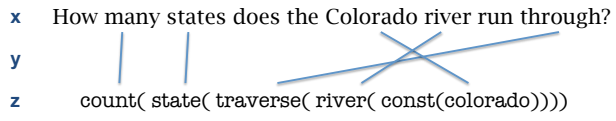


Figure 1: Example of an input sentence ($\mathbf{x}$), meaning representation ($\mathbf{z}$) and the alignment between the two ($\mathbf{y}$) for the Geoquery domain

Current approaches obtain $\mathbf{w}$ using annotated data, typically consisting of ($\mathbf{x}$, $\mathbf{z}$) pairs. In Sec. 3 we describe our unsupervised learning procedure, that is how to obtain $\mathbf{w}$ without annotated data.

2.1 Target Meaning Representation

The output of the semantic parser is a logical formula, grounding the semantics of the input sentence in the domain language (i.e., the Geoquery domain). We use a subset of first order logic consisting of typed constants (corresponding to specific states, etc.) and functions, which capture relations between domains entities and properties of entities (e.g., `population` : $E \rightarrow N$). The semantics of the input sentence is constructed via functional composition, done by the substitution operator. For example, given the function `next_to(x)` and the expression `const(texas)`, substitution replaces the occurrence of the free variable `x` with the expression, resulting in a new formula: `next_to(const(texas))`. For further details we refer the reader to (Zelle and Mooney, 1996).

2.2 Semantic Parsing Decisions

The inference problem described in Eq. 1 selects the top ranking output formula. In practice this decision is decomposed into smaller decisions, capturing local mapping of input tokens to logical fragments and their composition into larger fragments. These decisions are further decomposed into a feature representation, described in Sec. 4.

The first type of decisions are encoded directly by the alignment ($\mathbf{y}$) between the input tokens and their corresponding predicates. We refer to these as **first**

order decisions. The pairs connected by the alignment ($\mathbf{y}$) in Fig. 1 are examples of such decisions.

The final output structure $\mathbf{z}$ is constructed by composing individual predicates into a complete formula. For example, consider the formula presented in Fig. 1: `river( const(colorado))` is a composition of two predicates `river` and `const(colorado)`. We refer to the composition of two predicates, associated with their respective input tokens, as **second order decisions**.

In order to formulate these decisions, we introduce the following notation. c is a constituent in the input sentence $\mathbf{x}$ and $\mathcal{D}$ is the set of all function and constant symbols in the domain. The alignment $\mathbf{y}$ is a set of mappings between constituents and symbols in the domain $\mathbf{y} = \{(c, s)\}$ where $s \in \mathcal{D}$.

We denote by s_i the i-th output predicate composition in $\mathbf{z}$, by $s_{i-1}(s_i)$ the composition of the $(i-1)$-th predicate on the i-th predicate and by $y(s_i)$ the input word corresponding to that predicate according to the alignment $\mathbf{y}$.

3 Unsupervised Semantic Parsing

Our learning framework takes a self training approach in which the learner is iteratively trained over its own predictions. Successful application of this approach depends heavily on two important factors - how to select high quality examples to train the model on, and how to define the learning objective so that learning can halt once a good model is found.

Both of these questions are trivially answered when working in a supervised setting: by using the labeled data for training the model, and defining the learning objective with respect to the annotated data (for example, loss-minimization in the supervised version of our system).

In this work we suggest to address both of the above concerns by approximating the quality of the model's predictions using a confidence measure computed over the statistics of the self generated predictions. Output structures which fall close to the center of mass of these statistics will receive a high confidence score.

The first issue is addressed by using examples assigned a high confidence score to train the model, acting as labeled examples.

We also note that since the confidence score provides a good indication for the model's prediction performance, it can be used to approximate the overall *model* performance, by observing the model's total confidence score over all its predictions. This allows us to set a performance driven goal for our learning process - return the model maximizing the confidence score over all predictions. We describe the details of integrating the confidence score into the learning framework in Sec. 3.1.

Although using the model's prediction score (i.e., $\mathbf{w}^T\Phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$) as an indication of correctness is a natural choice, we argue and show empirically, that unsupervised learning driven by confidence estimation results in a better performing model. This empirical behavior also has theoretical justification: training the model using examples selected according to the model's parameters (i.e., the top ranking structures) may not generalize much further beyond the existing model, as the training examples will simply reinforce the existing model. The statistics used for confidence estimation are different than those used by the model to create the output structures, and can therefore capture additional information unobserved by the prediction model. This assumption is based on the well established idea of multi-view learning, applied successfully to many NL applications (Blum and Mitchell, 1998; Collins and Singer, 1999). According to this idea if two models use different views of the data, each of them can enhance the learning process of the other.

The success of our learning procedure hinges on finding good confidence measures, whose confidence prediction correlates well with the true quality of the prediction. The ability of unsupervised confidence estimation to provide high quality confidence predictions can be explained by the observation that prominent prediction patterns are more likely to be correct. If a non-random model produces a prediction pattern multiple times it is likely to be an indication of an underlying phenomenon in the data, and therefore more likely to be correct. Our specific choice of confidence measures is guided by the intuition that unlike structure prediction (i.e., solving the inference problem) which requires taking statistics over complex and intricate patterns, identifying high quality predictions can be done using much simpler patterns that are significantly easier to capture.

In the reminder of this section we describe our

Algorithm 1 Unsupervised Confidence driven Learning

Input: Sentences $\{\mathbf{x}^l\}_{l=1}^N$,
initial weight vector $\mathbf{w}$

1: **define** $Confidence : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \rightarrow \mathcal{R}$,
$i = 0,\ S_i = \emptyset$
2: **repeat**
3: **for** $l = 1, \ldots, N$ **do**
4: $\hat{\mathbf{y}}, \hat{\mathbf{z}} = \arg\max_{\mathbf{y},\mathbf{z}} \mathbf{w}^T \Phi(\mathbf{x}^l, \mathbf{y}, \mathbf{z})$
5: $S_i = S_i \cup \{\mathbf{x}^l, \hat{\mathbf{y}}, \hat{\mathbf{z}}\}$
6: **end for**
7: $Confidence$ = compute confidence statistics
8: S_i^{conf} = select from S_i using $Confidence$
9: $\mathbf{w_i} \leftarrow Learn(\cup_i S_i^{conf})$
10: $i = i + 1$
11: **until** S_i^{conf} has no new unique examples
12: $best = \arg\max_{\mathbf{i}}(\sum_{s \in S_i} Confidence(s))/|S|$
13: **return** $\mathbf{w_{best}}$

learning approach. We begin by introducing the overall learning framework (Sec. 3.1), we then explain the rational behind confidence estimation over self-generated data and introduce the confidence measures used in our experiments (Sec. 3.2). We conclude with a description of the specific learning algorithms used for updating the model (Sec. 3.3).

3.1 Unsupervised Confidence-Driven Learning

Our learning framework works in an EM-like manner, iterating between two stages: making predictions based on its current set of parameters and then retraining the model using a subset of the predictions, assigned high confidence. The learning process "discovers" new high confidence training examples to add to its training set over multiple iterations, and converges when the model no longer adds new training examples.

While this is a natural convergence criterion, it provides no performance guarantees, and in practice it is very likely that the quality of the model (i.e., its performance) fluctuates during the learning process. We follow the observation that confidence estimation can be used to approximate the performance of the entire model and return the model with the highest overall prediction confidence.

We describe this algorithmic framework in detail in Alg. 1. Our algorithm takes as input a set of natural language sentences and a set of parameters used for making the initial predictions[1]. The algorithm then iterates between the two stages - predicting the output structure for each sentence (line 4), and updating the set of parameters (line 9). The specific learning algorithms used are discussed in Sec. 3.3. The training examples required for learning are obtained by selecting high confidence examples - the algorithm first takes statistics over the current predicted set of output structures (line 7), and then based on these statistics computes a confidence score for each structure, selecting the top ranked ones as positive training examples, and if needed, the bottom ones as negative examples (line 8). The set of top confidence examples (for either correct or incorrect prediction), at iteration i of the algorithm, is denoted S_i^{conf}. The exact nature of the confidence computation is discussed in Sec. 3.2.

The algorithm iterates between these two stages, at each iteration it adds more self-annotated examples to its training set, learning therefore converges when no new examples are added (line 11). The algorithm keeps track of the models it trained at each stage throughout this process, and returns the one with the highest averaged overall confidence score (lines 12-13). At each stage, the overall confidence score is computed by averaging over all the confidence scores of the predictions made at that stage.

3.2 Unsupervised Confidence Estimation

Confidence estimation is calculated over a batch of input ($\mathbf{x}$) - output ($\mathbf{z}$) pairs. Each pair decomposes into smaller first order and second order decisions (defined Sec. 2.2). Confidence estimation is done by computing the statistics of these decisions, over the entire set of predicted structures. In the rest of this section we introduce the confidence measures used by our system.

Translation Model The first approach essentially constructs a simplified translation model, capturing word-to-predicate mapping patterns. This can be considered as an abstraction of the prediction model: we collapse the intricate feature representation into

[1] Since we commit to the max-score output prediction, rather than summing over all possibilities, we require a reasonable initialization point. We initialized the weight vector using simple, straight-forward heuristics described in Sec. 5.

high level decisions and take statistics over these decisions. Since it takes statistics over considerably less variables than the actual prediction model, we expect this model to make reliable confidence predictions. We consider two variations of this approach, the first constructs a unigram model over the first order decisions and the second a bigram model over the second order decisions. Formally, given a set of predicted structures we define the following confidence scores:

Unigram Score:

$$p(\mathbf{z}|\mathbf{x}) = \prod_{i=1}^{|z|} p(s_i|y(s_i))$$

Bigram Score:

$$p(\mathbf{z}|\mathbf{x}) = \prod_{i=1}^{|z|} p(s_{i-1}(s_i)|y(s_{i-1}), y(s_i))$$

Structural Proportion Unlike the first approach which decomposes the predicted structure into individual decisions, this approach approximates the model's performance by observing global properties of the structure. We take statistics over the proportion between the number of predicates in $\mathbf{z}$ and the number of words in $\mathbf{x}$.

Given a set of structure predictions S, we compute this proportion for each structure (denoted as $Prop(\mathbf{x}, \mathbf{z})$) and calculate the average proportion over the entire set (denoted as $AvProp(S)$). The confidence score assigned to a given structure $(\mathbf{x}, \mathbf{y})$ is simply the difference between its proportion and the averaged proportion, or formally

$$PropScore(S, (\mathbf{x}, \mathbf{z})) = AvProp(S) - Prop(x, z)$$

This measure captures the global complexity of the predicted structure and penalizes structures which are too complex (high negative values) or too simplistic (high positive values).

Combined The two approaches defined above capture different views of the data, a natural question is then - *can these two measures be combined to provide a more powerful estimation?* We suggest a third approach which combines the first two approaches. It first uses the score produced by the latter approach to filter out unlikely candidates, and then ranks the remaining ones with the former approach and selects those with the highest rank.

3.3 Learning Algorithms

Given a set of self generated structures, the parameter vector can be updated (line 9 in Alg. 1). We consider two learning algorithm for this purpose.

The first is a **binary learning** algorithm, which considers learning as a classification problem, that is finding a set of weights $\mathbf{w}$ that can best separate correct from incorrect structures. The algorithm decomposes each predicted formula and its corresponding input sentence into a feature vector $\Phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$ normalized by the size of the input sentence $|\mathbf{x}|$, and assigns a binary label to this vector[2]. The learning process is defined over both positive and negative training examples. To accommodate that we modify line 8 in Alg. 1, and use the confidence score to select the top ranking examples as positive examples, and the bottom ranking examples as negative examples. We use a linear kernel SVM with squared-hinge loss as the underlying learning algorithm.

The second is a **structured learning** algorithm which considers learning as a ranking problem, i.e., finding a set of weights $\mathbf{w}$ such that the "gold structure" will be ranked on top, preferably by a large margin to allow generalization.The structured learning algorithm can directly use the top ranking predictions of the model (line 8 in Alg. 1) as training data. In this case the underlying algorithm is a structural SVM with squared-hinge loss, using hamming distance as the distance function. We use the cutting-plane method to efficiently optimize the learning process' objective function.

4 Model

Semantic parsing as formulated in Eq. 1 is an inference procedure selecting the top ranked output logical formula. We follow the inference approach in (Roth and Yih, 2007; Clarke et al., 2010) and formalize this process as an Integer Linear Program (ILP). Due to space consideration we provide a brief description, and refer the reader to that paper for more details.

[2]Without normalization longer sentences would have more influence on binary learning problem. Normalization is therefore required to ensure that each sentence contributes equally to the binary learning problem regardless of its length.

4.1 Inference

The inference decision (Eq. 1) is decomposed into smaller decisions, capturing mapping of input tokens to logical fragments (first order) and their composition into larger fragments (second order). We encode a first-order decision as α_{cs}, a binary variable indicating that constituent c is aligned with the logical symbol $\mathtt{s}$. A second-order decision $\beta_{cs,dt}$, is encoded as a binary variable indicating that the symbol $\mathtt{t}$ (associated with constituent d) is an argument of a function $\mathtt{s}$ (associated with constituent c). We frame the inference problem over these decisions:

$$F_{\mathbf{w}}(\mathbf{x}) = \arg\max_{\alpha,\beta} \sum_{c \in \mathbf{x}} \sum_{s \in D} \alpha_{cs} \cdot \mathbf{w}^T \Phi_1(\mathbf{x}, c, s) + \sum_{c,d \in \mathbf{x}} \sum_{s,t \in D} \beta_{cs,dt} \cdot \mathbf{w}^T \Phi_2(\mathbf{x}, c, s, d, t) \quad (2)$$

We restrict the possible assignments to the decision variables, forcing the resulting output formula to be *syntactically* legal, for example by restricting active β-variables to be type consistent, and force the resulting functional composition to be acyclic. We take advantage of the flexible ILP framework, and encode these restrictions as global constraints over Eq. 2. We refer the reader to (Clarke et al., 2010) for a full description of the constraints used.

4.2 Features

The inference problem defined in Eq. (2) uses two feature functions: Φ_1 and Φ_2.

First-order decision features Φ_1 Determining if a logical symbol is aligned with a specific constituent depends mostly on lexical information. Following previous work (e.g., (Zettlemoyer and Collins, 2005)) we create a small lexicon, mapping logical symbols to surface forms.[3] Existing approaches rely on annotated data to extend the lexicon. Instead we rely on external knowledge (Miller et al., 1990) and add features which measure the lexical similarity between a constituent and a logical symbol's surface forms (as defined by the lexicon).

[3]The lexicon contains on average 1.42 words per function and 1.07 words per constant.

Model	Description
INITIAL MODEL	Manually set weights (Sec. 5.1)
PRED. SCORE	normalized prediction (Sec. 5.1)
ALL EXAMPLES	All top structures (Sec. 5.1)
UNIGRAM	Unigram score (Sec. 3.2)
BIGRAM	Bigram score (Sec. 3.2)
PROPORTION	Words-predicate prop (Sec. 3.2)
COMBINED	Combined estimators (Sec. 3.2)
RESPONSE BASED	Supervised (binary) (Sec. 5.1)
SUPERVISED	Fully Supervised (Sec. 5.1)

Table 1: Compared systems and naming conventions.

Second-order decision features Φ_2 Second order decisions rely on syntactic information. We use the dependency tree of the input sentence. Given a second-order decision $\beta_{cs,dt}$, the dependency feature takes the normalized distance between the head words in the constituents c and d. In addition, a set of features indicate which logical symbols are usually composed together, without considering their alignment to the text.

5 Experiments

In this section we describe our experimental evaluation. We compare several confidence measures and analyze their properties. Tab. 1 defines the naming conventions used throughout this section to refer to the different models we evaluated. We begin by describing our experimental setup and then proceed to describe the experiments and their results. For the sake of clarity we focus on the best performing models (COMBINED using BIGRAM and PROPORTION) first and discuss other models later in the section.

5.1 Experimental Settings

In all our experiments we used the Geoquery dataset (Zelle and Mooney, 1996), consisting of U.S. geography NL questions and their corresponding Prolog logical MR. We used the data split described in (Clarke et al., 2010), consisting of 250 queries for evaluation purposes. We compared our system to several supervised models, which were trained using a disjoint set of queries. Our learning system had access only to the NL questions, and the logical forms were only used to evaluate the system's performance. We report the proportion of correct structures (accuracy). Note that this evaluation cor-

responds to the 0/1 loss over the predicted structures.

Initialization Our learning framework requires an initial weight vector as input. We use a straight forward heuristic and provide uniform positive weights to three features. This approach is similar in spirit to previous works (Clarke et al., 2010; Zettlemoyer and Collins, 2007). We refer to this system as INITIAL MODEL throughout this section.

Competing Systems We compared our system to several other systems:

(1) **PRED. SCORE**: An unsupervised framework using the model's internal prediction score ($\mathbf{w}^T\Phi(\mathbf{x},\mathbf{y},\mathbf{z})$) for confidence estimation.

(2) **ALL EXAMPLES**: Treating all predicted structures as correct, i.e., at each iteration the model is trained over all the predictions it made. The reported score was obtained by selecting the model at the training iteration with the highest overall confidence score (see line 12 in Alg. 1).

(3) **RESPONSE BASED**: A natural upper bound to our framework is the approach used in (Clarke et al., 2010). While our approach is based on assessing the correctness os the model's predictions according to unsupervised confidence estimation, their framework is provided with external supervision for these decisions, indicating if the predicted structures are correct.

(4) **SUPERVISED**: A fully supervised framework trained over 250 $(\mathbf{x},\mathbf{z})$ pairs using structured SVM.

5.2 Results

Our experiments aim to clarify three key points:

(1) Can a semantic parser indeed be trained without any form of external supervision? this is our key question, as this is the first attempt to approach this task with an unsupervised learning protocol.[4] In order to answer it, we report the overall performance of our system in Tab. 2.

The manually constructed model INITIALMODEL achieves a performance of 0.22. We can expect learning to improve on this baseline. We compare three self-trained systems, ALL EXAMPLES, PREDICTIONSCORE and COMBINED, which differ in their sample selection strategy, but all use confidence estimation for selecting the final semantic parsing model. The ALL EXAMPLES approach achieves an accuracy score of 0.656. PREDICTIONSCORE only achieves a performance of 0.164 using the binary learning algorithm and 0.348 using the structured learning algorithm. Finally, our confidence-driven technique COMBINED achieved a score of 0.536 for the binary case and 0.664 for the structured case, the best performing models in both cases. As expected, the supervised systems RESPONSE BASED and SUPERVISED achieve the best performance.

These results show that training the model with training examples selected carefully will improve learning - as the best performance is achieved with perfect knowledge of the predictions correctness (RESPONSE BASED). Interestingly the difference between the structured version of our system and that of RESPONSE BASED is only 0.07, suggesting that we can recover the binary feedback signal with high precision. The low performance of the PREDICTIONSCORE model is also not surprising, and it demonstrates one of the key principles in confidence estimation - the score should be comparable across predictions done over different inputs, and not the same input, as done in PREDICTIONSCORE model.

(2) How does confidence driven sample selection contribute to the learning process? Comparing the systems driven by confidence sample-selection to the ALL EXAMPLES approach uncovers an interesting tradeoff between training with more (noisy) data and selectively training the system with higher quality examples. We argue that carefully selecting high quality training examples will result in better performance. The empirical results indeed support our argument, as the best performing model (RESPONSE BASED) is achieved by sample selection with perfect knowledge of prediction correctness. The confidence-based sample selection system (COMBINED) is the best performing system out of all the self-trained systems. Nonetheless, the ALL EXAMPLES strategy performs well when compared to COMBINED, justifying a closer look at that aspect of our system.

We argue that different confidence measures capture different properties of the data, and hypothe-

[4] While unsupervised learning for various semantic tasks has been widely discussed, this is the first attempt to tackle this task. We refer the reader to Sec. 6 for further discussion of this point.

size that combining their scores will improve the resulting model. In Tab. 3 we compare the results of the COMBINED measure to the results of its individual components - PROPORTION and BIGRAM. We compare these results both when using the binary and structured learning algorithms. Results show that using the COMBINED measure leads to an improved performance, better than any of the individual measures, suggesting that it can effectively exploit the properties of each confidence measure. Furthermore, COMBINED is the only sample selection strategy that outperforms ALL EXAMPLES.

(3) Can confidence measures serve as a good proxy for the model's performance? In the unsupervised settings we study the learning process may not converge to an optimal model. We argue that by selecting the model that maximizes the averaged confidence score, a better model can be found. We validate this claim empirically in Tab. 4. We compare the performance of the model selected using the confidence score to the performance of the final model considered by the learning algorithm (see Sec. 3.1 for details). We also compare it to the best model achieved in any of the learning iterations.

Since these experiments required running the learning algorithm many times, we focused on the binary learning algorithm as it converges considerably faster. In order to focus the evaluation on the effects of learning, we ignore the initial model generated manually (INITIAL MODEL) in these experiments. In order to compare models performance across the different iterations fairly, a uniform scale, such as UNIGRAM and BIGRAM, is required. In the case of the COMBINED measure we used the BIGRAM measure for performance estimation, since it is one of its underlying components. In the PRED. SCORE and PROPORTION models we used both their confidence prediction, and the simple UNIGRAM confidence score to evaluate *model* performance (the latter appear in parentheses in Tab. 4).

Results show that the over overall confidence score serves as a reliable proxy for the model performance - using UNIGRAM and BIGRAM the framework can select the best performing model, far better than the performance of the default model to which the system converged.

Algorithm	Supervision	Acc.
INITIAL MODEL	—	0.222
SELF-TRAIN: (*Structured*)		
PRED. SCORE	—	0.348
ALL EXAMPLES	—	0.656
COMBINED	—	**0.664**
SELF-TRAIN: (*Binary*)		
PRED. SCORE	—	0.164
COMBINED	—	0.536
RESPONSE BASED		
BINARY	250 (binary)	0.692
STRUCTURED	250 (binary)	0.732
SUPERVISED		
STRUCTURED	250 (struct.)	0.804

Table 2: Comparing our *Self-trained* systems with Response-based and supervised models. Results show that our COMBINED approach outperforms all other unsupervised models.

Algorithm	Accuracy
SELF-TRAIN: (*Structured*)	
PROPORTION	0.6
BIGRAM	0.644
COMBINED	0.664
SELF-TRAIN: (*Binary*)	
BIGRAM	0.532
PROPORTION	0.504
COMBINED	0.536

Table 3: Comparing COMBINED to its components BIGRAM and PROPORTION. COMBINED results in a better score than any of its components, suggesting that it can exploit the properties of each measure effectively.

Algorithm	Best	Conf. estim.	Default
PRED. SCORE	0.164	0.128 (0.164)	0.134
UNIGRAM	0.52	0.52	0.4
BIGRAM	0.532	0.532	0.472
PROPORTION	0.504	0.27 (0.504)	0.44
COMBINED	0.536	0.536	0.328

Table 4: Using confidence to approximate model performance. We compare the best result obtained in any of the learning algorithm iterations (Best), the result obtained by approximating the best result using the averaged prediction confidence (Conf. estim.) and the result of using the default convergence criterion (Default). Results in parentheses are the result of using the UNIGRAM confidence to approximate the model's performance.

6 Related Work

Semantic parsing has attracted considerable interest in recent years. Current approaches employ various machine learning techniques for this task, such as Inductive Logic Programming in earlier systems (Zelle and Mooney, 1996; Tang and Mooney, 2000) and statistical learning methods in modern ones (Ge and Mooney, 2005; Nguyen et al., 2006; Wong and Mooney, 2006; Kate and Mooney, 2006; Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007; Zettlemoyer and Collins, 2009).

The difficulty of providing the required supervision motivated learning approaches using weaker forms of supervision. (Chen and Mooney, 2008; Liang et al., 2009; Branavan et al., 2009; Titov and Kozhevnikov, 2010) ground NL in an external world state directly referenced by the text. The NL input in our setting is not restricted to such grounded settings and therefore we cannot exploit this form of supervision. Recent work (Clarke et al., 2010; Liang et al., 2011) suggest using response-based learning protocols, which alleviate some of the supervision effort. This work takes an additional step in this direction and suggest an unsupervised protocol.

Other approaches to unsupervised semantic analysis (Poon and Domingos, 2009; Titov and Klementiev, 2011) take a different approach to semantic representation, by clustering semantically equivalent dependency tree fragments, and identifying their predicate-argument structure. While these approaches have been applied successfully to semantic tasks such as question answering, they do not ground the input in a well defined output language, an essential component in our task.

Our unsupervised approach follows a self training protocol (Yarowsky, 1995; McClosky et al., 2006; Reichart and Rappoport, 2007b) enhanced with constraints restricting the output space (Chang et al., 2007; Chang et al., 2009). A Self training protocol uses its own predictions for training. We estimate the quality of the predictions and use only high confidence examples for training. This selection criterion provides an additional view, different than the one used by the prediction model. Multi-view learning is a well established idea, implemented in methods such as co-training (Blum and Mitchell, 1998).

Quality assessment of a learned model output was explored by many previous works (see (Caruana and Niculescu-Mizil, 2006) for a survey), and applied to several NL processing tasks such as syntactic parsing (Reichart and Rappoport, 2007a; Yates et al., 2006), machine translation (Ueffing and Ney, 2007), speech (Koo et al., 2001), relation extraction (Rosenfeld and Feldman, 2007), IE (Culotta and McCallum, 2004), QA (Chu-Carroll et al., 2003) and dialog systems (Lin and Weng, 2008).

In addition to sample selection we use confidence estimation as a way to approximate the overall quality of the model and use it for model selection. This use of confidence estimation was explored in (Reichart et al., 2010), to select between models trained with different random starting points. In this work we integrate this estimation deeper into the learning process, thus allowing our training procedure to return the best performing model.

7 Conclusions

We introduced an unsupervised learning algorithm for semantic parsing, the first for this task to the best of our knowledge. To compensate for the lack of training data we use a self-training protocol, driven by unsupervised confidence estimation. We demonstrate empirically that our approach results in a high preforming semantic parser and show that confidence estimation plays a vital role in this success, both by identifying good training examples as well as identifying good over all performance, used to improve the final model selection.

In future work we hope to further improve unsupervised semantic parsing performance. Particularly, we intend to explore new approaches for confidence estimation and their usage in the unsupervised and semi-supervised versions of the task.

Acknowledgments We thank the anonymous reviewers for their helpful feedback. This material is based upon work supported by DARPA under the Bootstrap Learning Program and Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT*.

S.R.K. Branavan, H. Chen, L. Zettlemoyer, and R. Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *ACL*.

R. Caruana and A. Niculescu-Mizil. 2006. An empirical comparison of supervised l earning algorithms. In *ICML*.

M. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proc. of the Annual Meeting of the ACL*.

M. Chang, D. Goldwasser, D. Roth, and Y. Tu. 2009. Unsupervised constraint driven learning for transliteration discovery. In *NAACL*.

D. Chen and R. Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *ICML*.

J. Chu-Carroll, J. Prager K. Czuba, and A. Ittycheriah. 2003. In question answering, two heads are better than on. In *HLT-NAACL*.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *CoNLL*, 7.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *EMNLP–VLC*.

A. Culotta and A. McCallum. 2004. Confidence estimation for information extraction. In *HLT-NAACL*.

R. Ge and R. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *CoNLL*.

R. Kate and R. Mooney. 2006. Using string-kernels for learning semantic parsers. In *ACL*.

Y. Koo, C. Lee, and B. Juang. 2001. Speech recognition and utterance verification based on a generalized confidence score. *IEEE Transactions on Speech and Audio Processing*, 9(8):821–832.

P. Liang, M. I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *ACL*.

P. Liang, M.I. Jordan, and D. Klein. 2011. Deep compositional semantics from shallow supervision. In *ACL*.

F. Lin and F. Weng. 2008. Computing confidence scores for all sub parse trees. In *ACL*.

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

G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K.J. Miller. 1990. Wordnet: An on-line lexical database. *International Journal of Lexicography*.

L. Nguyen, A. Shimazu, and X. Phan. 2006. Semantic parsing with structured svm ensemble classification models. In *ACL*.

H. Poon and P. Domingos. 2009. Unsupervised semantic parsing. In *EMNLP*.

R. Reichart and A. Rappoport. 2007a. An ensemble method for selection of high quality parses. In *ACL*.

R. Reichart and A. Rappoport. 2007b. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *ACL*.

R. Reichart, R. Fattal, and A. Rappoport. 2010. Improved unsupervised pos induction using intrinsic clustering quality and a zipfian constraint. In *CoNLL*.

B. Rosenfeld and R. Feldman. 2007. Using corpus statistics on entities to improve semi–supervised relation extraction from the web. In *ACL*.

D. Roth and W. Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*.

L. Tang and R. Mooney. 2000. Automated construction of database interfaces: integrating statistical and relational learning for semantic parsing. In *EMNLP*.

L. R. Tang and R. J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *ECML*.

I. Titov and A. Klementiev. 2011. A bayesian model for unsupervised semantic parsing. In *ACL*.

I. Titov and M. Kozhevnikov. 2010. Bootstrapping semantic analyzers from non-contradictory texts. In *ACL*.

N. Ueffing and H. Ney. 2007. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33(1):9–40.

Y.W. Wong and R. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *NAACL*.

Y.W. Wong and R. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *ACL*.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised method. In *ACL*.

A. Yates, S. Schoenmackers, and O. Etzioni. 2006. Detecting parser errors using web-based semantic filters. In *EMNLP*.

J. M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic proramming. In *AAAI*.

L. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*.

L. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *CoNLL*.

L. Zettlemoyer and M. Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *ACL*.

Aspect Ranking: Identifying Important Product Aspects from Online Consumer Reviews

Jianxing Yu, Zheng-Jun Zha, Meng Wang, Tat-Seng Chua
School of Computing
National University of Singapore
{jianxing, zhazj, wangm, chuats}@comp.nus.edu.sg

Abstract

In this paper, we dedicate to the topic of aspect ranking, which aims to automatically identify important product aspects from online consumer reviews. The important aspects are identified according to two observations: (a) the important aspects of a product are usually commented by a large number of consumers; and (b) consumers' opinions on the important aspects greatly influence their overall opinions on the product. In particular, given consumer reviews of a product, we first identify the product aspects by a shallow dependency parser and determine consumers' opinions on these aspects via a sentiment classifier. We then develop an aspect ranking algorithm to identify the important aspects by simultaneously considering the aspect frequency and the influence of consumers' opinions given to each aspect on their overall opinions. The experimental results on 11 popular products in four domains demonstrate the effectiveness of our approach. We further apply the aspect ranking results to the application of document-level sentiment classification, and improve the performance significantly.

1 Introduction

The rapidly expanding e-commerce has facilitated consumers to purchase products online. More than $156 million online product retail sales have been done in the US market during 2009 (Forrester Research, 2009). Most retail Web sites encourage consumers to write reviews to express their opinions on various aspects of the products. This gives rise to huge collections of consumer reviews on the Web. These reviews have become an important resource for both consumers and firms. Consumers commonly seek quality information from online consumer reviews prior to purchasing a product, while many firms use online consumer reviews as an important resource in their product development, marketing, and consumer relationship management. As illustrated in Figure 1, most online reviews express consumers' overall opinion ratings on the product, and their opinions on multiple aspects of the product. While a product may have hundreds of aspects, we argue that some aspects are more important than the others and have greater influence on consumers' purchase decisions as well as firms' product development strategies. Take *iPhone 3GS* as an example, some aspects like "*battery*" and "*speed*," are more important than the others like "*moisture sensor*." Generally, identifying the important product aspects will benefit both consumers and firms. Consumers can conveniently make wise purchase decision by paying attentions on the important aspects, while firms can focus on improving the quality of

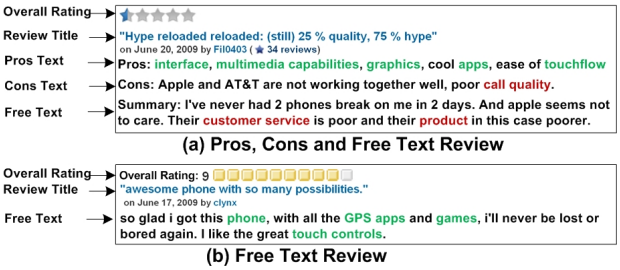


Figure 1: Sample reviews on *iPhone 3GS* product

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1496–1505,
Portland, Oregon, June 19-24, 2011.

these aspects and thus enhance the product reputation effectively. However, it is impractical for people to identify the important aspects from the numerous reviews manually. Thus, it becomes a compelling need to automatically identify the important aspects from consumer reviews.

A straightforward solution for important aspect identification is to select the aspects that are frequently commented in consumer reviews as the important ones. However, consumers' opinions on the frequent aspects may not influence their overall opinions on the product, and thus not influence consumers' purchase decisions. For example, most consumers frequently criticize the bad "*signal connection*" of *iPhone 4*, but they may still give high overall ratings to *iPhone 4*. On the other hand, some aspects, such as "*design*" and "*speed*," may not be frequently commented, but usually more important than "*signal connection*." Hence, the frequency-based solution is not able to identify the truly important aspects.

Motivated by the above observations, in this paper, we propose an effective approach to automatically identify the important product aspects from consumer reviews. Our assumption is that the important aspects of a product should be the aspects that are frequently commented by consumers, and consumers' opinions on the important aspects greatly influence their overall opinions on the product. Given the online consumer reviews of a specific product, we first identify the aspects in the reviews using a shallow dependency parser (Wu et al., 2009), and determine consumers' opinions on these aspects via a sentiment classifier. We then design an aspect ranking algorithm to identify the important aspects by simultaneously taking into account the aspect frequency and the influence of consumers' opinions given to each aspect on their overall opinions. Specifically, we assume that consumer's overall opinion rating on a product is generated based on a weighted sum of his/her specific opinions on multiple aspects of the product, where the weights essentially measure the degree of importance of the aspects. A probabilistic regression algorithm is then developed to derive these importance weights by leveraging the aspect frequency and the consistency between the overall opinions and the weighted sum of opinions on various aspects. We conduct experiments on 11 popular products in four domains. The consumer reviews on these products are crawled from the prevalent forum Web sites (e.g., cnet.com and viewpoint.com etc.) More details of our review corpus are discussed in Section 3. The experimental results demonstrate the effectiveness of our approach on important aspects identification. Furthermore, we apply the aspect ranking results to the application of document-level sentiment classification by carrying out the term-weighting based on the aspect importance. The results show that our approach can improve the performance significantly.

The main contributions of this paper include,

1) We dedicate to the topic of aspect ranking, which aims to automatically identify important aspects of a product from consumer reviews.

2) We develop an aspect ranking algorithm to identify the important aspects by simultaneously considering the aspect frequency and the influence of consumers' opinions given to each aspect on their overall opinions.

3) We apply aspect ranking results to the application of document-level sentiment classification, and improve the performance significantly.

There is another work named aspect ranking (Snyder et al., 2007). The task in this work is different from ours. This work mainly focuses on predicting opinionated ratings on aspects rather than identifying important aspects.

The rest of this paper is organized as follows. Section 2 elaborates our aspect ranking approach. Section 3 presents the experimental results, while Section 4 introduces the application of document-level sentiment classification. Section 5 reviews related work and Section 6 concludes this paper with future works.

2 Aspect Ranking Framework

In this section, we first present some notations and then elaborate the key components of our approach, including the aspect identification, sentiment classification, and aspect ranking algorithm.

2.1 Notations and Problem Formulation

Let $\mathcal{R} = \{r_1, \cdots, r_{|\mathcal{R}|}\}$ denotes a set of online consumer reviews of a specific product. Each review $r \in \mathcal{R}$ is associated with an overall opinion rating

$\mathcal{O}_r$, and covers several aspects with consumer comments on these aspects. Suppose there are m aspects $\mathcal{A} = \{a_1, \cdots, a_m\}$ involved in the review corpus $\mathcal{R}$, where a_k is the k-th aspect. We define o_{rk} as the opinion on aspect a_k in review r. We assume that the overall opinion rating $\mathcal{O}_r$ is generated based on a weighted sum of the opinions on specific aspects o_{rk} (Wang et al., 2010). The weights are denoted as $\{\omega_{rk}\}_{k=1}^{m}$, each of which essentially measures the degree of importance of the aspect a_k in review r. Our task is to derive the important weights of aspects, and identify the important aspects.

Next, we will introduce the key components of our approach, including aspect identification that identifies the aspects a_k in each review r, aspect sentiment classification which determines consumers' opinions o_{rk} on various aspects, and aspect ranking algorithm that identifies the important aspects.

2.2 Aspect Identification

As illustrated in Figure 1, there are usually two types of reviews, *Pros and Cons* review and free text reviews on the Web. For *Pros* and *Cons* reviews, the aspects are identified as the frequent noun terms in the reviews, since the aspects are usually noun or noun phrases (Liu, 2009), and it has been shown that simply extracting the frequent noun terms from the *Pros* and *Cons* reviews can get high accurate aspect terms (Liu el al., 2005). To identify the aspects in free text reviews, we first parse each review using the Stanford parser [1], and extract the noun phrases (*NP*) from the parsing tree as aspect candidates. While these candidates may contain much noise, we leverage the *Pros* and *Cons* reviews to assist identify aspects from the candidates. In particular, we explore the frequent noun terms in *Pros* and *Cons* reviews as features, and train a one-class *SVM* (Manevitz et al., 2002) to identify aspects in the candidates. While the obtained aspects may contain some synonym terms, such as "*earphone*" and "*headphone*," we further perform synonym clustering to get unique aspects. Specifically, we first expand each aspect term with its synonym terms obtained from the synonym terms Web site [2], and then cluster the terms to obtain unique aspects based on unigram feature.

[1] http://nlp.stanford.edu/software/lex-parser.shtml

[2] http://thesaurus.com

2.3 Aspect Sentiment Classification

Since the *Pros* and *Cons* reviews explicitly express positive and negative opinions on the aspects, respectively, our task is to determine the opinions in free text reviews. To this end, we here utilize *Pros* and *Cons* reviews to train a *SVM* sentiment classifier. Specifically, we collect sentiment terms in the *Pros* and *Cons* reviews as features and represent each review into feature vector using Boolean weighting. Note that we select sentiment terms as those appear in the sentiment lexicon provided by *MPQA* project (Wilson et al., 2005). With these features, we then train a *SVM* classifier based on *Pros* and *Cons* reviews. Given a free text review, since it may cover various opinions on multiple aspects, we first locate the opinionated expression modifying each aspect, and determine the opinion on the aspect using the learned *SVM* classifier. In particular, since the opinionated expression on each aspect tends to contain sentiment terms and appear closely to the aspect (Hu and Liu, 2004), we select the expressions which contain sentiment terms and are at the distance of less than 5 from the aspect *NP* in the parsing tree.

2.4 Aspect Ranking

Generally, consumer's opinion on each specific aspect in the review influences his/her overall opinion on the product. Thus, we assume that the consumer gives the overall opinion rating $\mathcal{O}_r$ based on the weighted sum of his/her opinion o_{rk} on each aspect a_k: $\sum_{k=1}^{m} \omega_{rk} o_{rk}$, which can be rewritten as $\boldsymbol{\omega_r}^T \boldsymbol{o_r}$, where $\boldsymbol{\omega_r}$ and $\boldsymbol{o_r}$ are the weight and opinion vectors. Inspired by the work of Wang et al. (2010), we view $\mathcal{O}_r$ as a sample drawn from a *Gaussian Distribution*, with mean $\boldsymbol{\omega_r}^T \boldsymbol{o_r}$ and variance σ^2,

$$p(\mathcal{O}_r) = \frac{1}{\sqrt{2\pi\sigma^2}} exp[-\frac{(\mathcal{O}_r - \boldsymbol{\omega_r}^T \boldsymbol{o_r})^2}{2\sigma^2}]. \quad (1)$$

To model the uncertainty of the importance weights $\boldsymbol{\omega_r}$ in each review, we assume $\boldsymbol{\omega_r}$ as a sample drawn from a *Multivariate Gaussian Distribution*, with $\boldsymbol{\mu}$ as the mean vector and $\boldsymbol{\Sigma}$ as the covariance matrix,

$$p(\boldsymbol{\omega_r}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} exp[-\frac{1}{2}(\boldsymbol{\omega_r} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\omega_r} - \boldsymbol{\mu})]. \quad (2)$$

We further incorporate aspect frequency as a prior knowledge to define the distribution of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. Specifically, the distribution of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ is defined based on its Kullback-Leibler (KL) divergence to a prior distribution with a mean vector $\boldsymbol{\mu_0}$ and an identity covariance matrix $\boldsymbol{I}$ in Eq.3. Each element in $\boldsymbol{\mu_0}$ is defined as the frequency of the corresponding aspect: $frequency(a_k)/\sum_{i=1}^{m} frequency(a_i)$.

$$p(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = exp[-\varphi \cdot KL(Q(\boldsymbol{\mu}, \boldsymbol{\Sigma})||Q(\boldsymbol{\mu_0}, \boldsymbol{I}))], \tag{3}$$

where $KL(\cdot,\cdot)$ is the *KL* divergence, $Q(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a *Multivariate Gaussian Distribution*, and φ is a tradeoff parameter.

Base on the above definition, the probability of generating the overall opinion rating $\mathcal{O}_r$ on review r is given as,

$$\begin{aligned} p(\mathcal{O}_r|\Psi, r) = \int & p(\mathcal{O}_r|\boldsymbol{\omega_r}^T\boldsymbol{o_r}, \sigma^2) \\ & \cdot p(\boldsymbol{\omega_r}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \cdot p(\boldsymbol{\mu}, \boldsymbol{\Sigma}) d\boldsymbol{\omega_r}, \end{aligned} \tag{4}$$

where $\Psi = \{\boldsymbol{\omega}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \sigma^2\}$ are the model parameters.

Next, we utilize Maximum Log-likelihood (*ML*) to estimate the model parameters given the consumer reviews corpus. In particular, we aim to find an optimal $\hat{\Psi}$ to maximize the probability of observing the overall opinion ratings in the reviews corpus.

$$\begin{aligned} \hat{\Psi} &= \arg\max_{\Psi} \sum_{r\in\mathcal{R}} log(p(\mathcal{O}_r|\Psi, r)) \\ &= \arg\min_{\Psi} (|\mathcal{R}| - 1)\log\det(\boldsymbol{\Sigma}) + \sum_{r\in\mathcal{R}} [\log\sigma^2 + \\ &\frac{(\mathcal{O}_r - \boldsymbol{\omega_r}^T\boldsymbol{o_r})^2}{\sigma^2} + (\boldsymbol{\omega_r} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\omega_r} - \boldsymbol{\mu})] + \\ &(tr(\boldsymbol{\Sigma}) + (\boldsymbol{\mu_0} - \boldsymbol{\mu})^T\boldsymbol{I}(\boldsymbol{\mu_0} - \boldsymbol{\mu})). \end{aligned} \tag{5}$$

For the sake of simplicity, we denote the objective function $\sum_{r\in\mathcal{R}} log(p(\mathcal{O}_r|\Psi, r))$ as $\Gamma(\Psi)$.

The derivative of the objective function with respect to each model parameter vanishes at the minimizer:

$$\begin{aligned} \frac{\partial\Gamma(\Psi)}{\partial\boldsymbol{\omega_r}} &= -\frac{(\boldsymbol{\omega_r}^T\boldsymbol{o_r} - \mathcal{O}_r)\boldsymbol{o_r}}{\sigma^2} - \boldsymbol{\Sigma}^{-1}(\boldsymbol{\omega_r} - \boldsymbol{\mu}) \\ &= 0; \end{aligned} \tag{6}$$

$$\begin{aligned} \frac{\partial\Gamma(\Psi)}{\partial\boldsymbol{\mu}} &= \sum_{r\in\mathcal{R}} [-\boldsymbol{\Sigma}^{-1}(\boldsymbol{\omega_r} - \boldsymbol{\mu})] - \varphi \cdot \boldsymbol{I}(\boldsymbol{\mu_0} - \boldsymbol{\mu}) \\ &= 0; \end{aligned} \tag{7}$$

$$\begin{aligned} \frac{\partial\Gamma(\Psi)}{\partial\boldsymbol{\Sigma}} &= \sum_{r\in\mathcal{R}} \{-(\boldsymbol{\Sigma}^{-1})^T - [-(\boldsymbol{\Sigma}^{-1})^T(\boldsymbol{\omega_r} - \boldsymbol{\mu}) \\ &(\boldsymbol{\omega_r} - \boldsymbol{\mu})^T(\boldsymbol{\Sigma}^{-1})^T]\} + \varphi \cdot \left[(\boldsymbol{\Sigma}^{-1})^T - \boldsymbol{I}\right] = 0; \end{aligned} \tag{8}$$

$$\begin{aligned} \frac{\partial\Gamma(\Psi)}{\partial\sigma^2} &= \sum_{r\in\mathcal{R}} (-\frac{1}{\sigma^2} + \frac{(\mathcal{O}_r - \boldsymbol{\omega_r}^T\boldsymbol{o_r})^2}{\sigma^4}) = 0, \end{aligned} \tag{9}$$

which lead to the following solutions:

$$\hat{\boldsymbol{\omega}}_{\boldsymbol{r}} = (\frac{\boldsymbol{o_r}\boldsymbol{o_r}^T}{\sigma^2} + \boldsymbol{\Sigma}^{-1})^{-1}(\frac{\mathcal{O}_r\boldsymbol{o_r}}{\sigma^2} + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}); \tag{10}$$

$$\hat{\boldsymbol{\mu}} = (|\mathcal{R}|\boldsymbol{\Sigma}^{-1} + \varphi \cdot \boldsymbol{I})^{-1}(\boldsymbol{\Sigma}^{-1}\sum_{r\in\mathcal{R}} \boldsymbol{\omega_r} + \varphi \cdot \boldsymbol{I}\boldsymbol{\mu_0}); \tag{11}$$

$$\begin{aligned} \hat{\boldsymbol{\Sigma}} = \{&[\frac{1}{\varphi}\sum_{r\in\mathcal{R}} \left[(\boldsymbol{\omega_r} - \boldsymbol{\mu})(\boldsymbol{\omega_r} - \boldsymbol{\mu})^T\right] + \\ &(\frac{|\mathcal{R}| - \varphi}{2\varphi})^2\boldsymbol{I}]^{1/2} - \frac{(|\mathcal{R}| - \varphi)}{2\varphi}\boldsymbol{I}\}^T; \end{aligned} \tag{12}$$

$$\hat{\sigma}^2 = \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} (\mathcal{O}_r - \boldsymbol{\omega_r}^T\boldsymbol{o_r})^2. \tag{13}$$

We can see that the above parameters are involved in each other's solution. We here utilize *Alternating Optimization* technique to derive the optimal parameters in an iterative manner. We first hold the parameters $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ and σ^2 fixed and update the parameters $\boldsymbol{\omega_r}$ for each review $r \in \mathcal{R}$. Then, we update the parameters $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ and σ^2 with fixed $\boldsymbol{\omega_r}$ ($r \in \mathcal{R}$). These two steps are alternatively iterated until the Eq.5 converges. As a result, we obtain the optimal importance weights $\boldsymbol{\omega_r}$ which measure the importance of aspects in review $r \in \mathcal{R}$. We then compute the final importance score ϖ_k for each aspect a_k by integrating its importance score in all the reviews as,

$$\varpi_k = \frac{1}{|\mathcal{R}|}\sum\nolimits_{r\in\mathcal{R}} \omega_{rk}, \quad k = 1, \cdots, m \tag{14}$$

It is worth noting that the aspect frequency is considered again in this integration process. According to the importance score ϖ_k, we can identify important aspects.

3 Evaluations

In this section, we evaluate the effectiveness of our approach on aspect identification, sentiment classification, and aspect ranking.

3.1 Data and Experimental Setting

The details of our product review data set is given in Table 1. This data set contains consumer reviews on 11 popular products in 4 domains. These reviews were crawled from the prevalent forum Web sites, including cnet.com, viewpoints.com, reevoo.com and gsmarena.com. All of the reviews were posted

between June, 2009 and Sep 2010. The aspects of the reviews, as well as the opinions on the aspects were manually annotated as the gold standard for evaluations.

Product Name	*Domain*	*Review#*	*Sentence#*
Canon EOS 450D (Canon EOS)	camera	440	628
Fujifilm Finepix AX245W (Fujifilm)	camera	541	839
Panasonic Lumix DMC-TZ7 (Panasonic)	camera	650	1,546
Apple MacBook Pro (MacBook)	laptop	552	4,221
Samsung NC10 (Samsung)	laptop	2,712	4,946
Apple iPod Touch 2nd (iPod Touch)	MP3	4,567	10,846
Sony NWZ-S639 16GB (Sony NWZ)	MP3	341	773
BlackBerry Bold 9700 (BlackBerry)	phone	4,070	11,008
iPhone 3GS 16GB (iPhone 3GS)	phone	12,418	43,527
Nokia 5800 XpressMusic (Nokia 5800)	phone	28,129	75,001
Nokia N95	phone	15,939	44,379

Table 1: Statistics of the Data Sets, # denotes the size of the reviews/sentences.

To examine the performance on aspect identification and sentiment classification, we employed F_1-measure, which was the combination of *precision* and *recall*, as the evaluation metric. To evaluate the performance on aspect ranking, we adopted *Normalized Discounted Cumulative Gain* at top *k* (NDCG@*k*) (Jarvelin and Kekalainen, 2002) as the performance metric. Given an aspect ranking list $a_1, \cdots, a_k$, NDCG@*k* is calculated by

$$NDCG@k = \frac{1}{Z}\sum_{i=1}^{k}\frac{2^{t(i)}-1}{\log(1+i)}, \qquad (15)$$

where $t(i)$ is the function that represents the reward given to the aspect at position i, Z is a normalization term derived from the top k aspects of a perfect ranking, so as to normalize NDCG@*k* to be within $[0, 1]$. This evaluation metric will favor the ranking which ranks the most important aspects at the top. For the reward $t(i)$, we labeled each aspect as one of the three scores: *Un-important* (score 1), *Ordinary* (score 2) and *Important* (score 3). Three volunteers were invited in the annotation process as follows. We first collected the top *k* aspects in all the rankings produced by various evaluated methods (maximum *k* is 15 in our experiment). We then sampled some reviews covering these aspects, and provided the reviews to each annotator to read. Each review contains the overall opinion rating, the highlighted aspects, and opinion terms. Afterward, the annotators were required to assign an importance score to each aspect. Finally, we took the average of their scorings as the corresponding importance scores of the aspects. In addition, there is only one parameter φ that needs to be tuned in our approach. Throughout the experiments, we empirically set φ as 0.001.

3.2 Evaluations on Aspect Identification

We compared our aspect identification approach against two baselines: a) the method proposed by Hu and Liu (2004), which was based on the association rule mining, and b) the method proposed by Wu et al. (2009), which was based on a dependency parser.

The results are presented in Table 2. On average, our approach significantly outperforms Hu's method and Wu' method in terms of F_1-measure by over 5.87% and 3.27%, respectively. In particular, our approach obtains high precision. Such results imply that our approach can accurately identify the aspects from consumer reviews by leveraging the *Pros* and *Cons* reviews.

Data set	*Hu's Method*	*Wu's Method*	*Our Method*
Canon EOS	0.681	0.686	**0.728**
Fujifilm	0.685	0.666	**0.710**
Panasonic	0.636	0.661	**0.706**
MacBook	0.680	0.733	**0.747**
Samsung	0.594	0.631	**0.712**
iPod Touch	0.650	0.660	**0.718**
Sony NWZ	0.631	0.692	**0.760**
BlackBerry	0.721	0.730	**0.734**
iPhone 3GS	0.697	0.736	**0.740**
Nokia 5800	0.715	0.745	**0.747**
Nokia N95	0.700	0.737	**0.741**

Table 2: Evaluations on Aspect Identification. * significant t-test, p-values$<$0.05.

3.3 Evaluations on Sentiment Classification

In this experiment, we implemented the following sentiment classification methods (Pang and Lee, 2008):

1) Unsupervised method. We employed one unsupervised method which was based on opinionated term counting via *SentiWordNet* (Ohana et al., 2009).

2) Supervised method. We employed three supervised methods proposed in Pang et al. (2002), including Naïve Bayes (*NB*), Maximum Entropy (*ME*), *SVM*. These classifiers were trained based on the *Pros* and *Cons* reviews as described in Section 2.3.

The comparison results are showed in Table 3. We can see that supervised methods significantly outperform unsupervised method. For example, the *SVM* classifier outperforms the unsupervised method in terms of average F_1-measure by over 10.37%. Thus, we can deduce from such results that the *Pros* and *Cons* reviews are useful for sentiment classification. In addition, among the supervised classifiers, *SVM* classifier performs the best in most products, which is consistent with the previous research (Pang et al., 2002).

Data set	*Senti*	*NB*	*SVM*	*ME*
Canon EOS	0.628	0.720	**0.739**	0.726
Fujifilm	0.690	0.781	**0.791**	0.778
Panasonic	0.625	0.694	**0.719**	0.697
MacBook	0.708	0.820	**0.828**	0.797
Samsung	0.675	**0.723**	0.717	0.714
iPod Touch	0.711	0.792	**0.805**	0.791
Sony NWZ	0.621	0.722	**0.737**	0.725
BlackBerry	0.699	**0.819**	0.794	0.788
iPhone 3GS	0.717	0.811	**0.829**	0.822
Nokia 5800	0.736	0.840	**0.851**	0.817
Nokia N95	0.706	0.829	**0.849**	0.826

Table 3: Evaluations on Sentiment Classification. Senti denotes the method based on SentiWordNet. * significant t-test, p-values$<$0.05.

3.4 Evaluations on Aspect Ranking

In this section, we compared our aspect ranking algorithm against the following three methods.

1) Frequency-based method. The method ranks the aspects based on aspect frequency.

2) Correlation-based method. This method measures the correlation between the opinions on specific aspects and the overall opinion. It counts the number of the cases when such two kinds of opinions are consistent, and ranks the aspects based on the number of the consistent cases.

3) Hybrid method. This method captures both the aspect frequency and correlation by a linear combination, as $\lambda\cdot$ *Frequency-based Ranking* $+ (1-\lambda)\cdot$ *Correlation-based Ranking*, where λ is set to 0.5.

The comparison results are showed in Table 4. On average, our approach outperforms the frequency-based method, correlation-based method, and hybrid method in terms of NDCG@*5* by over 6.24%, 5.79% and 5.56%, respectively. It improves the performance over such three methods in terms of NDCG@*10* by over 3.47%, 2.94% and 2.58%, respectively, while in terms of NDCG@*15* by over 4.08%, 3.04% and 3.49%, respectively. We can deduce from the results that our aspect ranking algorithm can effectively identify the important aspects from consumer reviews by leveraging the aspect frequency and the influence of consumers' opinions given to each aspect on their overall opinions. Table 5 shows the aspect ranking results of these four methods. Due to the space limitation, we here only show top 10 aspects of the product *iphone 3GS*. We can see that our approach performs better than the others. For example, the aspect "*phone*" is ranked at the top by the other methods. However, "*phone*" is a general but not important aspect.

#	*Frequency*	*Correlated*	*Hybrid*	*Our Method*
1	Phone	Phone	Phone	Usability
2	Usability	Usability	Usability	Apps
3	3G	Apps	Apps	3G
4	Apps	3G	3G	Battery
5	Camera	Camera	Camera	Looking
6	Feature	Looking	Looking	Storage
7	Looking	Feature	Feature	Price
8	Battery	Screen	Battery	Software
9	Screen	Battery	Screen	Camera
10	Flash	Bluetooth	Flash	Call quality

Table 5: iPhone 3GS Aspect Ranking Results.

To further investigate the reasonability of our ranking results, we refer to one of the public user feedback reports, the "*china unicom 100 customers iPhone user feedback report*" (Chinaunicom Report, 2009). The report demonstrates that the top four aspects of *iPhone* product, which users most concern with, are "*3G Network*" (30%), "*usability*" (30%), "*out-looking design*" (26%), "*application*" (15%). All of these aspects are in the top 10 of our ranking results.

Therefore, we can conclude that our approach is able to automatically identify the important aspects from numerous consumer reviews.

4 Applications

The identification of important aspects can support a wide range of applications. For example, we can

	Frequency			*Correlation*			*Hybrid*			*Our Method*		
Data set	*@5*	*@10*	*@15*	*@5*	*@10*	*@15*	*@5*	*@10*	*@15*	*@5*	*@10*	*@15*
Canon EOS	0.735	0.771	0.740	0.735	0.762	0.779	0.735	0.798	0.742	**0.862**	**0.824**	**0.794**
Fujifilm	0.816	0.705	0.693	0.760	0.756	0.680	0.816	0.759	0.682	**0.863**	**0.801**	**0.760**
Panasonic	0.744	0.807	0.783	0.763	0.815	0.792	0.744	0.804	0.786	**0.796**	**0.834**	**0.815**
MacBook	0.744	0.771	0.762	0.763	0.746	0.769	0.763	**0.785**	**0.772**	**0.874**	0.776	0.760
Samsung	0.964	0.765	0.794	0.964	0.820	0.840	0.964	0.820	0.838	**0.968**	**0.826**	**0.854**
iPod Touch	0.836	0.830	0.727	0.959	**0.851**	0.744	0.948	0.785	0.733	**0.959**	0.817	**0.801**
Sony NWZ	0.937	0.743	0.742	0.937	**0.781**	0.797	0.937	0.740	0.794	**0.944**	0.775	**0.815**
BlackBerry	0.837	0.824	0.766	0.847	0.825	0.771	0.847	**0.829**	0.768	**0.874**	0.797	**0.779**
iPhone 3GS	0.897	0.836	0.832	0.886	0.814	0.825	0.886	0.829	0.826	**0.948**	**0.902**	**0.860**
Nokia 5800	0.834	0.779	0.796	0.834	0.781	0.779	0.834	0.781	0.779	**0.903**	**0.811**	**0.814**
Nokia N95	0.675	0.680	0.717	0.619	0.619	0.691	0.619	0.678	0.696	**0.716**	**0.731**	**0.748**

Table 4: Evaluations on Aspect Ranking. @5, @10, @15 denote the evaluation metrics of NDCG@*5*, NDCG@*10*, and NDCG@*15*, respectively. * significant t-test, p-values<0.05.

provide product comparison on the important aspects to users, so that users can make wise purchase decisions conveniently.

In the following, we apply the aspect ranking results to assist document-level review sentiment classification. Generally, a review document contains consumer's positive/negative opinions on various aspects of the product. It is difficult to get the accurate overall opinion of the whole review without knowing the importance of these aspects. In addition, when we learn a document-level sentiment classifier, the features generated from unimportant aspects lack of discriminability and thus may deteriorate the performance of the classifier (Fang et al., 2010). While the important aspects and the sentiment terms on these aspects can greatly influence the overall opinions of the review, they are highly likely to be discriminative features for sentiment classification. These observations motivate us to utilize aspect ranking results to assist classifying the sentiment of review documents.

Specifically, we randomly sampled 100 reviews of each product as the testing data and used the remaining reviews as the training data. We first utilized our approach to identify the importance aspects from the training data. We then explored the aspect terms and sentiment terms as features, based on which each review is represented as a feature vector. Here, we give more emphasis on the important aspects and the sentiment terms that modify these aspects. In particular, we set the term-weighting as $1 + \varphi \cdot \varpi_k$, where ϖ_k is the importance score of the aspect a_k, φ is set to 100. Based on the weighted features, we then trained a *SVM* classifier using the training reviews to determine the overall opinions on the testing reviews. For the performance comparison, we compared our approach against two baselines, including Boolean weighting method and frequency weighting (*tf*) method (Paltoglou et al., 2010) that do not utilize the importance of aspects. The comparison results are shown in Table 6. We can see that our approach (*IA*) significantly outperforms the other methods in terms of average F_1-measure by over 2.79% and 4.07%, respectively. The results also show that the Boolean weighting method outperforms the frequency weighting method in terms of average F_1-measure by over 1.25%, which are consistent with the previous research by Pang et al. (2002). On the other hand, from the *IA* weighting formula, we observe that without using the important aspects, our term-weighting function will be equal to Boolean weighting. Thus, we can speculate that the identification of important aspects is beneficial to improving the performance of document-level sentiment classification.

5 Related Work

Existing researches mainly focused on determining opinions on the reviews, or identifying aspects from these reviews. They viewed each aspect equally without distinguishing the important ones. In this section, we review existing researches related to our work.

Analysis of the opinion on whole review text had

	$SVM + Boolean$			$SVM + tf$			$SVM + IA$		
Data set	P	R	F_1	P	R	F_1	P	R	F_1
Canon EOS	0.689	0.663	0.676	0.679	0.654	0.666	0.704	0.721	**0.713**
Fujifilm	0.700	0.687	0.693	0.690	0.670	0.680	0.731	0.724	**0.727**
Panasonic	0.659	0.717	0.687	0.650	0.693	0.671	0.696	0.713	**0.705**
MacBook	0.744	0.700	0.721	0.768	0.675	0.718	0.790	0.717	**0.752**
Samsung	0.755	0.690	0.721	0.716	0.725	0.720	0.732	0.765	**0.748**
iPod Touch	0.686	0.746	0.714	0.718	0.667	0.691	0.749	0.726	**0.737**
Sony NWZ	0.719	0.652	0.684	0.665	0.646	0.655	0.732	0.684	**0.707**
BlackBerry	0.763	0.719	0.740	0.752	0.709	0.730	0.782	0.758	**0.770**
iPhone 3GS	0.777	0.775	0.776	0.772	0.762	0.767	0.820	0.788	**0.804**
Nokia 5800	0.755	0.836	0.793	0.744	0.815	0.778	0.805	0.821	**0.813**
Nokia N95	0.722	0.699	0.710	0.695	0.708	0.701	0.768	0.732	**0.750**

Table 6: Evaluations on Term Weighting methods for Document-level Review Sentiment Classification. *IA* denotes the term weighing based on the important aspects. * significant t-test, p-values$<$0.05.

been extensively studied (Pang and Lee, 2008). Earlier research had been studied unsupervised (Kim et al., 2004), supervised (Pang et al., 2002; Pang et al., 2005) and semi-supervised approaches (Goldberg et al., 2006) for the classification. For example, Mullen et al. (2004) proposed an unsupervised classification method which exploited pointwise mutual information (*PMI*) with syntactic relations and other attributes. Pang et al. (2002) explored several machine learning classifiers, including Naïve Bayes, Maximum Entropy, SVM, for sentiment classification. Goldberg et al. (2006) classified the sentiment of the review using the graph-based semi-supervised learning techniques, while Li el al. (2009) tackled the problem using matrix factorization techniques with lexical prior knowledge.

Since the consumer reviews usually expressed opinions on multiple aspects, some works had drilled down to the aspect-level sentiment analysis, which aimed to identify the aspects from the reviews and to determine the opinions on the specific aspects instead of the overall opinion. For the topic of aspect identification, Hu and Liu (2004) presented the association mining method to extract the frequent terms as the aspects. Subsequently, Popescu et al. (2005) proposed their system *OPINE*, which extracted the aspects based on the *KnowItAll* Web information extraction system (Etzioni et al., 2005). Liu el al. (2005) proposed a supervised method based on language pattern mining to identify the aspects in the reviews. Later, Mei et al. (2007) proposed a probabilistic topic model to capture the mixture of aspects and sentiments simultaneously. Afterwards, Wu et al. (2009) utilized the dependency parser to extract the noun phrases and verb phrases from the reviews as the aspect candidates. They then trained a language model to refine the candidate set, and to obtain the aspects. On the other hand, for the topic of sentiment classification on the specific aspect, Snyder et al. (2007) considered the situation when the consumers' opinions on one aspect could influence their opinions on others. They thus built a graph to analyze the meta-relations between opinions, such as agreement and contrast. And they proposed a Good Grief algorithm to leveraging such meta-relations to improve the prediction accuracy of aspect opinion ratings. In addition, Wang et al. (2010) proposed the topic of latent aspect rating which aimed to infer the opinion rating on the aspect. They first employed a bootstrapping-based algorithm to identify the major aspects via a few seed word aspects. They then proposed a generative Latent Rating Regression model (LRR) to infer aspect opinion ratings based on the review content and the associated overall rating.

While there were usually huge collection of reviews, some works had concerned the topic of aspect-based sentiment summarization to combat the information overload. They aimed to summarize all the reviews and integrate major opinions on various aspects for a given product. For example, Titov et al. (2008) explored a topic modeling method to generate a summary based on multiple aspects. They utilized topics to describe aspects and incor-

porated a regression model fed by the ground-truth opinion ratings. Additionally, Lu el al. (2009) proposed a structured PLSA method, which modeled the dependency structure of terms, to extract the aspects in the reviews. They then aggregated opinions on each specific aspects and selected representative text segment to generate a summary.

In addition, some works proposed the topic of product ranking which aimed to identify the best products for each specific aspect (Zhang et al., 2010). They used a PageRank style algorithm to mine the aspect-opinion graph, and to rank the products for each aspect.

Different from previous researches, we dedicate our work to identifying the important aspects from the consumer reviews of a specific product.

6 Conclusions and Future Works

In this paper, we have proposed to identify the important aspects of a product from online consumer reviews. Our assumption is that the important aspects of a product should be the aspects that are frequently commented by consumers and consumers' opinions on the important aspects greatly influence their overall opinions on the product. Based on this assumption, we have developed an aspect ranking algorithm to identify the important aspects by simultaneously considering the aspect frequency and the influence of consumers' opinions given to each aspect on their overall opinions. We have conducted experiments on 11 popular products in four domains. Experimental results have demonstrated the effectiveness of our approach on important aspects identification. We have further applied the aspect ranking results to the application of document-level sentiment classification, and have significantly improved the classification performance. In the future, we will apply our approach to support other applications.

Acknowledgments

This work is supported in part by NUS-Tsinghua Extreme Search (NExT) project under the grant number: R-252-300-001-490. We give warm thanks to the project and anonymous reviewers for their comments.

References

P. Beineke, T. Hastie, C. Manning, and S. Vaithyanathan. An Exploration of Sentiment Summarization. *AAAI*, 2003.

G. Carenini, R.T. Ng, and E. Zwart. Extracting Knowledge from Evaluative Text. *K-CAP*, 2005.

G. Carenini, R.T. Ng, and E. Zwart. Multi-document Summarization of Evaluative Text. *ACL*, 2006.

China Unicom 100 Customers iPhone User Feedback Report, 2009.

Y. Choi and C. Cardie. Hierarchical Sequential Learning for Extracting Opinions and Their Attributes. *ACL*, 2010.

H. Cui, V. Mittal, and M. Datar. Comparative Experiments on Sentiment Classification for Online Product Reviews. *AAAI*, 2006.

S. Dasgupta and V. Ng. Mine the Easy, Classify the Hard: A Semi-supervised Approach to Automatic Sentiment Classification. *ACL*, 2009.

K. Dave, S. Lawrence, and D.M. Pennock. Opinion Extraction and Semantic Classification of Product Reviews. *WWW*, 2003.

A. Esuli and F. Sebastiani. A Publicly Available Lexical Resource for Opinion Mining. *LREC*, 2006.

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Unsupervised Named-entity Extraction from the Web: An Experimental Study. *Artificial Intelligence*, 2005.

J. Fang, B. Price, and L. Price. Pruning Non-Informative Text Through Non-Expert Annotations to Improve Aspect-Level Sentiment Classification. *COLING*, 2010.

O. Feiguina and G. Lapalme. Query-based Summarization of Customer Reviews. *AI*, 2007.

Forrester Research. State of Retailing Online 2009: Marketing Report. *http://www.shop.org/soro*, 2009.

A. Goldberg and X. Zhu. Seeing Stars when There aren't Many Stars: Graph-based Semi-supervised Learning for Sentiment Categorization. *ACL*, 2006.

M. Gamon, A. Aue, S. Corston-Oliver, and E. Ringger. Pulse: Mining Customer Opinions from Free Text. *IDA*, 2005.

M. Hu and B. Liu. Mining and Summarizing Customer Reviews. *SIGKDD*, 2004.

K. Jarvelin and J. Kekalainen. Cumulated Gain-based Evaluation of IR Techniques. *TOIS*, 2002.

S. Kim and E. Hovy. Determining the Sentiment of Opinions. *COLING*, 2004.

J. Kim, J.J. Li, and J.H. Lee. Discovering the Discriminative Views: Measuring Term Weights for Sentiment Analysis. *ACL*, 2009.

Kelsey Research and comscore. Online Consumer-Generated Reviews Have Significant Impact on Offline Purchase Behavior.

K. Lerman, S. Blair-Goldensohn, and R. McDonald. Sentiment Summarization: Evaluating and Learning User Preferences. *EACL*, 2009.

B. Li, L. Zhou, S. Feng, and K.F. Wong. A Unified Graph Model for Sentence-Based Opinion Retrieval. *ACL*, 2010.

T. Li and Y. Zhang, and V. Sindhwani. A Non-negative Matrix Tri-factorization Approach to Sentiment Classification with Lexical Prior Knowledge. *ACL*, 2009.

B. Liu, M. Hu, and J. Cheng. Opinion Observer: Analyzing and Comparing Opinions on the Web. *WWW*, 2005.

B. Liu. Handbook Chapter: Sentiment Analysis and Subjectivity. Handbook of Natural Language Processing. *Marcel Dekker, Inc. New York, NY, USA*, 2009.

Y. Lu, C. Zhai, and N. Sundaresan. Rated Aspect Summarization of Short Comments. *WWW*, 2009.

L.M. Manevitz and M. Yousef. One-class svms for Document Classification. *The Journal of Machine Learning*, 2002.

R. McDonal, K. Hannan, T. Neylon, M. Wells, and J. Reynar. Structured Models for Fine-to-coarse Sentiment Analysis. *ACL*, 2007.

Q. Mei, X. Ling, M. Wondra, H. Su, and C.X. Zhai. Topic Sentiment Mixture: Modeling Facets and Opinions in Weblogs. *WWW*, 2007.

H.J. Min and J.C. Park. Toward Finer-grained Sentiment Identification in Product Reviews Through Linguistic and Ontological Analyses. *ACL*, 2009.

T. Mullen and N. Collier. Sentiment Analysis using Support Vector Machines with Diverse Information Sources. *EMNLP*, 2004.

N. Nanas, V. Uren, and A.D. Roeck. Building and Applying a Concept Hierarchy Representation of a User Profile. *SIGIR*, 2003.

H. Nishikawa, T. Hasegawa, Y. Matsuo, and G. Kikui. Optimizing Informativeness and Readability for Sentiment Summarization. *ACL*, 2010.

B. Ohana and B. Tierney. Sentiment Classification of Reviews Using SentiWordNet. *IT&T Conference*, 2009.

G. Paltoglou and M. Thelwall. A study of Information Retrieval Weighting Schemes for Sentiment Analysis. *ACL*, 2010.

B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques. *EMNLP*, 2002.

B. Pang, L. Lee, and S. Vaithyanathan. A Sentimental Education: Sentiment Analysis using Subjectivity Summarization based on Minimum cuts Techniques. *ACL*, 2004.

B. Pang and L. Lee. Seeing stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. *ACL*, 2005.

B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2008.

A.-M. Popescu and O. Etzioni. Extracting Product Features and Opinions from Reviews. *HLT/EMNLP*, 2005.

R. Prabowo and M. Thelwall. Sentiment analysis: A Combined Approach. *Journal of Informetrics*, 2009.

G. Qiu, B. Liu, J. Bu, and C. Chen.. Expanding Domain Sentiment Lexicon through Double Propagation. *IJCAI*, 2009.

M. Sanderson and B. Croft. Document-word Co-regularization for Semi-supervised Sentiment Analysis. *ICDM*, 2008.

B. Snyder and R. Barzilay. Multiple Aspect Ranking using the Good Grief Algorithm. *NAACL HLT*, 2007.

S. Somasundaran, G. Namata, L. Getoor, and J. Wiebe. Opinion Graphs for Polarity and Discourse Classification. *ACL*, 2009.

Q. Su, X. Xu, H. Guo, X. Wu, X. Zhang, B. Swen, and Z. Su. Hidden Sentiment Association in Chinese Web Opinion Mining. *WWW*, 2008.

C. Toprak, N. Jakob, and I. Gurevych. Sentence and Expression Level Annotation of Opinions in User-Generated Discourse. *ACL*, 2010.

P. Turney. Thumbs up or Thumbs down? Semantic Orientation Applied to Unsupervised Classification of Reviews. *ACL*, 2002.

I. Titov and R. McDonald. A Joint Model of Text and Aspect Ratings for Sentiment Summarization. *ACL*, 2008.

H. Wang, Y. Lu, and C.X. Zhai. Latent Aspect Rating Analysis on Review Text Data: A Rating Regression Approach. *KDD*, 2010.

B. Wei and C. Pal. Cross Lingual Adaptation: An Experiment on Sentiment Classifications. *ACL*, 2010.

T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing Contextual Polarity in Phrase-level Sentiment Analysis. *HLT/EMNLP*, 2005.

T. Wilson and J. Wiebe. Annotating Attributions and Private States. *ACL*, 2005.

Y. Wu, Q. Zhang, X. Huang, and L. Wu. Phrase Dependency Parsing for Opinion Mining. *ACL*, 2009.

K. Zhang, R. Narayanan, and A. Choudhary. Voice of the Customers: Mining Online Customer Reviews for Product Feature-based Ranking. *WOSN*, 2010.

J. Zhu, H. Wang, and B.K. Tsou. Aspect-based Sentence Segmentation for Sentiment Summarization. *TSA*, 2009.

L. Zhuang, F. Jing, and X.Y. Zhu. Movie Review Mining and Summarization. *CIKM*, 2006.

Collective Classification of Congressional Floor-Debate Transcripts

Clinton Burfoot, Steven Bird and **Timothy Baldwin**
Department of Computer Science and Software Engineering
University of Melbourne, VIC 3010, Australia
{cburfoot, sb, tim}@csse.unimelb.edu.au

Abstract

This paper explores approaches to sentiment classification of U.S. Congressional floor-debate transcripts. Collective classification techniques are used to take advantage of the informal citation structure present in the debates. We use a range of methods based on local and global formulations and introduce novel approaches for incorporating the outputs of machine learners into collective classification algorithms. Our experimental evaluation shows that the mean-field algorithm obtains the best results for the task, significantly outperforming the benchmark technique.

1 Introduction

Supervised document classification is a well-studied task. Research has been performed across many document types with a variety of classification tasks. Examples are topic classification of newswire articles (Yang and Liu, 1999), sentiment classification of movie reviews (Pang et al., 2002), and satire classification of news articles (Burfoot and Baldwin, 2009). This and other work has established the usefulness of document classifiers as stand-alone systems and as components of broader NLP systems.

This paper deals with methods relevant to supervised document classification in domains with *network* structures, where *collective classification* can yield better performance than approaches that consider documents in isolation. Simply put, a network structure is any set of relationships between documents that can be used to assist the document classification process. Web encyclopedias and scholarly publications are two examples of document domains where network structures have been used to assist classification (Gantner and Schmidt-Thieme, 2009; Cao and Gao, 2005).

The contribution of this research is in four parts: (1) we introduce an approach that gives better than state of the art performance for collective classification on the ConVote corpus of congressional debate transcripts (Thomas et al., 2006); (2) we provide a comparative overview of collective document classification techniques to assist researchers in choosing an algorithm for collective document classification tasks; (3) we demonstrate effective novel approaches for incorporating the outputs of SVM classifiers into collective classifiers; and (4) we demonstrate effective novel feature models for iterative local classification of debate transcript data.

In the next section (Section 2) we provide a formal definition of collective classification and describe the ConVote corpus that is the basis for our experimental evaluation. Subsequently, we describe and critique the established benchmark approach for congressional floor-debate transcript classification, before describing approaches based on three alternative collective classification algorithms (Section 3). We then present an experimental evaluation (Section 4). Finally, we describe related work (Section 5) and offer analysis and conclusions (Section 6).

2 Task Definition

2.1 Collective Classification

Given a network and an object o in the network, there are three types of correlations that can be used

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1506–1515,
Portland, Oregon, June 19-24, 2011.

to infer a label for o: (1) the correlations between the label of o and its observed attributes; (2) the correlations between the label of o and the observed attributes and labels of nodes connected to o; and (3) the correlations between the label of o and the unobserved labels of objects connected to o (Sen et al., 2008).

Standard approaches to classification generally ignore any network information and only take into account the correlations in (1). Each object is classified as an individual instance with features derived from its observed attributes. Collective classification takes advantage of the network by using all three sources. Instances may have features derived from their source objects or from other objects. Classification proceeds in a joint fashion so that the label given to each instance takes into account the labels given to all of the other instances.

Formally, collective classification takes a graph, made up of nodes $\mathcal{V} = \{V_1, \ldots, V_n\}$ and edges E. The task is to label the nodes $V_i \in \mathcal{V}$ from a label set $\mathcal{L} = \{L_1, \ldots, L_q\}$, making use of the graph in the form of a neighborhood function $\mathcal{N} = \{N_1, \ldots, N_n\}$, where $N_i \subseteq \mathcal{V} \setminus \{V_i\}$.

2.2 The ConVote Corpus

ConVote, compiled by Thomas et al. (2006), is a corpus of U.S. congressional debate transcripts. It consists of 3,857 speeches organized into 53 debates on specific pieces of legislation. Each speech is tagged with the identity of the speaker and a "for" or "against" label derived from congressional voting records. In addition, places where one speaker cites another have been annotated, as shown in Figure 1.

We apply collective classification to ConVote debates by letting $\mathcal{V}$ refer to the individual speakers in a debate and populating $\mathcal{N}$ using the citation graph between speakers. We set $\mathcal{L} = \{y, n\}$, corresponding to "for" and "against" votes respectively. The text of each instance is the concatenation of the speeches by a speaker within a debate. This results in a corpus of 1,699 instances with a roughly even class distribution. Approximately 70% of these are *connected*, i.e. they are the source or target of one or more citations. The remainder are *isolated*.

3 Collective Classification Techniques

In this section we describe techniques for performing collective classification on the ConVote corpus. We differentiate between *dual-classifier* and *iterative-classifier* approaches.

Dual-classifier approach: This approach uses a collective classification algorithm that takes inputs from two classifiers: (1) a *content-only* classifier that determines the likelihood of a y or n label for an instance given its text content; and (2) a *citation* classifier that determines, based on citation information, whether a given pair of instances are "same class" or "different class".

Let Ψ denote a set of functions representing the classification preferences produced by the content-only and citation classifiers:

- For each $V_i \in \mathcal{V}$, $\phi_i \in \Psi$ is a function ϕ_i: $\mathcal{L} \rightarrow \mathbb{R}^+ \cup \{0\}$.
- For each $(V_i, V_j) \in E$, $\psi_{ij} \in \Psi$ is a function ψ_{ij}: $\mathcal{L} \times \mathcal{L} \rightarrow \mathbb{R}^+ \cup \{0\}$.

Later in this section we will describe three collective classification algorithms capable of performing overall classification based on these inputs: (1) the minimum-cut approach, which is the benchmark for collective classification with ConVote, established by Thomas et al.; (2) loopy belief propagation; and (3) mean-field. We will show that these latter two techniques, which are both approximate solutions for Markov random fields, are superior to minimum-cut for the task.

Figure 2 gives a visual overview of the dual-classifier approach.

Iterative-classifier approach: This approach incorporates content-only and citation features into a single *local* classifier that works on the assumption that correct neighbor labels are already known. This approach represents a marked deviation from the dual-classifier approach and offers unique advantages. It is fully described in Section 3.4.

Figure 3 gives a visual overview of the iterative-classifier approach.

For a detailed introduction to collective classification see Sen et al. (2008).

Debate 006

Speaker 400378 [against]

Mr. Speaker, . . . all over Washington and in the country, people are talking today about the majority's last-minute decision to abandon . . .

. . .

Speaker 400115 [for]

. . .

Mr. Speaker, . . . I just want to say to the ***gentlewoman from New York*** *that every single member of this institution . . .*

. . .

Figure 1: Sample speech fragments from the ConVote corpus. The phrase *gentlewoman from New York* by speaker 400115 is annotated as a reference to speaker 400378.

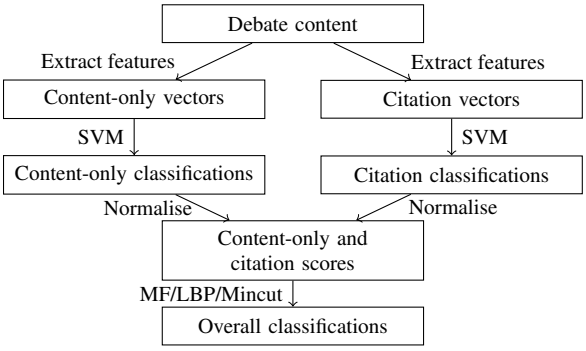


Figure 2: Dual-classifier approach.

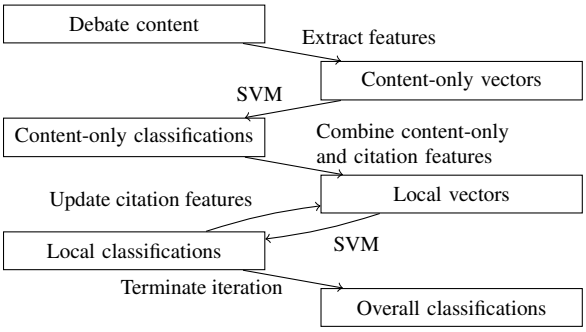


Figure 3: Iterative-classifier approach.

3.1 Dual-classifier Approach with Minimum-cut

Thomas et al. use linear kernel SVMs as their base classifiers. The content-only classifier is trained to predict y or n based on the unigram presence features found in speeches. The citation classifier is trained to predict "same class" or "different class" labels based on the unigram presence features found in the context windows (30 tokens before, 20 tokens after) surrounding citations for each pair of speakers in the debate.

The decision plane distance computed by the content-only SVM is normalized to a positive real number and stripped of outliers:

$$\phi_i(y) = \begin{cases} 1 & d_i > 2\sigma_i; \\ \left(1 + \frac{d_i}{2\sigma_i}\right)/2 & |d_i| \leq 2\sigma_i; \\ 0 & d_i < -2\sigma_i \end{cases}$$

where σ_i is the standard deviation of the decision plane distance, d_i, over all of the instances in the debate and $\phi_i(n) = 1 - \phi_i(y)$. The citation classifier output is processed similarly:[1]

$$\psi_{ij}(y, y) = \begin{cases} 0 & d_{ij} < \theta; \\ \alpha \cdot d_{ij}/4\sigma_{ij} & \theta \leq d_{ij} \leq 4\sigma_{ij}; \\ \alpha & d_{ij} > 4\sigma_{ij} \end{cases}$$

where σ_{ij} is the standard deviation of the decision plane distance, d_{ij} over all of the citations in the debate and $\psi_{ij}(n, n) = \psi_{ij}(y, y)$. The α and θ variables are free parameters.

A given class assignment v is assigned a cost that is the sum of per-instance and per-pair class costs derived from the content-only and citation classifiers respectively:

$$c(v) = \sum_{V_i \in \mathcal{V}} \phi_i(\bar{v}_i) + \sum_{(V_i, V_j) \in E : v_i \neq v_j} \psi_{ij}(v_i, v_i)$$

where v_i is the label of node V_i and $\bar{v}_i$ denotes the complement class of v_i.

[1] Thomas et al. classify each citation context window separately, so their ψ values are actually calculated in a slightly more complicated way. We adopted the present approach for conceptual simplicity and because it gave superior performance in preliminary experiments.

The cost function is modeled in a flow graph where extra source and sink nodes represent the y and n labels respectively. Each node in $\mathcal{V}$ is connected to the source and sink with capacities $\phi_i(y)$ and $\phi_i(n)$ respectively. Pairs classified in the "same class" class are linked with capacities defined by ψ.

An exact optimum and corresponding overall classification is efficiently computed by finding the minimum-cut of the flow graph (Blum and Chawla, 2001). The free parameters are tuned on a set of held-out data.

Thomas et al. demonstrate improvements over content-only classification, without attempting to show that the approach does better than any alternatives; the main appeal is the simplicity of the flow graph model. There are a number of theoretical limitations to the approach, which we now discuss.

As Thomas et al. point out, the model has no way of representing the "different class" output from the citation classifier and these citations must be discarded. This, to us, is the most significant problem with the model. Inspection of the corpus shows that approximately 80% of citations indicate agreement, meaning that for the present task the impact of discarding this information may not be large. However, the primary utility in collective approaches lies in their ability to fill in gaps in information not picked up by content-only classification. All available link information should be applied to this end, so we need models capable of accepting both positive and negative information.

The normalization techniques used for converting SVM outputs to graph weights are somewhat arbitrary. The use of standard deviations appears problematic as, intuitively, the strength of a classification should be independent of its variance. As a case in point, consider a set of instances in a debate all classified as similarly weak positives by the SVM. Use of ψ_i as defined above would lead to these being erroneously assigned the maximum score because of their low variance.

The minimum-cut approach places instances in either the positive or negative class depending on which side of the cut they fall on. This means that no measure of classification *confidence* is available. This extra information is useful at the very least to give a human user an idea of how much to trust the classification. A measure of classification confidence may also be necessary for incorporation into a broader system, e.g., a meta-classifier (Andreevskaia and Bergler, 2008; Li and Zong, 2008).

Tuning the α and θ parameters is likely to become a source of inaccuracy in cases where the tuning and test debates have dissimilar link structures. For example, if the tuning debates tend to have fewer, more accurate links the α parameter will be higher. This will not produce good results if the test debates have more frequent, less accurate links.

3.2 Heuristics for Improving Minimum-cut

Bansal et al. (2008) offer preliminary work describing additions to the Thomas et al. minimum-cut approach to incorporate "different class" citation classifications. They use post hoc adjustments of graph capacities based on simple heuristics. Two of the three approaches they trial appear to offer performance improvements:

The *SetTo* heuristic: This heuristic works through E in order and tries to force V_i and V_j into different classes for every "different class" ($d_{ij} < 0$) citation classifier output where $i < j$. It does this by altering the four relevant content-only preferences, $\phi_i(y)$, $\phi_i(n)$, $\phi_j(y)$, and $\phi_j(n)$. Assume without loss of generality that the largest of these values is $\phi_i(y)$. If this preference is respected, it follows that V_j should be put into class n. Bansal et al. instantiate this chain of reasoning by setting:

- $\phi'_i(y) = \max(\beta, \phi_i(y))$
- $\phi'_j(n) = \max(\beta, \phi_j(n))$

where ϕ' is the replacement content-only function, β is a free parameter $\in (.5, 1]$, $\phi'_i(n) = 1 - \phi'_i(y)$, and $\phi'_j(y) = 1 - \phi'_j(y)$.

The *IncBy* heuristic: This heuristic is a more conservative version of the *SetTo* heuristic. Instead of replacing the content-only preferences with fixed constants, it increments and decrements the previous values so they are somewhat preserved:

- $\phi'_i(y) = \min(1, \phi_i(y) + \beta)$
- $\phi'_j(n) = \min(1, \phi_j(n) + \beta)$

There are theoretical shortcomings with these approaches. The most obvious problem is the arbitrary nature of the manipulations, which produce a flow

graph that has an indistinct relationship to the outputs of the two classifiers.

Bensal et al. trial a range of β values, with varying impacts on performance. No attempt is made to demonstrate a method for choosing a good β value. It is not clear that the tuning approach used to set α and θ would be successful here. In any case, having a third parameter to tune would make the process more time-consuming and increase the risks of incorrect tuning, described above.

As Bansal et al. point out, proceeding through E in order means that earlier changes may be undone for speakers who have multiple "different class" citations.

Finally, we note that the confidence of the citation classifier is not embodied in the graph structure. The most marginal "different class" citation, classified just on the negative side of the decision plane, is treated identically to the most confident one furthest from the decision plane.

3.3 Dual-classifier Approach with Markov Random Field Approximations

A pairwise Markov random field (Taskar et al., 2002) is given by the pair (G, Ψ), where G and Ψ are as previously defined, Ψ being re-termed as a set of *clique potentials*. Given an assignment v to the nodes $\mathcal{V}$, the pairwise Markov random field is associated with the probability distribution:

$$P(v) = \frac{1}{\mathcal{Z}} \prod_{V_i \in \mathcal{V}} \phi_i(v_i) \prod_{(V_i,V_j) \in E} \psi_{ij}(v_i, v_j)$$

where:

$$\mathcal{Z} = \sum_{v'} \prod_{V_i \in \mathcal{V}} \phi_i(v'_i) \prod_{(V_i,V_j) \in E} \psi_{ij}(v'_i, v'_j)$$

and v'_i denotes the label of V_i for an alternative assignment in v'.

In general, exact inference over a pairwise Markov random field is known to be NP-hard. There are certain conditions under which exact inference is tractable, but real-world data is not guaranteed to satisfy these. A class of approximate inference algorithms known as *variational methods* (Jordan et al., 1999) solve this problem by substituting a simpler "trial" distribution which is fitted to the Markov random field distribution.

Loopy Belief Propagation: Applied to a pairwise Markov random field, loopy belief propagation is a message passing algorithm that can be concisely expressed as the following set of equations:

$$\begin{aligned} m_{i \to j}(v_j) &= \alpha \sum_{v_i \in \mathcal{L}} \{\psi_{ij}(v_i, v_j)\phi_i(v_i) \\ &\qquad \prod_{V_k \in \mathcal{N}_i \cap \mathcal{V} \setminus V_j} m_{k \to i}(v_i), \forall v_j \in \mathcal{L}\} \\ b_i(v_i) &= \alpha \phi_i(v_i) \prod_{V_j \in \mathcal{N}_i \cap \mathcal{V}} m_{j \to i}(v_i), \forall v_i \in \mathcal{L} \end{aligned}$$

where $m_{i \to j}$ is a message sent by V_i to V_j and α is a normalization constant that ensures that each message and each set of marginal probabilities sum to 1. The algorithm proceeds by making each node communicate with its neighbors until the messages stabilize. The marginal probability is then derived by calculating $b_i(v_i)$.

Mean-Field: The basic mean-field algorithm can be described with the equation:

$$b_j(v_j) = \alpha \phi_j(v_j) \prod_{V_i \in \mathcal{N}_j \cap \mathcal{V}} \prod_{v_i \in \mathcal{L}} \psi_{ij}^{b_i(v_i)}(v_i, v_j), v_j \in \mathcal{L}$$

where α is a normalization constant that ensures $\sum_{v_j} b_j(v_j) = 1$. The algorithm computes the fixed point equation for every node and continues to do so until the marginal probabilities $b_j(v_j)$ stabilize.

Mean-field can be shown to be a variational method in the same way as loopy belief propagation, using a simpler trial distribution. For details see Sen et al. (2008).

Probabilistic SVM Normalisation: Unlike minimum-cut, the Markov random field approaches have inherent support for the "different class" output of the citation classifier. This allows us to apply a more principled SVM normalisation technique. Platt (1999) describes a technique for converting the output of an SVM classifier to a calibrated posterior probability. Platt finds that the posterior can be fit using a parametric form of a sigmoid:

$$P(y = 1|d) = \frac{1}{1 + \exp(Ad + B)}$$

This is equivalent to assuming that the output of the SVM is proportional to the log odds of a positive example. Experimental analysis shows error rate is

improved over a plain linear SVM and probabilities are of comparable quality to those produced using a regularized likelihood kernel method.

By applying this technique to the base classifiers, we can produce new, simpler Ψ functions, $\phi_i(y) = P_i$ and $\psi_{ij}(y, y) = P_{ij}$ where P_i is the probabilistic normalized output of the content-only classifier and P_{ij} is the probabilistic normalized output of the citation classifier.

This approach addresses the problems with the Thomas et al. method where the use of standard deviations can produce skewed normalizations (see Section 3.1). By using probabilities we also open up the possibility of replacing the SVM classifiers with any other model than can be made to produce a probability. Note also that there are no parameters to tune.

3.4 Iterative Classifier Approach

The dual-classifier approaches described above represent *global* attempts to solve the collective classification problem. We can choose to narrow our focus to the *local* level, in which we aim to produce the best classification for a single instance with the assumption that all other parts of the problem (i.e. the correct labeling of the other instances) are solved.

The Iterative Classification Algorithm (Bilgic et al., 2007), defined in Algorithm 1, is a simple technique for performing collective classification using such a local classifier. After bootstrapping with a content-only classifier, it repeatedly generates new estimates for v_i based on its current knowledge of $\mathcal{N}_i$. The algorithm terminates when the predictions stabilize or a fixed number of iterations is completed. Each iteration is completed using a newly generated ordering $\mathcal{O}$, over the instances $\mathcal{V}$.

We propose three feature models for the local classifier.

Citation presence and Citation count: Given that the majority of citations represent the "same class" relationship (see Section 3.1), we can anticipate that content-only classification performance will be improved if we add features to represent the presence of neighbours of each class.

We define the function $c(i, l) = \sum_{v_j \in \mathcal{N}_i \cap \mathcal{V}} \delta_{v_j, l}$ giving the number of neighbors for node V_i with label l, where δ is the Kronecker delta. We incorporate these *citation count* values, one for the supporting class and one for the opposing class, obtaining a new feature vector $(u_i^1, u_i^2, \ldots, u_i^j, c(i, y), c(i, n))$ where $u_i^1, u_i^2, \ldots, u_i^j$ are the elements of $\vec{u}_i$, the binary unigram feature vector used by the content-only classifier to represent instance i.

Algorithm 1 Iterative Classification Algorithm

```
for each node V_i ∈ V do {bootstrapping}
    compute a_i using only local attributes of node
    v_i ← f(a_i)
end for
repeat {iterative classification}
    randomly generate ordering O over nodes in V
    for each node V_i ∈ O do
        {compute new estimate of v_i}
        compute a_i using current assignments to N_i
        v_i ← f(a_i)
    end for
until labels have stabilized or maximum iterations
reached
```

Alternatively, we can represent neighbor labels using binary *citation presence* values where any non-zero count becomes a 1 in the feature vector.

Context window: We can adopt a more nuanced model for citation information if we incorporate the citation context window features into the feature vector. This is, in effect, a synthesis of the content-only and citation feature models. Context window features come from the product space $\mathcal{L} \times \mathcal{C}$, where $\mathcal{C}$ is the set of unigrams used in citation context windows and $\vec{c}_i$ denotes the context window features for instance i. The new feature vector becomes: $(u_i^1, u_i^2, \ldots, u_i^j, c_i^1, c_i^2, \ldots, c_i^k)$. This approach implements the intuition that speakers indicate their voting intentions by the words they use to refer to speakers whose vote is known. Because neighbor relations are bi-directional the reverse is also true: Speakers indicate other speakers' voting intentions by the words they use to refer to them.

As an example, consider the context window feature `AGREE-FOR`, indicating the presence of the *agree* unigram in the citation window *I agree with the gentleman from Louisiana*, where the label for the *gentleman from Louisiana* instance is y. This feature will be correctly correlated with the y label. Similarly, if the unigram were *disagree* the feature would be correlated with the n label.

4 Experiments

In this section we compare the performance of our dual-classifier and iterative-classifier approaches. We also evaluate the performance of the three feature models for local classification.

All accuracies are given as the percentages of instances correctly classified. Results are macro-averaged using 10 $\times$ 10-fold cross validation, i.e. 10 runs of 10-fold cross validation using different randomly assigned data splits.

Where quoted, statistical significance has been calculated using a two-tailed paired t-test measured over all 100 pairs with 10 degrees of freedom. See Bouckaert (2003) for an experimental justification for this approach.

Note that the results presented in this section are not directly comparable with those reported by Thomas et al. and Bansal et al. because their experiments do not use cross-validation. See Section 4.3 for further discussion of experimental configuration.

4.1 Local Classification

We evaluate three models for local classification: citation presence features, citation count features and context window features. In each case the SVM classifier is given feature vectors with both content-only and citation information, as described in Section 3.4.

Table 1 shows that context window performs the best with 89.66% accuracy, approximately 1.5% ahead of citation count and 3.5% ahead of citation presence. All three classifiers significantly improve on the content-only classifier.

These relative scores seem reasonable. Knowing the words used in citations of each class is better than knowing the number of citations in each class, and better still than only knowing which classes of citations exist.

These results represent an upper-bound for the performance of the iterative classifier, which relies on iteration to produce the reliable information about citations given here by oracle.

4.2 Collective Classification

Table 2 shows overall results for the three collective classification algorithms. The iterative classifier was run separately with citation count and context window citation features, the two best performing local classification methods, both with a threshold of 30 iterations.

Method	Accuracy (%)
Majority	52.46
Content-only	75.29
Citation presence	85.01
Citation count	88.18
Context window	89.66

Table 1: Local classifier accuracy. All three local classifiers are significant over the in-isolation classifier ($p < .001$).

Results are shown for connected instances, isolated instances, and all instances. Collective classification techniques can only have an impact on connected instances, so these figures are most important. The figures for all instances show the performance of the classifiers in our real-world task, where both connected and isolated instances need to be classified and the end-user may not distinguish between the two types.

Each of the four collective classifiers outperform the minimum-cut benchmark over connected instances, with the iterative classifier (context window) (79.05%) producing the smallest gain of less than 1% and mean-field doing best with a nearly 6% gain (84.13%). All show a statistically significant improvement over the content-only classifier. Mean-field shows a statistically significant improvement over minimum-cut.

The dual-classifier approaches based on loopy belief propagation and mean-field do better than the iterative-classifier approaches by an average of about 3%.

Iterative classification performs slightly better with citation count features than with context window features, despite the fact that the context window model performs better in the local classifier evaluation. We speculate that this may be due to citation count performing better when given incorrect neighbor labels. This is an aspect of local classifier performance we do not otherwise measure, so a clear conclusion is not possible. Given the closeness of the results it is also possible that natural statistical variation is the cause of the difference.

The performance of the minimum-cut method is not reliably enhanced by either the *SetTo* or *IncBy* heuristics. Only *IncBy*(.15) gives a very small improvement (0.14%) over plain minimum-cut. All of the other combinations tried diminished performance slightly.

4.3 A Note on Error Propagation and Experimental Configuration

Early in our experimental work we noticed that performance often varied greatly depending on the debates that were allocated to training, tuning and testing. This observation is supported by the per-fold scores that are the basis for the macro-average performance figures reported in Table 2, which tend to have large standard deviations. The absolute standard deviations over the 100 evaluations for the minimum-cut and mean-field methods were 11.19% and 8.94% respectively. These were significantly larger than the standard deviation for the content-only baseline, which was 7.34%. This leads us to conclude that the performance of collective classification methods is highly variable.

Bilgic and Getoor (2008) offer a possible explanation for this. They note that the cost of incorrectly classifying a given instance can be magnified in collective classification, because errors are propagated throughout the network. The extent to which this happens may depend on the random interaction between base classification accuracy and network structure. There is scope for further work to more fully explain this phenomenon.

From these statistical and theoretical factors we infer that more reliable conclusions can be drawn from collective classification experiments that use cross-validation instead of a single, fixed data split.

5 Related work

Somasundaran et al. (2009) use ICA to improve sentiment polarity classification of dialogue acts in a corpus of multi-party meeting transcripts. Link features are derived from annotations giving *frame* relations and *target* relations. Respectively, these relate dialogue acts based on the sentiment expressed and the object towards which the sentiment is expressed. Somasundaran et al. provides another argument for the usefulness of collective classification (specifically ICA), in this case as applied at a dialogue act level and relying on a complex system of annotations for link information.

Somasundaran and Wiebe (2009) propose an unsupervised method for classifying the stance of each contribution to an online debate concerning the merits of competing products. Concessions to other stances are modeled, but there are no overt citations in the data that could be used to induce the network structure required for collective classification.

Pang and Lee (2005) use metric labeling to perform multi-class collective classification of movie reviews. Metric labeling is a multi-class equivalent of the minimum-cut technique in which optimization is done over a cost function incorporating content-only and citation scores. Links are constructed between test instances and a set of k nearest neighbors drawn only from the training set. Restricting the links in this way means the optimization problem is simple. A similarity metric is used to find nearest neighbors.

The Pang and Lee method is an instance of implicit link construction, an approach which is beyond the scope of this paper but nevertheless an important area for future research. A similar technique is used in a variation on the Thomas et al. experiment where additional links between speeches are inferred via a similarity metric (Burfoot, 2008). In cases where both citation and similarity links are present, the overall link score is taken as the sum of the two scores. This seems counter-intuitive, given that the two links are unlikely to be independent. In the framework of this research, the approach would be to train a link meta-classifier to take scores from both link classifiers and output an overall link probability.

Within NLP, the use of LBP has not been restricted to document classification. Examples of other applications are dependency parsing (Smith and Eisner, 2008) and alignment (Cromires and Kurohashi, 2009). Conditional random fields (CRFs) are an approach based on Markov random fields that have been popular for segmenting and labeling sequence data (Lafferty et al., 2001). We rejected linear-chain CRFs as a candidate approach for our evaluation on the grounds that the arbitrarily connected graphs used in collective classification can not be fully represented in graphical format, i.e.

	Connected	Isolated	All
Majority	52.46	46.29	50.51
Content only	75.31	78.90	76.28
Minimum-cut	78.31	78.90	78.40
Minimum-cut (*SetTo*(.6))	78.22	78.90	78.32
Minimum-cut (*SetTo*(.8))	78.01	78.90	78.14
Minimum-cut (*SetTo*(1))	77.71	78.90	77.93
Minimum-cut (*IncBy*(.05))	78.14	78.90	78.25
Minimum-cut (*IncBy*(.15))	78.45	78.90	78.46
Minimum-cut (*IncBy*(.25))	78.02	78.90	78.15
Iterative-classifier (citation count)	80.07⋆	78.90	79.69⋆
Iterative-classifier (context window)	79.05	78.90	78.93
Loopy Belief Propagation	83.37†	78.90	81.93†
Mean-Field	84.12†	78.90	82.45†

Table 2: Speaker classification accuracies (%) over connected, isolated and all instances. The marked results are statistically significant over the content only benchmark ($\star\ p < .01$, $\dagger\ p < .001$). The mean-field results are statistically significant over minimum-cut ($p < .05$).

linear-chain CRFs do not scale to the complexity of graphs used in this research.

6 Conclusions and future work

By applying alternative models, we have demonstrated the best recorded performance for collective classification of ConVote using bag-of-words features, beating the previous benchmark by nearly 6%. Moreover, each of the three alternative approaches trialed are theoretically superior to the minimum-cut approach approach for three main reasons: (1) they support multi-class classification; (2) they support negative and positive citations; (3) they require no parameter tuning.

The superior performance of the dual-classifier approach with loopy belief propagation and mean-field suggests that either algorithm could be considered as a first choice for collective document classification. Their advantage is increased by their ability to output classification confidences as probabilities, while minimum-cut and the local formulations only give absolute class assignments. We do not dismiss the iterative-classifier approach entirely. The most compelling point in its favor is its ability to unify content only and citation features in a single classifier. Conceptually speaking, such an approach should allow the two types of features to inter-relate in more nuanced ways. A case in point comes from our use of a fixed size context window to build a citation classifier. Future approaches may be able to do away with this arbitrary separation of features by training a local classifier to consider all words in terms of their impact on content-only classification and their relations to neighbors.

Probabilistic SVM normalization offers a convenient, principled way of incorporating the outputs of an SVM classifier into a collective classifier. An opportunity for future work is to consider normalization approaches for other classifiers. For example, confidence-weighted linear classifiers (Dredze et al., 2008) have been shown to give superior performance to SVMs on a range of tasks and may therefore be a better choice for collective document classification.

Of the three models trialled for local classifiers, context window features did best when measured in an oracle experiment, but citation count features did better when used in a collective classifier. We conclude that context window features are a more nuanced and powerful approach that is also more likely to suffer from data sparseness. Citation count features would have been the less effective in a scenario where the fact of the citation existing was less informative, for example, if a citation was 50% likely to indicate agreement rather than 80% likely. There is much scope for further research in this area.

References

Alina Andreevskaia and Sabine Bergler. 2008. When specialists and generalists work together: Overcoming domain dependence in sentiment tagging. In *ACL*, pages 290–298.

Mohit Bansal, Claire Cardie, and Lillian Lee. 2008. The power of negative thinking: Exploiting label disagreement in the min-cut classification framework. In *COLING*, pages 15–18.

Mustafa Bilgic and Lise Getoor. 2008. Effective label acquisition for collective classification. In *KDD*, pages 43–51.

Mustafa Bilgic, Galileo Namata, and Lise Getoor. 2007. Combining collective classification and link prediction. In *ICDM Workshops*, pages 381–386. IEEE Computer Society.

Avrim Blum and Shuchi Chawla. 2001. Learning from labeled and unlabeled data using graph mincuts. In *ICML*, pages 19–26.

Remco R. Bouckaert. 2003. Choosing between two learning algorithms based on calibrated tests. In *ICML*, pages 51–58.

Clint Burfoot and Timothy Baldwin. 2009. Automatic satire detection: Are you having a laugh? In *ACL-IJCNLP Short Papers*, pages 161–164.

Clint Burfoot. 2008. Using multiple sources of agreement information for sentiment classification of political transcripts. In *Australasian Language Technology Association Workshop 2008*, pages 11–18. ALTA.

Minh Duc Cao and Xiaoying Gao. 2005. Combining contents and citations for scientific document classification. In *18th Australian Joint Conference on Artificial Intelligence*, pages 143–152.

Fabien Cromires and Sadao Kurohashi. 2009. An alignment algorithm using belief propagation and a structure-based distortion model. In *EACL*, pages 166–174.

Mark Dredze, Koby Crammer, and Fernando Pereira. 2008. Confidence-weighted linear classification. In *ICML*, pages 264–271.

Zeno Gantner and Lars Schmidt-Thieme. 2009. Automatic content-based categorization of Wikipedia articles. In *2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, pages 32–37.

Michael Jordan, Zoubin Ghahramani, Tommi Jaakkola, Lawrence Saul, and David Heckerman. 1999. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

Shoushan Li and Chengqing Zong. 2008. Multi-domain sentiment classification. In *ACL*, pages 257–260.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *EMNLP*, pages 79–86.

John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press.

Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI Magazine*, 29:93–106.

David A. Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *EMNLP*, pages 145–156.

Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *ACL-IJCNLP*, pages 226–234.

Swapna Somasundaran, Galileo Namata, Janyce Wiebe, and Lise Getoor. 2009. Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In *EMNLP*, pages 170–179.

Ben Taskar, Pieter Abbeel, and Daphne Koller. 2002. Discriminative probabilistic models for relational data. In *UAI*.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *EMNLP*, pages 327–335.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *Proceedings ACM SIGIR*, pages 42–49.

Integrating history-length interpolation and classes in language modeling

Hinrich Schütze
Institute for NLP
University of Stuttgart
Germany

Abstract

Building on earlier work that integrates different factors in language modeling, we view (i) backing off to a shorter history and (ii) class-based generalization as two complementary mechanisms of using a larger equivalence class for prediction when the default equivalence class is too small for reliable estimation. This view entails that the classes in a language model should be learned from rare events only and should be preferably applied to rare events. We construct such a model and show that both training on rare events and preferable application to rare events improve perplexity when compared to a simple direct interpolation of class-based with standard language models.

1 Introduction

Language models, probability distributions over strings of words, are fundamental to many applications in natural language processing. The main challenge in language modeling is to estimate string probabilities accurately given that even very large training corpora cannot overcome the inherent sparseness of word sequence data. One way to improve the accuracy of estimation is *class-based generalization*. The idea is that even though a particular word sequence s may not have occurred in the training set (or too infrequently for accurate estimation), the occurrence of sequences similar to s can help us better estimate $p(s)$.

Plausible though this line of reasoning is, the language models most commonly used today do not incorporate class-based generalization. This is partially due to the additional cost of creating classes and using classes as part of the model. But an equally important reason is that most models that integrate class-based information do so by way of a simple interpolation and achieve only a modest improvement in performance.

In this paper, we propose a new type of class-based language model. The key novelty is that we recognize that certain probability estimates are hard to improve based on classes. In particular, the best probability estimate for frequent events is often the maximum likelihood estimator and this estimator is hard to improve by using other information sources like classes or word similarity. We therefore design a model that attempts to focus the effect of class-based generalization on rare events.

Specifically, we propose to employ the same strategy for this that history-length interpolated (HI) models use. We define HI models as models that interpolate the predictions of different-length histories, e.g., $p(w_3|w_1w_2) = \lambda_1(w_1w_2)p'(w_3|w_1w_2) + \lambda_2(w_1w_2)p'(w_3|w_2) + (1 - \lambda_1(w_1w_2) - \lambda_2(w_1w_2))p'(w_3)$ where p' is a simple estimate; in this section, we use $p' = p_{\text{ML}}$, the maximum likelihood estimate, as an example. Jelinek-Mercer (Jelinek and Mercer, 1980) and modified Kneser-Ney (Kneser and Ney, 1995) models are examples of HI models.

HI models address the challenge that frequent events are best estimated by a method close to maximum likelihood by selecting appropriate values for the interpolation weights. For example, if $w_1w_2w_3$ is frequent, then λ_1 will be close to 1, thus ensuring that $p(w_3|w_1w_2) \approx p_{\text{ML}}(w_3|w_1w_2)$ and that the components $p_{\text{ML}}(w_3|w_2)$ and $p_{\text{ML}}(w_3)$, which are unhelpful in this case, will only slightly change the reliable estimate $p_{\text{ML}}(w_3|w_1w_2)$.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1516–1525,
Portland, Oregon, June 19-24, 2011.

The main contribution of this paper is to propose the same mechanism for class language models. In fact, we will use the interpolation weights of a KN model to determine how much weight to give to each component of the interpolation. The difference to a KN model is merely that the lower-order distribution is not the lower-order KN distribution (as in KN), but instead an interpolation of the lower-order KN distribution and a class-based distribution. We will show that this method of integrating history interpolation and classes significantly increases the performance of a language model.

Focusing the effect of classes on rare events has another important consequence: if this is the right way of using classes, then they should not be formed based on *all events* in the training set, but only based on *rare events*. We show that doing this increases performance.

Finally, we introduce a second discounting method into the model that differs from KN. This can be motivated by the fact that with two sources of generalization (history-length and classes) more probability mass should be allocated to these two sources than to the single source used in KN. We propose a *polynomial discount* and show a significant improvement compared to using KN discounting only.

This paper is structured as follows. Section 2 discusses related work. Section 3 reviews the KN model and introduces two models, the Dupont-Rosenfeld model (a "recursive" model) and a top-level interpolated model, that integrate the KN model (a history interpolation model) with a class model. Section 4 details our experimental setup. Results are presented in Section 5. Based on an analysis of strengths and weaknesses of Dupont-Rosenfeld and top-level interpolated models, we present a new polynomial discounting mechanism that does better than either in Section 6. Section 7 presents our conclusions.

2 Related work

A large number of different class-based models have been proposed in the literature. The well-known model by Brown et al. (1992) is a class sequence model, in which $p(u|w)$ is computed as the product of a class transition probability and an emission probability, $p(g(u)|g(w))p(u|g(u))$, where $g(u)$ is the class of u. Other approaches condition the probability of a class on n-grams of lexical items (as opposed to classes) (Whittaker and Woodland, 2001; Emami and Jelinek, 2005; Uszkoreit and Brants, 2008). In this work, we use the Brown type of model: it is simpler and has fewer parameters. Models that condition classes on lexical n-grams could be extended in a way similar to what we propose here.

Classes have been used with good results in a number of applications, e.g., in speech recognition (Yokoyama et al., 2003), sentiment analysis (Wiegand and Klakow, 2008), and question answering (Momtazi and Klakow, 2009). Classes have also been shown to improve the performance of exponential models (Chen, 2009).

Our use of classes of lexical n-grams for $n > 1$ has several precedents in the literature (Suhm and Waibel, 1994; Kuo and Reichl, 1999; Deligne and Sagisaka, 2000; Justo and Torres, 2009). The novelty of our approach is that we integrate phrase-level classes into a KN model.

Hierarchical clustering (McMahon and Smith, 1996; Zitouni and Zhou, 2007; Zitouni and Zhou, 2008) has the advantage that the size of the class to be used in a specific context is not fixed, but can be chosen at an optimal level of the hierarchy. There is no reason why our non-hierarchical flat model could not be replaced with a hierarchical model and we would expect this to improve results.

The key novelty of our clustering method is that clusters are formed based on rare events in the training corpus. This type of clustering has been applied to other problems before, in particular to unsupervised part-of-speech tagging (Schütze, 1995; Clark, 2003; Reichart et al., 2010). However, the importance of rare events for clustering in language modeling has not been investigated before.

Our work is most similar to the lattice-based language models proposed by Dupont and Rosenfeld (1997). Bilmes and Kirchhoff (2003) generalize lattice-based language models further by allowing arbitrary factors in addition to words and classes. We use a special case of lattice-based language models in this paper. Our contributions are that we introduce the novel idea of rare-event clustering into language modeling and that we show that the modified model performs better than a strong word-trigram

symbol	denotation
$\sum[[w]]$	$\sum_w$ (sum over all unigrams w)
$c(w_j^i)$	count of w_j^i
$n_{1+}(\bullet w_j^i)$	# of distinct w occurring before w_j^i

Table 1: Notation used for Kneser-Ney.

baseline.

3 Models

In this section, we introduce the three models that we compare in our experiments: Kneser-Ney model, Dupont-Rosenfeld model, and top-level interpolation model.

3.1 Kneser-Ney model

Our baseline model is the modified Kneser-Ney (KN) trigram model as proposed by Chen and Goodman (1999). We give a comprehensive description of our implementation of KN because the details are important for the integration of the class model given below. We use the notation in Table 1.

We estimate p_{KN} on the training set as follows.

$$\begin{aligned} p_{\text{KN}}(w_3|w_1^2) &= \frac{c(w_1^3) - d'''(c(w_1^3))}{\sum[[w]]\ c(w_1^2 w)} \\ &\quad +\gamma_3(w_1^2) p_{\text{KN}}(w_3|w_2) \end{aligned}$$

$$\gamma_3(w_1^2) = \frac{\sum[[w]]\ d'''(c(w_1^2 w))}{\sum[[w]]\ c(w_1^2 w)}$$

$$\begin{aligned} p_{\text{KN}}(w_3|w_2) &= \frac{n_{1+}(\bullet w_2^3) - d''(n_{1+}(\bullet w_2^3))}{\sum[[w]]\ n_{1+}(\bullet w_2 w)} \\ &\quad +\gamma_2(w_2) p_{\text{KN}}(w_3) \end{aligned}$$

$$\gamma_2(w_2) = \frac{\sum[[w]]\ d''(n_{1+}(\bullet w_2 w))}{\sum[[w]]\ n_{1+}(\bullet w_2 w)}$$

$$p_{\text{KN}}(w_3) = \begin{cases} \frac{n_{1+}(\bullet w_3) - d'(n_{1+}(\bullet w_3))}{\sum[[w]]\ n_{1+}(\bullet w)} & \text{if } c(w_3) > 0 \\ \gamma_1 & \text{if } c(w_3) = 0 \end{cases}$$

$$\gamma_1 = \frac{\sum[[w]]\ d'(n_{1+}(\bullet w))}{\sum[[w]]\ n_{1+}(\bullet w)}$$

The parameters d', d'', and d''' are the discounts for unigrams, bigrams and trigrams, respectively, as defined by Chen and Goodman (1996, p. 20, (26)). Note that our notation deviates from C&G in that they use the single symbol D_1 for the three different values $d'(1)$, $d''(1)$, and $d'''(1)$ etc.

3.2 Dupont-Rosenfeld model

History-interpolated models attempt to find a good tradeoff between using a maximally informative history for accurate prediction of frequent events and generalization for rare events by using lower-order distributions; they employ this mechanism recursively by progressively shortening the history.

The key idea of the improved model we will adopt is that class generalization ought to play the same role in history-interpolated models as the lower-order distributions: they should improve estimates for unseen and rare events. Following Dupont and Rosenfeld (1997), we implement this idea by linearly interpolating the class-based distribution with the lower order distribution, recursively at each level. For a trigram model, this means that we interpolate $p_{\text{KN}}(w_3|w_2)$ and $p_{\text{B}}(w_3|w_1 w_2)$ on the first backoff level and $p_{\text{KN}}(w_3)$ and $p_{\text{B}}(w_3|w_2)$ on the second backoff level, where p_{B} is the (Brown) class model (see Section 4 for details on p_{B}). We call this model p_{DR} for Dupont-Rosenfeld model and define it as follows:

$$\begin{aligned} p_{\text{DR}}(w_3|w_1^2) = & \frac{c(w_1^3) - d'''(c(w_1^3))}{\sum[[w]]\ c(w_1^2 w)} \\ + & \gamma_3(w_1^2)[\beta_1(w_1^2) p_{\text{B}}(w_3|w_1^2) \\ & +(1 - \beta_1(w_1^2)) p_{\text{DR}}(w_3|w_2)] \end{aligned}$$

$$\begin{aligned} p_{\text{DR}}(w_3|w_2) = & \frac{n_{1+}(\bullet w_2^3) - d''(n_{1+}(\bullet w_2^3))}{\sum[[w]]\ n_{1+}(\bullet w_2 w)} \\ + & \gamma_2(w_2)[\beta_2(w_2) p_{\text{B}}(w_3|w_2) \\ & +(1 - \beta_2(w_2)) p_{\text{DR}}(w_3)] \end{aligned}$$

where $\beta_i(v)$ is equal to a parameter α_i if the history (w_1^2 or w_2) is part of a cluster and 0 otherwise:

$$\beta_i(v) = \begin{cases} \alpha_i & \text{if } v \in \mathcal{B}_{2-(i-1)} \\ 0 & \text{otherwise} \end{cases}$$

$\mathcal{B}_1$ (resp. $\mathcal{B}_2$) is the set of unigram (resp. bigram) histories that is covered by the clusters. We cluster bigram histories and unigram histories separately and write $p_{\text{B}}(w_3|w_1 w_2)$ for the bigram cluster model and $p_{\text{B}}(w_3|w_2)$ for the unigram cluster model. Clustering and the estimation of these two distributions are described in Section 4.

The unigram distribution of the Dupont-Rosenfeld model is set to the unigram distribution of the KN model: $p_{\text{DR}}(w) = p_{\text{KN}}(w)$.

The model (or family of models) defined by Dupont and Rosenfeld (1997) is more general than our version p_{DR}. Most importantly, it allows a truly parallel backoff whereas in our model the recursive backoff distribution p_{DR} is interpolated with a class distribution p_{B} that is not backed off. We prefer this version because it makes it easier to understand the contribution that unique-event vs. all-event classes make to improved language modeling; the parameters β are a good indicator of this effect.

An alternative way of setting up the Dupont-Rosenfeld model would be to interpolate $p_{\text{KN}}(w_3|w_1w_2)$ and $p_{\text{B}}(w_3|w_1w_2)$ etc – but this is undesirable. The strength of history interpolation is that estimates for frequent events are close to ML, e.g., $p_{\text{KN}}(\text{share}|\text{cents a}) \approx p_{\text{ML}}(\text{share}|\text{cents a})$ for our corpus. An ML estimate is accurate for large counts and we should not interpolate it directly with $p_{\text{B}}(w_3|w_1w_2)$. For p_{DR}, the discount d''' that is subtracted from $c(w_1w_2w_3)$ is small relative to $c(w_1w_2w_3)$ and therefore $p_{\text{DR}} \approx p_{\text{ML}}$ in this case (exactly as in p_{KN}).

3.3 Top-level interpolation

Class-based models are often combined with other models by interpolation, starting with the work by Brown et al. (1992). Since we cluster both unigrams and bigrams, we interpolate three models:

$$\begin{aligned} & p_{\text{TOP}}(w_3|w_1w_2) \\ = \; & \mu_1(w_1w_2)p_{\text{B}}(w_3|w_1w_2) + \mu_2(w_2)p_{\text{B}}(w_3|w_2) \\ + \; & (1 - \mu_1(w_1w_2) - \mu_2(w_2))p_{\text{KN}}(w_3|w_1w_2) \end{aligned}$$

where $\mu_1(w_1w_2) = \lambda_1$ if $w_1w_2 \in \mathcal{B}_2$ and 0 otherwise, $\mu_2(w_2) = \lambda_2$ if $w_2 \in \mathcal{B}_1$ and 0 otherwise and λ_1 and λ_2 are parameters. We call this the *top-level model* p_{TOP} because it interpolates the three models at the top level. Most previous work on class-based model has employed some form of top-level interpolation.

4 Experimental Setup

We run experiments on a Wall Street Journal (WSJ) corpus of 50M words, split 8:1:1 into training, validation and test sets. The training set contains 256,873 unique unigrams and 4,494,222 unique bigrams. Unknown words in validation and test sets are mapped to a special unknown word u.

We use the SRILM toolkit (Stolcke, 2002) for clustering. An important parameter of the class-based model is size $|\mathcal{B}_i|$ of the base set, i.e., the total number of n-grams (or rather i-grams) to be clustered. As part of the experiments we vary $|\mathcal{B}_i|$ systematically to investigate the effect of base set size. We cluster unigrams ($i = 1$) and bigrams ($i = 2$). For all experiments, $|\mathcal{B}_1| = |\mathcal{B}_2|$ (except in cases where $|\mathcal{B}_2|$ exceeds the number of unigrams, see below). SRILM does not directly support bigram clustering. We therefore represent a bigram as a hyphenated word in bigram clustering; e.g., *Pan Am* is represented as *Pan-Am*.

The input to the clustering is the vocabulary $\mathcal{B}_i$ and the cluster training corpus. For a particular base set size b, the unigram input vocabulary $\mathcal{B}_1$ is set to the b most frequent unigrams in the training set and the bigram input vocabulary $\mathcal{B}_2$ is set to the b most frequent bigrams in the training set.

In this section, we call the WSJ training corpus the *raw corpus* and the cluster training corpus the *cluster corpus* to be able to distinguish them. We run four different clusterings for each base set size (except for the large sets, see below). The cluster corpora are constructed as follows.

- **All-event unigram clustering.** The cluster corpus is simply the raw corpus.
- **All-event bigram clustering.** The cluster corpus is constructed as follows. A sentence of the raw corpus that contains s words is included twice, once as a sequence of the $\lfloor s/2 \rfloor$ bigrams "w_1–w_2 w_3–w_4 w_5–w_6 ..." and once as a sequence of the $\lfloor (s-1)/2 \rfloor$ bigrams "w_2–w_3 w_4–w_5 w_6–w_7 ...".
- **Unique-event unigram clustering.** The cluster corpus is the set of all sequences of two unigrams $\in \mathcal{B}_1$ that occur in the raw corpus, one sequence per line. Each sequence occurs only once in this cluster corpus.
- **Unique-event bigram clustering.** The cluster corpus is the set of all sequences of two bigrams $\in \mathcal{B}_2$ that occur in the training corpus,

one sequence per line. Each sequence occurs only once in this cluster corpus.

As mentioned above, we need both unigram and bigram clusters because we want to incorporate class-based generalization for histories of lengths 1 *and* 2. As we will show below this significantly increases performance. Since the focus of this paper is not on clustering algorithms, reformatting the training corpus as described above (as a sequence of hyphenated bigrams) is a simple way of using SRILM for bigram clustering.

The unique-event clusterings are motivated by the fact that in the Dupont-Rosenfeld model, frequent events are handled by discounted ML estimates. Classes are only needed in cases where an event was not seen or was not frequent enough in the training set. Consequently, we should form clusters not based on all events in the training corpus, but only on events that are rare – because this is the type of event that classes will then be applied to in prediction.

The two unique-event corpora can be thought of as reweighted collections in which each unique event receives the same weight. In practice this means that clustering is mostly influenced by rare events since, on the level of types, most events are rare. As we will see below, rare-event clusterings perform better than all-event clusterings. This is not surprising as the class-based component of the model can only benefit rare events and it is therefore reasonable to estimate this component based on a corpus dominated by rare events.

We started experimenting with reweighted corpora because class sizes become very lopsided in regular SRILM clustering as the size of the base set increases. The reason is that the objective function maximizes mutual information. Highly differentiated classes for frequent words contribute substantially to this objective function whereas putting all rare words in a few large clusters does not hurt the objective much. However, our focus is on using clustering for improving prediction for rare events; this means that the objective function is counterproductive when contexts are frequency-weighted as they occur in the corpus. After overweighting rare contexts, the objective function is more in sync with what we use clusters for in our model.

p_{ML}	maximum likelihood
p_{B}	Brown cluster model
p_{E}	cluster emission probability
p_{T}	cluster transition probability
p_{KN}	KN model
p_{DR}	Dupont-Rosenfeld model
p_{TOP}	top-level interpolation
p_{POLKN}	KN and polynomial discounting
p_{POL0}	polynomial discounting only

Table 2: Key to probability distributions

It is important to note that the same intuition underlies unique-event clustering that also motivates using the "unique-event" distributions $n_{1+}(\bullet w_2^3)/(\sum n_{1+}(\bullet w_2 w))$ and $n_{1+}(\bullet w_3)/(\sum n_{1+}(\bullet w))$ for the backoff distributions in KN. Viewed this way, the basic KN model also uses a unique-event corpus (although a different one) for estimating backoff probabilities.

In all cases, we set the number of clusters to $k = 512$. Our main goal in this paper is to compare different ways of setting up history-length/class interpolated models and we do not attempt to optimize k. We settled on a fixed number of $k = 512$ because Brown et al. (1992) used a total of 1000 classes. 512 unigram classes and 512 bigram classes roughly correspond to this number. We prefer powers of 2 to facilitate efficient storage of cluster ids (one such cluster id must be stored for each unigram and each bigram) and therefore choose $k = 512$. Clustering was performed on an Opteron 8214 processor and took from several minutes for the smallest base sets to more than a week for the largest set of 400,000 items.

To estimate n-gram emission probabilities p_{E}, we first introduce an additional cluster for all unigrams that are not in the base set; emission probabilities are then estimated by maximum likelihood. Cluster transition probabilities p_{T} are computed using add-one smoothing. Both p_{E} and p_{T} are estimated on the raw corpus. The two class distributions are then defined as follows:

$$p_{\text{B}}(w_3|w_1w_2) = p_{\text{T}}(g(w_3)|g(w_1w_2))p_{\text{E}}(w_3|g(w_3))$$
$$p_{\text{B}}(w_3|w_2) = p_{\text{T}}(g(w_3)|g(w_2))p_{\text{E}}(w_3|g(w_3))$$

where $g(v)$ is the class of the uni- or bigram v.

	p_{DR}						p_{TOP}						
	all events			unique events				all events			unique events		
	$\lvert\mathcal{B}_i\rvert$	α_1	α_2	perp.	α_1	α_2	perp.	$\lvert\mathcal{B}_i\rvert$	λ_1	λ_2	perp.	λ_1	λ_2	perp.
1a	1×10^4	.20	.40	87.42	.2	.4	87.41	1b 1×10^4	.020	.03	87.65	.02	.02	87.71
2a	2×10^4	.20	.50	86.97	.2	.5	86.88	2b 2×10^4	.030	.04	87.43	.03	.03	87.47
3a	3×10^4	.10	.40	87.14	.2	.5	86.57	3b 3×10^4	.020	.03	87.52	.03	.03	87.34
4a	4×10^4	.10	.40	87.22	.3	.5	86.31	4b 4×10^4	.010	.04	87.58	.03	.04	87.24
5a	5×10^4	.05	.30	87.54	.3	.6	86.10	5b 5×10^4	.003	.03	87.74	.03	.04	87.15
6a	6×10^4	.01	.30	87.71	.3	.6	85.96	6b 6×10^4	.000	.02	87.82	.03	.04	87.09

Perplexity of KN model: 88.03

Table 3: Optimal parameters for Dupont-Rosenfeld (left) and top-level (right) models on the validation set and perplexity on the validation set. The two tables compare performance when using a class model trained on all events vs a class model trained on unique events. $|\mathcal{B}_1| = |\mathcal{B}_2|$ is the number of unigrams and bigrams in the clusters; e.g., lines 1a and 1b are for models that cluster 10,000 unigrams and 10,000 bigrams.

Table 2 is a key to the probability distributions we use.

5 Results

Table 3 shows the performance of p_{DR} and p_{TOP} for a range of base set sizes $|\mathcal{B}_i|$ and for classes trained on all events and on unique events. Parameters α_i and λ_i are optimized on the validation set. Perplexity is reported for the validation set. All following tables also optimize on the validation set and report results on the validation set. The last table, Table 7, also reports perplexity for the test set.

Table 3 confirms previous findings that classes improve language model performance. All models have a perplexity that is lower than KN (88.03).

When comparing all-event and unique-event clusterings, a clear tendency is apparent. In all-event clustering, the best performance is reached for $|\mathcal{B}_i| = 20000$: perplexity is 86.97 with this base set size for p_{DR} (line 2a) and 87.43 for p_{TOP} (line 2b). In unique-event clustering, performance keeps improving with larger and larger base sets; the best perplexities are obtained for $|\mathcal{B}_i| = 60000$: 85.96 for p_{DR} and 87.09 for p_{TOP} (lines 6a, 6b).

The parameter values also reflect this difference between all-event and unique-event clustering. For unique-event results of p_{DR}, we have $\alpha_1 \geq .2$ and $\alpha_2 \geq .4$ (1a–6a). This indicates that classes and history interpolation are both valuable when the model is backing off. But for all-event clustering, the values of α_i decrease: from a peak of .20 and .50 (2a) to .01 and .30 (6a), indicating that with larger base sets, less and less value can be derived from classes. This again is evidence that rare-event clustering is the correct approach: only clusters derived in rare-event clustering receive high weights α_i in the interpolation.

This effect can also be observed for p_{TOP}: the value of λ_1 (the weight of bigrams) is higher for unique-event clustering than for all-event clustering (with the exception of lines 1b&2b). The quality of bigram clusters seems to be low in all-event clustering when the base set becomes too large.

Perplexity is generally lower for unique-event clustering than for all-event clustering: this is the case for all values of $|\mathcal{B}_i|$ for p_{DR} (1a–6a); and for $|\mathcal{B}_i| > 20000$ for p_{TOP} (3b–6b).

Table 4 compares the two models in two different conditions: (i) b-: using unigram clusters only and (ii) b+: using unigram clusters and bigram clusters. For all events, there is no difference in performance. However, for unique events, the model that includes bigrams (b+) does better than the model without bigrams (b-). The effect is larger for p_{DR} than for p_{TOP} because (for unique events) a larger weight for the unigram model ($\lambda_2 = .05$ instead of $\lambda_2 = .04$) apparently partially compensates for the missing bigram clusters.

Table 3 shows that rare-event models do better than all-event models. Given that training large class models with SRILM on all events would take several weeks or even months, we restrict our direct

	p_{DR}						p_{TOP}					
	all			unique			all			unique		
	α_1	α_2	perp.	α_1	α_2	perp.	λ_1	λ_2	perp.	λ_1	λ_2	perp.
b-		.3	87.71		.5	86.62		.02	87.82		.05	87.26
b+	.01	.3	87.71	.3	.6	85.96	0	.02	87.82	.03	.04	87.09

Table 4: Using both unigram and bigram clusters is better than using unigrams only. Results for $|\mathcal{B}_i| = 60{,}000$.

	p_{DR}			p_{TOP}		
$|\mathcal{B}_i|$	α_1	α_2	perp.	λ_1	λ_2	perp.
1 $6{\times}10^4$	0.3	0.6	85.96	0.03	0.04	87.09
2 $1{\times}10^5$	0.3	0.6	85.59	0.04	0.04	86.93
3 $2{\times}10^5$	0.3	0.6	85.20	0.05	0.04	86.77
4 $4{\times}10^5$	0.3	0.7	85.14	0.05	0.04	86.74

Table 5: Dupont-Rosenfeld and top-level models for $|\mathcal{B}_i| \in \{60000, 100000, 200000, 400000\}$. Clustering trained on unique-event corpora.

comparison of all-event and rare-event models to $|\mathcal{B}_i| \leq 60,000$ in Tables 3-4 and report only rare-event numbers for $|\mathcal{B}_i| > 60,000$ in what follows.

As we can see in Table 5, the trends observed in Table 3 continue as $|\mathcal{B}_i|$ is increased further. For both models, perplexity steadily decreases as $|\mathcal{B}_i|$ is increased from 60,000 to 400,000. (Note that for $|\mathcal{B}_i| = 400000$, the actual size of $\mathcal{B}_1$ is 256,873 since there are only that many words in the training corpus.) The improvements in perplexity become smaller for larger base set sizes, but it is reassuring to see that the general trend continues for large base set sizes. Our explanation is that the class component is focused on rare events and the items that are being added to the clustering for large base sets are all rare events.

The perplexity for p_{DR} is clearly lower than that of p_{TOP}, indicating the superiority of the Dupont-Rosenfeld model.[1]

[1]Dupont and Rosenfeld (1997) found a relatively large improvement of the "global" linear interpolation model – p_{top} in our terminology – compared to the baseline whereas p_{top} performs less well in our experiments. One possible explanation is that our KN baseline is stronger than the word trigram baseline they used.

6 Polynomial discounting

Further comparative analysis of p_{DR} and p_{TOP} revealed that p_{DR} is not uniformly better than p_{TOP}. We found that p_{TOP} does poorly on frequent events. For example, for the history $w_1w_2 =$ *cents a*, the continuation $w_3 =$ *share* dominates. p_{DR} deals well with this situation because $p_{\text{DR}}(w_3|w_1w_2)$ is the discounted ML estimate, with a discount that is small relative to the 10,768 occurrences of *cents a share* in the training set. In the p_{TOP} model on the last line in Table 5, the discounted ML estimate is multiplied by $1 - .05 - .04 = .91$, which results in a much less accurate estimate of $p_{\text{TOP}}(\text{share}|\text{cents a})$.

In contrast, p_{TOP} does well for productive histories, for which it is likely that a continuation unseen in the training set will occur. An example is the history *in the* – almost any adjective or noun can follow. There are 6251 different words that (i) occur after *in the* in the validation set, (ii) did not occur after *in the* in the training set, and (iii) occurred at least 10 times in the training set. Because their training set unigram frequency is at least 10, they have a good chance of being assigned to a class that captures their distributional behavior well and $p_{\text{B}}(w_3|w_1w_2)$ is then likely to be a good estimate. For a history with these properties, it is advantageous to further discount the discounted ML estimates by multiplying them with .91. p_{TOP} then gives the remaining probability mass of .09 to words w_3 whose probability would otherwise be underestimated.

What we have just described is already partially addressed by the KN model – $\gamma(v)$ will be relatively large for a productive history like $v =$ *in the*. However, it looks like the KN discounts are not large enough for productive histories, at least not in a combined history-length/class model. Apparently, when incorporating the strengths of a class-based model into KN, the default discounting mechanism does not reallocate enough probability mass

from high-frequency to low-frequency events. We conclude from this analysis that we need to increase the discount values d for large counts.

We could add a constant to d, but one of the basic premises of the KN model, derived from the assumption that n-gram marginals should be equal to relative frequencies, is that the discount is larger for more frequent n-grams although in many implementations of KN only the cases $c(w_1^3) = 1$, $c(w_1^3) = 2$, and $c(w_1^3) \geq 3$ are distinguished.

This suggests that the ideal discount $d(x)$ in an integrated history-length/class language model should grow monotonically with $c(v)$. The simplest way of implementing this heuristically is a polynomial of form ρx^r where ρ and r are parameters. r controls the rate of growth of the discount as a function of x; ρ is a factor that can be scaled for optimal performance.

The incorporation of the additional polynomial discount into KN is straightforward. We use a discount function $e(x)$ that is the sum of $d(x)$ and the polynomial:

$$e(x) = d(x) + \begin{cases} \rho x^r & \text{for } x \geq 4 \\ 0 & \text{otherwise} \end{cases}$$

where $(e, d) \in \{(e', d'), (e'', d''), (e''', d''')\}$. This model is identical to p_{DR} except that d is replaced with e. We call this model p_{POLKN}. p_{POLKN} directly implements the insight that, when using class-based generalization, discounts for counts $x \geq 4$ should be larger than they are in KN.

We also experiment with a second version of the model:

$$e(x) = \rho x^r$$

This second model, called p_{POL0}, is simpler and does not use KN discounts. It allows us to determine whether a polynomial discount by itself (without using KN discounts in addition) is sufficient.

Results for the two models are shown in Table 6 and compared with the two best models from Table 5, for $|\mathcal{B}_i| = 400{,}000$, classes trained on unique events. p_{POLKN} and p_{POL0} achieve a small improvement in perplexity when compared to p_{DR} (line 3&4 vs 2). This shows that using discounts that are larger than KN discounts for large counts is potentially advantageous.

		α_1/λ_1	α_2/λ_2	ρ	r	perp.
1	p_{TOP}	.05	.04			86.74
2	p_{DR}	.30	.70			85.14
3	p_{POLKN}	.30	.70	.05	.89	85.01
4	p_{POL0}	.30	.70	.80	.41	84.98

Table 6: Results for polynomial discounting compared to p_{DR} and p_{TOP}. $|\mathcal{B}_i| = 400{,}000$, clusters trained on unique events.

	tb:l	model	$\lvert\mathcal{B}_i\rvert$		perplexity val	test
1	3	p_{KN}			88.03	88.28
2	3:6a	p_{DR}	$6{\times}10^4$	ae b+	87.71	87.97
3	3:6a	p_{DR}	$6{\times}10^4$	ue b+	85.96	86.22
4	3:6b	p_{TOP}	$6{\times}10^4$	ae b+	87.82	88.08
5	3:6b	p_{TOP}	$6{\times}10^4$	ue b+	87.09	87.35
6	4	p_{DR}	$6{\times}10^4$	ae b-	87.71	87.97
7	4	p_{DR}	$6{\times}10^4$	ue b-	86.62	86.88
8	4	p_{TOP}	$6{\times}10^4$	ae b-	87.82	88.08
9	4	p_{TOP}	$6{\times}10^4$	ue b-	87.26	87.51
10	5:4	p_{DR}	$2{\times}10^5$	ue b+	85.14	85.39
11	5:4	p_{TOP}	$2{\times}10^5$	ue b+	86.74	86.98
12	6:3	p_{POLKN}	$4{\times}10^5$	ue b+	85.01	85.26
13	6:4	p_{POL0}	$4{\times}10^5$	ue b+	84.98	85.22

Table 7: Performance of key models on validation and test sets. tb:l = Table and line the validation result is taken from. ae/ue = all-event/unique-event. b- = unigrams only. b+ = bigrams and unigrams.

The linear interpolation $\alpha p + (1 - \alpha) q$ of two distributions p and q is a form of linear discounting: p is discounted by $1 - \alpha$ and q by α. See (Katz, 1987; Jelinek, 1990; Ney et al., 1994). It can thus be viewed as polynomial discounting for $r = 1$. Absolute discounting could be viewed as a form of polynomial discounting for $r = 0$. We know of no other work that has explored exponents between 0 and 1 and shown that for this type of exponent, one obtains competitive discounts that could be argued to be simpler than more complex discounts like KN discounts.

6.1 Test set performance

We report the test set performance of the key models we have developed in this paper in Table 7. The experiments were run with the optimal parameters

on the validation set as reported in the table referenced in column "tb:l"; e.g., on line 2 of Table 7, $(\alpha_1, \alpha_2) = (.01, .3)$ as reported on line 6a of Table 3.

There is an almost constant difference between validation and test set perplexities, ranging from +.2 to +.3, indicating that test set results are consistent with validation set results. To test significance, we assigned the 2.8M positions in the test set to 48 different bins according to the majority part-of-speech tag of the word in the training set.[2] We can then compute perplexity for each bin, compare perplexities for different experiments and use the sign test for determining significance. We indicate results that were significant at $p < .05$ ($n = 48$, $k \geq 32$ successes) using a star, e.g., $3 <^* 2$ means that test set perplexity on line 3 is significantly lower than test set perplexity on line 2.

The main findings on the validation set also hold for the test set: (i) Trained on unique events and with a sufficiently large $|\mathcal{B}_i|$, both p_{DR} and p_{TOP} are better than KN: $10 <^* 1$, $11 <^* 1$. (ii) Training on unique events is better than training on all events: $3 <^* 2$, $5 <^* 4$, $7 <^* 6$, $9 <^* 8$. (iii) For unique events, using bigram and unigram classes gives better results than using unigram classes only: $3 <^* 7$. Not significant: $5 < 9$. (iv) The Dupont-Rosenfeld model p_{DR} is better than the top-level model p_{TOP}: $10 <^* 11$. (v) The model POL0 (polynomial discounting) is the best model overall: Not significant: $13 < 12$. (vi) Polynomial discounting is significantly better than KN discounting for the Dupont-Rosenfeld model p_{DR} although the absolute difference in perplexity is small: $13 <^* 10$.

Overall, p_{DR} and p_{POL0} achieve considerable reductions in test set perplexity from 88.28 to 85.39 and 85.22, respectively. The main result of the experiments is that Dupont-Rosenfeld models (which focus on rare events) are better than the standardly used top-level models; and that training classes on unique events is better than training classes on all events.

[2] Words with a rare majority tag (e.g., FW 'foreign word') and unknown words were assigned to a special class OTHER.

7 Conclusion

Our hypothesis was that classes are a generalization mechanism for rare events that serves the same function as history-length interpolation and that classes should therefore be (i) primarily trained on rare events and (ii) receive high weight only if it is likely that a rare event will follow and be weighted in a way analogous to the weighting of lower-order distributions in history-length interpolation.

We found clear statistically significant evidence for both (i) and (ii). (i) Classes trained on unique-event corpora perform better than classes trained on all-event corpora. (ii) The p_{DR} model (which adjusts the interpolation weight given to classes based on the prevalence of nonfrequent events following) is better than top-level model p_{TOP} (which uses a fixed weight for classes). Most previous work on class-based models has employed top-level interpolation. Our results strongly suggest that the Dupont-Rosenfeld model is a superior model.

A comparison of Dupont-Rosenfeld and top-level results suggested that the KN discount mechanism does not discount high-frequency events enough. We empirically determined that better discounts are obtained by letting the discount grow as a function of the count of the discounted event and implemented this as polynomial discounting, an arguably simpler way of discounting than Kneser-Ney discounting. The improvement of polynomial discounts vs. KN discounts was small, but statistically significant.

In future work, we would like to find a theoretical justification for the surprising fact that polynomial discounting does at least as well as Kneser-Ney discounting. We also would like to look at other backoff mechanisms (in addition to history length and classes) and incorporate them into the model, e.g., similarity and topic. Finally, training classes on unique events is an extreme way of highly weighting rare events. We would like to explore training regimes that lie between unique-event clustering and all-event clustering and upweight rare events less.

Acknowledgements. This research was funded by Deutsche Forschungsgemeinschaft (grant SFB 732). We are grateful to Thomas Müller, Helmut Schmid and the anonymous reviewers for their helpful comments.

References

Jeff Bilmes and Katrin Kirchhoff. 2003. Factored language models and generalized parallel backoff. In *HLT-NAACL*.

Peter F. Brown, Vincent J. Della Pietra, Peter V. de Souza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based *n*-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. *CoRR*, cmp-lg/9606011.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Stanley F. Chen. 2009. Shrinking exponential language models. In *HLT/NAACL*, pages 468–476.

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *EACL*, pages 59–66.

Sabine Deligne and Yoshinori Sagisaka. 2000. Statistical language modeling with a class-based *n*-multigram model. *Computer Speech & Language*, 14(3):261–279.

Pierre Dupont and Ronald Rosenfeld. 1997. Lattice based language models. Technical Report CMU-CS-97-173, Carnegie Mellon University.

Ahmad Emami and Frederick Jelinek. 2005. Random clustering for language modeling. In *ICASSP*, volume 1, pages 581–584.

Frederick Jelinek and Robert L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In Edzard S. Gelsema and Laveen N. Kanal, editors, *Pattern Recognition in Practice*, pages 381–397. North-Holland.

Frederick Jelinek. 1990. Self-organized language modeling for speech recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in speech recognition*, pages 450–506. Morgan Kaufmann.

Raquel Justo and M. Inés Torres. 2009. Phrase classes in two-level language models for ASR. *Pattern Analysis & Applications*, 12(4):427–437.

Slava M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *ICASSP*, volume 1, pages 181–184.

Hong-Kwang J. Kuo and Wolfgang Reichl. 1999. Phrase-based language models for speech recognition. In *European Conference on Speech Communication and Technology*, volume 4, pages 1595–1598.

John G. McMahon and Francis J. Smith. 1996. Improving statistical language model performance with automatically generated word hierarchies. *Computational Linguistics*, 22:217–247.

Saeedeh Momtazi and Dietrich Klakow. 2009. A word clustering approach for language model-based sentence retrieval in question answering systems. In *ACM Conference on Information and Knowledge Management*, pages 1911–1914.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8:1–38.

Roi Reichart, Omri Abend, and Ari Rappoport. 2010. Type level clustering evaluation: new measures and a pos induction case study. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 77–87.

Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *EACL 7*, pages 141–148.

Andreas Stolcke. 2002. SRILM - An extensible language modeling toolkit. In *International Conference on Spoken Language Processing*, pages 901–904.

Bernhard Suhm and Alex Waibel. 1994. Towards better language models for spontaneous speech. In *International Conference on Spoken Language Processing*, pages 831–834.

Jakob Uszkoreit and Thorsten Brants. 2008. Distributed word clustering for large scale class-based language modeling in machine translation. In *Annual Meeting of the Association for Computational Linguistics*, pages 755–762.

E.W.D. Whittaker and P.C. Woodland. 2001. Efficient class-based language modelling for very large vocabularies. In *ICASSP*, volume 1, pages 545–548.

Michael Wiegand and Dietrich Klakow. 2008. Optimizing language models for polarity classification. In *ECIR*, pages 612–616.

T. Yokoyama, T. Shinozaki, K. Iwano, and S. Furui. 2003. Unsupervised class-based language model adaptation for spontaneous speech recognition. In *ICASSP*, volume 1, pages 236–239.

Imed Zitouni and Qiru Zhou. 2007. Linearly interpolated hierarchical n-gram language models for speech recognition engines. In Michael Grimm and Kristian Kroschel, editors, *Robust Speech Recognition and Understanding*, pages 301–318. I-Tech Education and Publishing.

Imed Zitouni and Qiru Zhou. 2008. Hierarchical linear discounting class n-gram language models: A multi-level class hierarchy approach. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 4917–4920.

Structural Topic Model for Latent Topical Structure Analysis

Hongning Wang, Duo Zhang, ChengXiang Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana IL, 61801 USA
{wang296, dzhang22, czhai}@cs.uiuc.edu

Abstract

Topic models have been successfully applied to many document analysis tasks to discover topics embedded in text. However, existing topic models generally cannot capture the latent topical structures in documents. Since languages are intrinsically cohesive and coherent, modeling and discovering latent topical transition structures within documents would be beneficial for many text analysis tasks.

In this work, we propose a new topic model, Structural Topic Model, which simultaneously discovers topics and reveals the latent topical structures in text through explicitly modeling topical transitions with a latent first-order Markov chain. Experiment results show that the proposed Structural Topic Model can effectively discover topical structures in text, and the identified structures significantly improve the performance of tasks such as sentence annotation and sentence ordering.

1 Introduction

A great amount of effort has recently been made in applying statistical topic models (Hofmann, 1999; Blei et al., 2003) to explore word co-occurrence patterns, i.e. topics, embedded in documents. Topic models have become important building blocks of many interesting applications (see e.g., (Blei and Jordan, 2003; Blei and Lafferty, 2007; Mei et al., 2007; Lu and Zhai, 2008)).

In general, topic models can discover word clustering patterns in documents and project each document to a latent topic space formed by such word clusters. However, the topical structure in a document, i.e., the internal dependency between the topics, is generally not captured due to the exchangeability assumption (Blei et al., 2003), i.e., the document generation probabilities are invariant to content permutation. In reality, natural language text rarely consists of isolated, unrelated sentences, but rather collocated, structured and coherent groups of sentences (Hovy, 1993). Ignoring such latent topical structures inside the documents means wasting valuable clues about topics and thus would lead to non-optimal topic modeling.

Taking apartment rental advertisements as an example, when people write advertisements for their apartments, it's natural to first introduce *"size"* and *"address"* of the apartment, and then *"rent"* and *"contact"*. Few people would talk about *"restriction"* first. If this kind of topical structures are captured by a topic model, it would not only improve the topic mining results, but, more importantly, also help many other document analysis tasks, such as sentence annotation and sentence ordering.

Nevertheless, very few existing topic models attempted to model such structural dependency among topics. The Aspect HMM model introduced in (Blei and Moreno, 2001) combines pLSA (Hofmann, 1999) with HMM (Rabiner, 1989) to perform document segmentation over text streams. However, Aspect HMM separately estimates the topics in the training set and depends on heuristics to infer the transitional relations between topics. The Hidden Topic Markov Model (HTMM) proposed by (Gruber et al., 2007) extends the traditional topic models by assuming words in each sentence share the same topic assignment, and topics transit between adjacent sentences. However, the transitional structures among topics, i.e., how likely one topic would follow another topic, are not captured in this model.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1526–1535,
Portland, Oregon, June 19-24, 2011.

In this paper, we propose a new topic model, named Structural Topic Model (strTM) to model and analyze both latent topics and topical structures in text documents. To do so, strTM assumes: 1) words in a document are either drawn from a content topic or a *functional* (i.e., background) topic; 2) words in the same sentence share the same content topic; and 3) content topics in the adjacent sentences follow a topic transition that satisfies the first order Markov property. The first assumption distinguishes the semantics of the occurrence of each word in the document, the second requirement confines the unrealistic "bag-of-word" assumption into a tighter unit, and the third assumption exploits the connection between adjacent sentences.

To evaluate the usefulness of the identified topical structures by strTM, we applied strTM to the tasks of sentence annotation and sentence ordering, where correctly modeling the document structure is crucial. On the corpus of 8,031 apartment advertisements from craiglist (Grenager et al., 2005) and 1,991 movie reviews from IMDB (Zhuang et al., 2006), strTM achieved encouraging improvement in both tasks compared with the baseline methods that don't explicitly model the topical structure. The results confirm the necessity of modeling the latent topical structures inside documents, and also demonstrate the advantages of the proposed strTM over existing topic models.

2 Related Work

Topic models have been successfully applied to many problems, e.g., sentiment analysis (Mei et al., 2007), document summarization (Lu and Zhai, 2008) and image annotation (Blei and Jordan, 2003). However, in most existing work, the dependency among the topics is loosely governed by the prior topic distribution, e.g., Dirichlet distribution.

Some work has attempted to capture the interrelationship among the latent topics. Correlated Topic Model (Blei and Lafferty, 2007) replaces Dirichlet prior with logistic Normal prior for topic distribution in each document in order to capture the correlation between the topics. HMM-LDA (Griffiths et al., 2005) distinguishes the short-range syntactic dependencies from long-range semantic dependencies among the words in each document. But in HMM-LDA, only the latent variables for the syntactic classes are treated as a locally dependent sequence, while latent topics are treated the same as in other topic models. Chen et al. introduced the generalized Mallows model to constrain the latent topic assignments (Chen et al., 2009). In their model, they assume there exists a canonical order among the topics in the collection of related documents and the same topics are forced not to appear in disconnected portions of the topic sequence in one document (sampling without replacement). Our method relaxes this assumption by only postulating transitional dependency between topics in the adjacent sentences (sampling with replacement) and thus potentially allows a topic to appear multiple times in disconnected segments. As discussed in the previous section, HTMM (Gruber et al., 2007) is the most similar model to ours. HTMM models the document structure by assuming words in the same sentence share the same topic assignment and successive sentences are more likely to share the same topic. However, HTMM only loosely models the transition between topics as a binary relation: the same as the previous sentence's assignment or draw a new one with a certain probability. This simplified coarse modeling of dependency could not fully capture the complex structure across different documents. In contrast, our strTM model explicitly captures the regular topic transitions by postulating the first order Markov property over the topics.

Another line of related work is discourse analysis in natural language processing: discourse segmentation (Sun et al., 2007; Galley et al., 2003) splits a document into a linear sequence of multi-paragraph passages, where lexical cohesion is used to link together the textual units; discourse parsing (Soricut and Marcu, 2003; Marcu, 1998) tries to uncover a more sophisticated hierarchical coherence structure from text to represent the entire discourse. One work in this line that shares a similar goal as ours is the content models (Barzilay and Lee, 2004), where an HMM is defined over text spans to perform information ordering and extractive summarization. A deficiency of the content models is that the identification of clusters of text spans is done separately from transition modeling. Our strTM addresses this deficiency by defining a generative process to simultaneously capture the topics and the transitional re-

lationship among topics: allowing topic modeling and transition modeling to reinforce each other in a principled framework.

3 Structural Topic Model

In this section, we formally define the Structural Topic Model (strTM) and discuss how it captures the latent topics and topical structures within the documents simultaneously. From the theory of linguistic analysis (Kamp, 1981), we know that document exhibits internal structures, where structural segments encapsulate semantic units that are closely related. In strTM, we treat a sentence as the basic structure unit, and assume all the words in a sentence share the same topical aspect. Besides, two adjacent segments are assumed to be highly related (capturing cohesion in text); specifically, in strTM we pose a strong transitional dependency assumption among the topics: the choice of topic for each sentence directly depends on the previous sentence's topic assignment, i.e., first order Markov property. Moveover, taking the insights from HMM-LDA that not all the words are content conveying (some of them may just be a result of syntactic requirement), we introduce a dummy *functional* topic z_B for every sentence in the document. We use this functional topic to capture the document-independent word distribution, i.e., corpus background (Zhai et al., 2004). As a result, in strTM, every sentence is treated as a mixture of content and functional topics.

Formally, we assume a corpus consists of *D* documents with a vocabulary of size *V*, and there are *k* content topics embedded in the corpus. In a given document *d*, there are *m* sentences and each sentence *i* has N_i words. We assume the topic transition probability $p(z|z')$ is drawn from a Multinomial distribution *Mul(*$\alpha_{z'}$*)*, and the word emission probability under each topic $p(w|z)$ is drawn from a Multinomial distribution *Mul(*β_z*)*.

To get a unified description of the generation process, we add another dummy topic *T-START* in strTM, which is the initial topic with position "-1" for every document but does not emit any words. In addition, since our functional topic is assumed to occur in all the sentences, we don't need to model its transition with other content topics. We use a Binomial variable π to control the proportion between content and functional topics in each sentence. Therefore, there are *k*+1 topic transitions, one for *T-START* and others for *k* content topics; and *k* emission probabilities for the content topics, with an additional one for the functional topic z_B (in total *k*+1 emission probability distributions).

Conditioned on the model parameters $\Theta = (\alpha, \beta, \pi)$, the generative process of a document in strTM can be described as follows:

1. For each sentence s_i in document *d*:
   (a) Draw topic z_i from Multinomial distribution conditioned on the previous sentence s_{i-1}'s topic assignment z_{i-1}:
   $z_i \sim Mul(\alpha_{z_{i-1}})$
   (b) Draw each word w_{ij} in sentence s_i from the mixture of content topic z_i and functional topic z_B:
   $w_{ij} \sim \pi p(w_{ij}|\beta, z_i) + (1-\pi)p(w_{ij}|\beta, z_B)$

The joint probability of sentences and topics in one document defined by strTM is thus given by:

$$p(S_0, S_1, \ldots, S_m, \mathbf{z}|\alpha, \beta, \pi) = \prod_{i=1}^{m} p(z_i|\alpha, z_{i-1}) p(S_i|z_i) \quad (1)$$

where the topic to sentence emission probability is defined as:

$$p(S_i|z_i) = \prod_{j=0}^{N_i} \left[\pi p(w_{ij}|\beta, z_i) + (1-\pi) p(w_{ij}|\beta, z_B)\right] \quad (2)$$

This process is graphically illustrated in Figure 1.

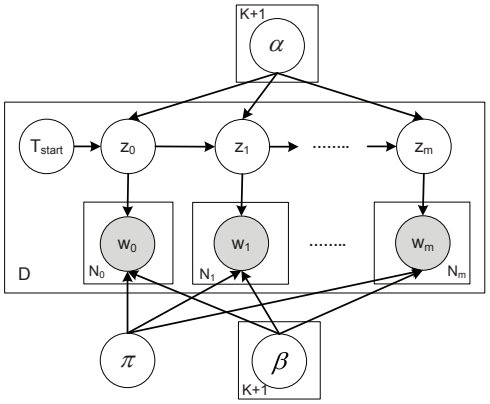


Figure 1: Graphical Representation of strTM.

From the definition of strTM, we can see that the document structure is characterized by a document-specific topic chain, and forcing the words in one

sentence to share the same content topic ensures semantic cohesion of the mined topics. Although we do not directly model the topic mixture for each document as the traditional topic models do, the word co-occurrence patterns within the same document are captured by topic propagation through the transitions. This can be easily understood when we write down the posterior probability of the topic assignment for a particular sentence:

$$\begin{aligned}
&p(z_i|S_0, S_1, \ldots, S_m, \Theta) \\
=&\frac{p(S_0, S_1, \ldots, S_m|z_i, \Theta)p(z_i)}{p(S_0, S_1, \ldots, S_m)} \\
\propto& p(S_0, S_1, \ldots, S_i, z_i) \times p(S_{i+1}, S_{i+2}, \ldots, S_m|z_i) \\
=&\sum_{z_{i-1}} p(S_0, \ldots, S_{i-1}, z_{i-1})p(z_i|z_{i-1})p(S_i|z_i) \\
&\times \sum_{z_{i+1}} p(S_{i+1}, \ldots, S_m|z_{i+1})p(z_{i+1}|z_i) \qquad (3)
\end{aligned}$$

The first part of Eq(3) describes the recursive influence on the choice of topic for the i*th* sentence from its preceding sentences, while the second part captures how the succeeding sentences affect the current topic assignment. Intuitively, when we need to decide a sentence's topic, we will look "backward" and "forward" over all the sentences in the document to determine a "suitable" one. In addition, because of the first order Markov property, the local topical dependency gets more emphasis, i.e., they are interacting directly through the transition probabilities $p(z_i|z_{i-1})$ and $p(z_{i+1}|z_i)$. And such interaction on sentences farther away would get damped by the multiplication of such probabilities. This result is reasonable, especially in a long document, since neighboring sentences are more likely to cover similar topics than two sentences far apart.

4 Posterior Inference and Parameter Estimation

The chain structure in strTM enables us to perform exact inference: posterior distribution can be efficiently calculated by the forward-backward algorithm, the optimal topic sequence can be inferred using the Viterbi algorithm, and parameter estimation can be solved by the Expectation Maximization (EM) algorithm. More technical details can be found in (Rabiner, 1989). In this section, we only discuss strTM-specific procedures.

In the E-Step of EM algorithm, we need to collect the expected count of a sequential topic pair (z, z') and a topic-word pair (z, w) to update the model parameters α and β in the M-Step. In strTM, $E[c(z, z')]$ can be easily calculated by forward-backward algorithm. But we have to go one step further to fetch the required sufficient statistics for $E[c(z, w)]$, because our emission probabilities are defined over sentences.

Through forward-backward algorithm, we can get the posterior probability $p(s_i, z|d, \Theta)$. In strTM, words in one sentence are independently drawn from either a specific content topic z or functional topic z_B according to the mixture weight π. Therefore, we can accumulate the expected count of (z, w) over all the sentences by:

$$E[c(z, w)] = \sum_{d, s \in d} \frac{\pi p(w|z)p(s, z|d, \Theta)c(w, s)}{\pi p(w|z) + (1 - \pi)p(w|z_B)} \qquad (4)$$

where $c(w, s)$ indicates the frequency of word w in sentence s.

Eq(4) can be easily explained as follows. Since we already observe topic z and sentence s co-occur with probability $p(s, z|d, \Theta)$, each word w in s should share the same probability of being observed with content topic z. Thus the expected count of $c(z, w)$ in this sentence would be $p(s, z|d, \Theta)c(w, s)$. However, since each sentence is also associated with the functional topic z_B, the word w may also be drawn from z_B. By applying the Bayes' rule, we can properly reallocate the expected count of $c(z, w)$ by Eq(4). The same strategy can be applied to obtain $E[c(z_B, w)]$.

As discussed in (Johnson, 2007), to avoid the problem that EM algorithm tends to assign a uniform word/state distribution to each hidden state, which deviates from the heavily skewed word/state distributions empirically observed, we can apply a Bayesian estimation approach for strTM. Thus we introduce prior distributions over the topic transition *Mul($\alpha_{z'}$)* and emission probabilities *Mul(β_z)*, and use the Variational Bayesian (VB) (Jordan et al., 1999) estimator to obtain a model with more skewed word/state distributions.

Since both the topic transition and emission probabilities are Multinomial distributions in strTM, the conjugate Dirichlet distribution is the natural

choice for imposing a prior on them (Diaconis and Ylvisaker, 1979). Thus, we further assume:

$$\alpha_z \sim Dir(\eta) \quad (5)$$

$$\beta_z \sim Dir(\gamma) \quad (6)$$

where we use exchangeable Dirichlet distributions to control the sparsity of α_z and β_z. As η and γ approach zero, the prior strongly favors the models in which each hidden state emits as few words/states as possible. In our experiments, we empirically tuned η and γ on different training corpus to optimize log-likelihood.

The resulting VB estimation only requires a minor modification to the M-Step in the original EM algorithm:

$$\bar{\alpha}_z = \frac{\Phi(E[c(z', z)] + \eta)}{\Phi(E[c(z)] + k\eta)} \quad (7)$$

$$\bar{\beta}_z = \frac{\Phi(E[c(w, z)] + \gamma)}{\Phi(E[c(z)] + V\gamma)} \quad (8)$$

where $\Phi(x)$ is the exponential of the first derivative of the log-gamma function.

The optimal setting of π for the proportion of content topics in the documents is empirically tuned by cross-validation over the training corpus to maximize the log-likelihood.

5 Experimental Results

In this section, we demonstrate the effectiveness of strTM in identifying latent topical structures from documents, and quantitatively evaluate how the mined topic transitions can help the tasks of sentence annotation and sentence ordering.

5.1 Data Set

We used two different data sets for evaluation: apartment advertisements (Ads) from (Grenager et al., 2005) and movie reviews (Review) from (Zhuang et al., 2006).

The Ads data consists of 8,767 advertisements for apartment rentals crawled from Craigslist website. 302 of them have been labeled with 11 fields, including *size*, *feature*, *address*, etc., on the sentence level. The review data contains 2,000 movie reviews discussing 11 different movies from IMDB. These reviews are manually labeled with 12 movie feature labels (We didn't use the additional opinion annotations in this data set.) , e.g., *VP* (vision effects), *MS* (music and sound effects), etc., also on the sentences, but the annotations in the review data set is much sparser than that in the Ads data set (see in Table 1). The sentence-level annotations make it possible to quantitatively evaluate the discovered topic structures.

We performed simple preprocessing on these two data sets: 1) removed a standard list of stop words, terms occurring in less than 2 documents; 2) discarded the documents with less than 2 sentences; 3) aggregated sentence-level annotations into document-level labels (binary vector) for each document. Table 1 gives a brief summary on these two data sets after the processing.

	Ads	*Review*
Document Size	8,031	1,991
Vocabulary Size	21,993	14,507
Avg Stn/Doc	8.0	13.9
Avg Labeled Stn/Doc	7.1*	5.1
Avg Token/Stn	14.1	20.0

*Only in 302 labeled ads

Table 1: Summary of evaluation data set

5.2 Topic Transition Modeling

First, we qualitatively demonstrate the topical structure identified by strTM from Ads data[1]. We trained strTM with 11 content topics in Ads data set, used word distribution under each class (estimated by maximum likelihood estimator on document-level labels) as priors to initialize the emission probability *Mul(β_z)* in Eq(6), and treated document-level labels as the prior for transition from T-START in each document, so that the mined topics can be aligned with the predefined class labels. Figure 2 shows the identified topics and the transitions among them. To get a clearer view, we discarded the transitions below a threshold of 0.1 and removed all the isolated nodes.

From Figure 2, we can find some interesting topical structures. For example, people usually start with *"size"*, *"features"* and *"address"*, and end with *"contact"* information when they post an apart-

[1]Due to the page limit, we only show the result in Ads data set.

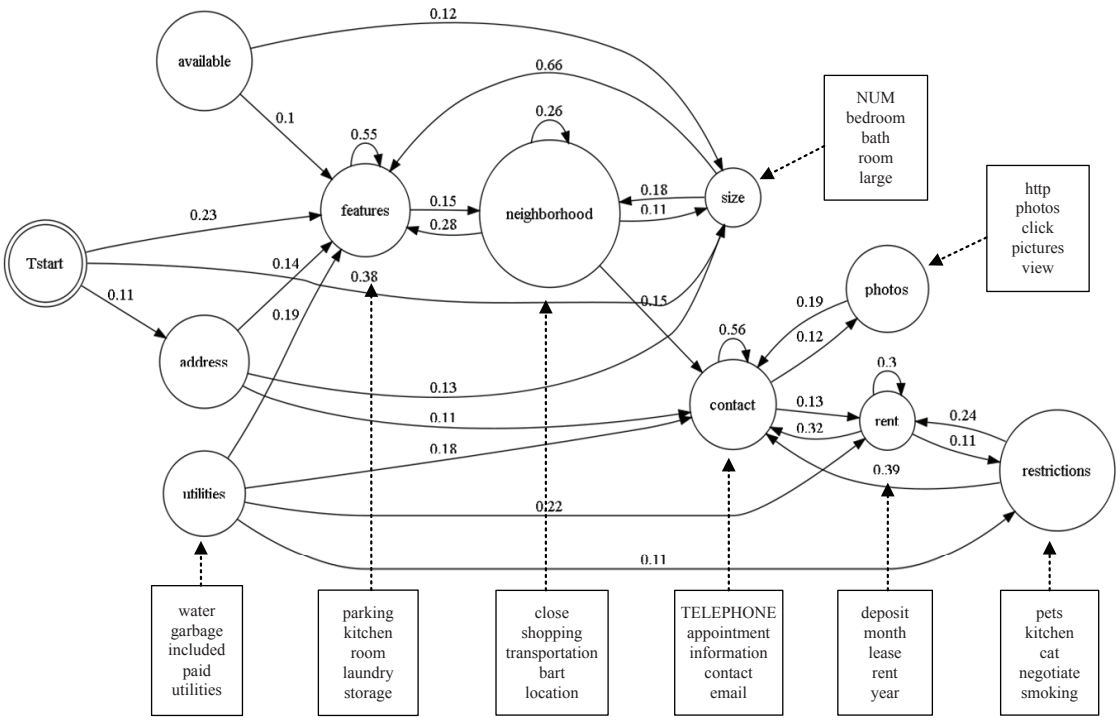


Figure 2: Estimated topics and topical transitions in Ads data set

ment ads. Also, we can discover a strong transition from *"size"* to *"features"*. This intuitively makes sense because people usually write "it's a two bedrooms apartment" first, and then describe other *"features"* about the apartment. The mined topics are also quite meaningful. For example, *"restrictions"* are usually put over pets and smoking, and parking and laundry are always the major *"features"* of an apartment.

To further quantitatively evaluate the estimated topic transitions, we used *Kullback-Leibler* (*KL*) divergency between the estimated transition matrix and the "ground-truth" transition matrix as the metric. Each element of the "ground-truth" transition matrix was calculated by Eq(9), where $c(z, z')$ denotes how many sentences annotated by z' immediately precede one annotated by z. δ is a smoothing factor, and we fixed it to 0.01 in the experiment.

$$\bar{p}(z|z') = \frac{c(z, z') + \delta}{c(z) + k\delta} \tag{9}$$

The *KL* divergency between two transition matrices is defined in Eq(10). Because we have a $k \times k$ transition matrix (T_{start} is not included), we calculated the average *KL* divergency against the ground-truth over all the topics:

$$avgKL = \frac{\sum_{i=1}^{k} KL(p(z|z'_i)||\bar{p}(z|z'_i)) + KL(\bar{p}(z|z'_i)||p(z|z'_i))}{2k} \tag{10}$$

where $\bar{p}(z|z')$ is the ground-truth transition probability estimated by Eq(9), and $p(z|z')$ is the transition probability given by the model.

We used pLSA (Hofmann, 1999), latent permutation model (lPerm) (Chen et al., 2009) and HTMM (Gruber et al., 2007) as the baseline methods for the comparison. Because none of these three methods can generate a topic transition matrix directly, we extended them a little bit to achieve this goal. For pLSA, we used the document-level labels as priors for the topic distribution in each document, so that the estimated topics can be aligned with the predefined class labels. After the topics were estimated, for each sentence we selected the topic that had the highest posterior probability to generate the sentence as its class label. For lPerm and HTMM, we used Kuhn-Munkres algorithm (Lovász and Plummer, 1986) to find the optimal topic-to-class alignment based on the sentence-level annotations. After the sentences were annotated with class labels, we estimated the topic transition matrices for all of these three methods by Eq(9).

Since only a small portion of sentences are annotated in the Review data set, very few neighboring sentences are annotated at the same time, which introduces many noisy transitions. As a result, we only performed the comparison on the Ads data set. The "ground-truth" transition matrix was estimated based on all the 302 annotated ads.

	pLSA+prior	lPerm	HTMM	strTM
avgKL	0.743	1.101	0.572	0.372
p-value	0.023	1e-4	0.007	–

Table 2: Comparison of estimated topic transitions on Ads data set

In Table 2, the p-value was calculated based on t-test of the *KL* divergency between each topic's transition probability against strTM. From the results, we can see that avgKL of strTM is smaller than the other three baseline methods, which means the estimated transitional relation by strTM is much closer to the ground-truth transition. This demonstrates that strTM captures the topical structure well, compared with other baseline methods.

5.3 Sentence Annotation

In this section, we demonstrate how the identified topical structure can benefit the task of sentence annotation. Sentence annotation is one step beyond the traditional document classification task: in sentence annotation, we want to predict the class label for each sentence in the document, and this will be helpful for other problems, including extractive summarization and passage retrieval. However, the lack of detailed annotations on sentences greatly limits the effectiveness of the supervised classification methods, which have been proved successful on document classifications.

In this experiment, we propose to use strTM to address this annotation task. One advantage of strTM is that it captures the topic transitions on the sentence level within documents, which provides a regularization over the adjacent predictions.

To examine the effectiveness of such structural regularization, we compared strTM with four baseline methods: pLSA, lPerm, HTMM and Naive Bayes model. The sentence labeling approaches for strTM, pLSA, lPerm and HTMM have been discussed in the previous section. As for Naive Bayes model, we used EM algorithm [2] with both labeled and unlabeled data for the training purpose (we used the same unigram features as in topics models). We set weights for the unlabeled data to be 10^{-3} in Naive Bayes with EM.

The comparison was performed on both data sets. We set the size of topics in each topic model equal to the number of classes in each data set accordingly. To tackle the situation where some sentences in the document are not strictly associated with any classes, we introduced an additional *NULL* content topic in all the topic models. During the training phase, none of the methods used the sentence-level annotations in the documents, so that we treated the whole corpus as the training and testing set.

To evaluate the prediction performance, we calculated accuracy, recall and precision based on the correct predictions over the sentences, and averaged over all the classes as the criterion.

Model	Accuracy	Recall	Precison
pLSA+prior	0.432	0.649	0.457
lPerm	0.610	0.514	0.471
HTMM	0.606	0.588	0.443
NB+EM	0.528	0.337	0.612
strTM	**0.747**	**0.674**	**0.620**

Table 3: Sentence annotation performance on Ads data set

Model	Accuracy	Recall	Precison
pLSA+prior	0.342	0.278	0.250
lPerm	0.286	0.205	0.184
HTMM	0.369	0.131	0.149
NB+EM	0.341	0.354	**0.431**
strTM	**0.541**	**0.398**	0.323

Table 4: Sentence annotation performance on Review data set

Annotation performance on the two data sets is shown in Table 3 and Table 4. We can see that strTM outperformed all the other baseline methods on most of the metrics: strTM has the best accuracy and recall on both of the two data sets. The improvement confirms our hypothesis that besides solely depending on the local word patterns to perform predic-

[2]Mallet package: `http://mallet.cs.umass.edu/`

tions, adjacent sentences provide a structural regularization in strTM (see Eq(3)). Compared with lPerm, which postulates a strong constrain over the topic assignment (sampling without replacement), strTM performed much better on both of these two data sets. This validates the benefit of modeling local transitional relation compared with the global ordering. Besides, strTM achieved over 46% accuracy improvement compared with the second best HTMM in the review data set. This result shows the advantage of explicitly modeling the topic transitions between neighbor sentences instead of using a binary relation to do so as in HTMM.

To further testify how the identified topical structure can help the sentence annotation task, we first randomly removed 100 annotated ads from the training corpus and used them as the testing set. Then, we used the ground-truth topic transition matrix estimated from the training data to order those 100 ads according to their fitness scores under the ground-truth topic transition matrix, which is defined in Eq(11). We tested the prediction accuracy of different models over two different partitions, top 50 and bottom 50, according to this order.

$$fitness(d) = \frac{1}{|d|}\sum_{i=0}^{|d|}\log \bar{p}(t_i|t_{i-1}) \qquad (11)$$

where t_i is the class label for i*th* sentence in document d, $|d|$ is the number of sentences in document d, and $\bar{p}(t_i|t_{i-1})$ is the transition probability estimated by Eq(9).

	Top 50	p-value	Bot 50	p-value
pLSA+prior	0.496	4e-12	0.542	0.004
lPerm	0.669	0.003	0.505	8e-4
HTMM	0.683	0.004	0.579	0.003
NB + EM	0.492	1e-12	0.539	0.002
strTM	0.752	–	0.644	–

Table 5: Sentence annotation performance according to structural fitness

The results are shown in Table 5. From this table, we can find that when the testing documents follow the regular patterns as in the training data, i.e., top 50 group, strTM performs significantly better than the other methods; when the testing documents don't share such structure, i.e., bottom 50 group, strTM's performance drops. This comparison confirms that when a testing document shares similar topic structure as the training data, the topical transitions captured by strTM can help the sentence annotation task a lot. In contrast, because pLSA and Naive Bayes don't depend on the document's structure, their performance does not change much over these two partitions.

5.4 Sentence Ordering

In this experiment, we illustrate how the learned topical structure can help us better arrange sentences in a document. Sentence ordering, or text planning, is essential to many text synthesis applications, including multi-document summarization (Goldstein et al., 2000) and concept-to-text generation (Barzilay and Lapata, 2005).

In strTM, we evaluate all the possible orderings of the sentences in a given document and selected the optimal one which gives the highest generation probability:

$$\bar{\sigma}(m) = \arg\max_{\sigma(m)} \sum_{\mathbf{z}} p(S_{\sigma[0]}, S_{\sigma[1]}, \ldots, S_{\sigma[m]}, \mathbf{z}|\Theta) \qquad (12)$$

where $\sigma(m)$ is a permutation of 1 to m, and $\sigma[i]$ is the i*th* element in this permutation.

To quantitatively evaluate the ordering result, we treated the original sentence order (OSO) as the perfect order and used *Kendall's* $\tau(\sigma)$ (Lapata, 2006) as the evaluation metric to compute the divergency between the optimum ordering given by the model and OSO. *Kendall's* $\tau(\sigma)$ is widely used in information retrieval domain to measure the correlation between two ranked lists and it indicates how much an ordering differs from OSO, which ranges from 1 (perfect matching) to -1 (totally mismatching).

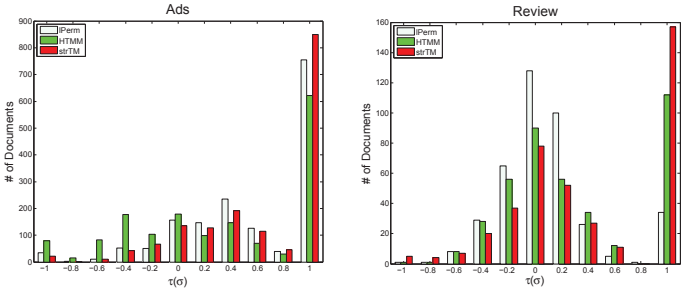
Since only the HTMM and lPerm take the order of sentences in the document into consideration, we used them as the baselines in this experiment. We ranked OSO together with candidate permutations according to the corresponding model's generation probability. However, when the size of documents becomes larger, it's infeasible to permutate all the orderings, therefore we randomly permutated 200 possible orderings of sentences as candidates when there were more than 200 possible candidates. The

2bedroom 1bath in very nice complex! Pool, carport, laundry facilities!! *Call Don (650)207-5769 to see!* Great location!! Also available, 2bed.2bath for $1275 in same complex.	$\Longrightarrow$	2bedroom 1bath in very nice complex! Pool, carport, laundry facilities!! Great location!! Also available, 2bed.2bath for $1275 in same complex. *Call Don (650)207-5769 to see!*
2 bedrooms 1 bath + a famyly room in a cul-de-sac location. *Please drive by and call Marilyn for appointment 650-652-5806.* Address: 517 Price Way, Vallejo. No Pets Please!	$\Longrightarrow$	2 bedrooms 1 bath + a famyly room in a cul-de-sac location. Address: 517 Price Way, Vallejo. No Pets Please! *Please drive by and call Marilyn for appointment 650-652-5806.*

Table 6: Sample results for document ordering by strTM

experiment was performed on both data sets with 80% data for training and the other 20% for testing.

We calculated the $\tau(\sigma)$ of all these models for each document in the two data sets and visualized the distribution of $\tau(\sigma)$ in each data set with histogram in Figure 3. From the results, we could observe that strTM's $\tau(\sigma)$ is more skewed towards the positive range (with mean 0.619 in Ads data set and 0.398 in review data set) than lPerm's results (with mean 0.566 in Ads data set and 0.08 in review data set) and HTMM's results (with mean 0.332 in Ads data set and 0.286 in review data set). This indicates that strTM better captures the internal structure within the documents.



(a) Ads (b) Review

Figure 3: Document Ordering Performance in $\tau(\sigma)$.

We see that all methods performed better on the Ads data set than the review data set, suggesting that the topical structures are more coherent in the Ads data set than the review data. Indeed, in the Ads data, strTM perfectly recovered 52.9% of the original sentence order. When examining some mismatched results, we found that some of them were due to an "outlier" order given by the original document (in comparison to the "regular" patterns in the set). In Table 6, we show two such examples where we see the learned structure "suggested" to move the contact information to the end, which intuitively gives us a more regular organization of the ads. It's hard to say that in this case, the system's ordering is inferior to that of the original; indeed, the system order is arguably more natural than the original order.

6 Conclusions

In this paper, we proposed a new structural topic model (strTM) to identify the latent topical structure in documents. Different from the traditional topic models, in which exchangeability assumption precludes them to capture the structure of a document, strTM captures the topical structure explicitly by introducing transitions among the topics. Experiment results show that both the identified topics and topical structure are intuitive and meaningful, and they are helpful for improving the performance of tasks such as sentence annotation and sentence ordering, where correctly recognizing the document structure is crucial. Besides, strTM is shown to outperform not only the baseline topic models that fail to model the dependency between the topics, but also the semi-supervised Naive Bayes model for the sentence annotation task.

Our work can be extended by incorporating richer features, such as named entity and co-reference, to enhance the model's capability of structure finding. Besides, advanced NLP techniques for document analysis, e.g., shallow parsing, may also be used to further improve structure finding.

7 Acknowledgments

We thank the anonymous reviewers for their useful comments. This material is based upon work supported by the National Science Foundation under Grant Numbers IIS-0713581 and CNS-0834709, and NASA grant NNX08AC35A.

References

R. Barzilay and M. Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 331–338.

R. Barzilay and L. Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceedings of HLT-NAACL*, pages 113–120.

D.M. Blei and M.I. Jordan. 2003. Modeling annotated data. In *Proceedings of the 26th annual international ACM SIGIR conference*, pages 127–134.

D.M. Blei and J.D. Lafferty. 2007. A correlated topic model of science. *The Annals of Applied Statistics*, 1(1):17–35.

D.M. Blei and P.J. Moreno. 2001. Topic segmentation with an aspect hidden Markov model. In *Proceedings of the 24th annual international ACM SIGIR conference*, page 348. ACM.

D.M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3(2-3):993 – 1022.

H. Chen, SRK Branavan, R. Barzilay, and D.R. Karger. 2009. Global models of document structure using latent permutations. In *Proceedings of HLT-NAACL*, pages 371–379.

P. Diaconis and D. Ylvisaker. 1979. Conjugate priors for exponential families. *The Annals of statistics*, 7(2):269–281.

M. Galley, K. McKeown, E. Fosler-Lussier, and H. Jing. 2003. Discourse segmentation of multi-party conversation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 562–569.

J. Goldstein, V. Mittal, J. Carbonell, and M. Kantrowitz. 2000. Multi-document summarization by sentence extraction. In *NAACL-ANLP 2000 Workshop on Automatic summarization*, pages 40–48.

T. Grenager, D. Klein, and C.D. Manning. 2005. Unsupervised learning of field segmentation models for information extraction. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 371–378.

T.L. Griffiths, M. Steyvers, D.M. Blei, and J.B. Tenenbaum. 2005. Integrating topics and syntax. *Advances in neural information processing systems*, 17:537–544.

Amit Gruber, Yair Weiss, and Michal Rosen-Zvi. 2007. Hidden topic markov models. volume 2, pages 163–170.

T. Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57.

E.H. Hovy. 1993. Automated discourse generation using discourse structure relations. *Artificial intelligence*, 63(1-2):341–385.

M. Johnson. 2007. Why doesn't EM find good HMM POS-taggers. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305.

M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul. 1999. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

H. Kamp. 1981. A theory of truth and semantic representation. *Formal methods in the study of language*, 1:277–322.

M. Lapata. 2006. Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics*, 32(4):471–484.

L. Lovász and M.D. Plummer. 1986. *Matching theory*. Elsevier Science Ltd.

Y. Lu and C. Zhai. 2008. Opinion integration through semi-supervised topic modeling. In *Proceeding of the 17th international conference on World Wide Web*, pages 121–130.

Daniel Marcu. 1998. The rhetorical parsing of natural language texts. In *ACL '98*, pages 96–103.

Q. Mei, X. Ling, M. Wondra, H. Su, and C.X. Zhai. 2007. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the 16th international conference on World Wide Web*, pages 171–180.

L.R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

R. Soricut and D. Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the NAACL-HTC*, pages 149–156.

B. Sun, P. Mitra, C.L. Giles, J. Yen, and H. Zha. 2007. Topic segmentation with shared topic detection and alignment of multiple documents. In *Proceedings of the 30th ACM SIGIR*, pages 199–206.

ChengXiang Zhai, Atulya Velivelli, and Bei Yu. 2004. A cross-collection mixture model for comparative text minning. In *Proceeding of the 10th ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 743–748.

L. Zhuang, F. Jing, and X.Y. Zhu. 2006. Movie review mining and summarization. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 43–50.

Automatic Labelling of Topic Models

Jey Han Lau,[♠♡] **Karl Grieser,**[♡] **David Newman,**[♠♢] and **Timothy Baldwin**[♠♡]
♠ NICTA Victoria Research Laboratory
♡ Dept of Computer Science and Software Engineering, University of Melbourne
♢ Dept of Computer Science, University of California Irvine
jhlau@csse.unimelb.edu.au, kgrieser@csse.unimelb.edu.au, newman@uci.edu, tb@ldwin.net

Abstract

We propose a method for automatically labelling topics learned via LDA topic models. We generate our label candidate set from the top-ranking topic terms, titles of Wikipedia articles containing the top-ranking topic terms, and sub-phrases extracted from the Wikipedia article titles. We rank the label candidates using a combination of association measures and lexical features, optionally fed into a supervised ranking model. Our method is shown to perform strongly over four independent sets of topics, significantly better than a benchmark method.

1 Introduction

Topic modelling is an increasingly popular framework for simultaneously soft-clustering terms and documents into a fixed number of "topics", which take the form of a multinomial distribution over terms in the document collection (Blei et al., 2003). It has been demonstrated to be highly effective in a wide range of tasks, including multi-document summarisation (Haghighi and Vanderwende, 2009), word sense discrimination (Brody and Lapata, 2009), sentiment analysis (Titov and McDonald, 2008), information retrieval (Wei and Croft, 2006) and image labelling (Feng and Lapata, 2010).

One standard way of interpreting a topic is to use the marginal probabilities $p(w_i|t_j)$ associated with each term w_i in a given topic t_j to extract out the 10 terms with highest marginal probability. This results in term lists such as:[1]

stock market investor fund trading investment firm exchange companies share

[1]Here and throughout the paper, we will represent a topic t_j via its ranking of top-10 topic terms, based on $p(w_i|t_j)$.

which are clearly associated with the domain of stock market trading. The aim of this research is to automatically generate topic labels which explicitly identify the semantics of the topic, i.e. which take us from a list of terms requiring interpretation to a single label, such as STOCK MARKET TRADING in the above case.

The approach proposed in this paper is to first generate a topic label candidate set by: (1) sourcing topic label candidates from Wikipedia by querying with the top-N topic terms; (2) identifying the top-ranked document titles; and (3) further post-processing the document titles to extract sub-strings. We translate each topic label into features extracted from Wikipedia, lexical association with the topic terms in Wikipedia documents, and also lexical features for the component terms. This is used as the basis of a support vector regression model, which ranks each topic label candidate.

Our contributions in this work are: (1) the generation of a novel evaluation framework and dataset for topic label evaluation; (2) the proposal of a method for both generating and scoring topic label candidates; and (3) strong in- and cross-domain results across four independent document collections and associated topic models, demonstrating the ability of our method to automatically label topics with remarkable success.

2 Related Work

Topics are conventionally interpreted via their top-N terms, ranked based on the marginal probability $p(w_i|t_j)$ in that topic (Blei et al., 2003; Griffiths and Steyvers, 2004). This entails a significant cognitive load in interpretation, prone to subjectivity. Topics are also sometimes presented with manual post-hoc labelling for ease of interpretation in research publications (Wang and McCallum, 2006; Mei et al.,

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1536–1545,
Portland, Oregon, June 19-24, 2011.

2006). This has obvious disadvantages in terms of subjectivity, and lack of reproducibility/automation.

The closest work to our method is that of Mei et al. (2007), who proposed various unsupervised approaches for automatically labelling topics, based on: (1) generating label candidates by extracting either bigrams or noun chunks from the document collection; and (2) ranking the label candidates based on KL divergence with a given topic. Their proposed methodology generates a generic list of label candidates for *all* topics using only the document collection. The best method uses bigrams exclusively, in the form of the top-1000 bigrams based on the Student's t-test. We reimplement their method and present an empirical comparison in Section 5.3.

In other work, Magatti et al. (2009) proposed a method for labelling topics induced by a hierarchical topic model. Their label candidate set is the Google Directory (gDir) hierarchy, and label selection takes the form of ontological alignment with gDir. The experiments presented in the paper are highly preliminary, although the results certainly show promise. However, the method is only applicable to a hierarchical topic model and crucially relies on a pre-existing ontology and the class labels contained therein.

Pantel and Ravichandran (2004) addressed the more specific task of labelling a semantic class by applying Hearst-style lexico-semantic patterns to each member of that class. When presented with semantically homogeneous, fine-grained near-synonym clusters, the method appears to work well. With topic modelling, however, the top-ranking topic terms tended to be *associated* and not lexically *similar* to one another. It is thus highly questionable whether their method could be applied to topic models, but it would certainly be interesting to investigate whether our model could conversely be applied to the labelling of sets of near-synonyms.

In recent work, Lau et al. (2010) proposed to approach topic labelling via best term selection, i.e. selecting one of the top-10 topic terms to label the overall topic. While it is often possible to label topics with topic terms (as is the case with the stock market topic above), there are also often cases where topic terms are not appropriate as labels. We reuse a selection of the features proposed by Lau et al. (2010), and return to discuss it in detail in Section 3.

While not directly related to topic labelling, Chang et al. (2009) were one of the first to propose human labelling of topic models, in the form of synthetic intruder word and topic detection tasks. In the intruder word task, they include a term w with low marginal probability $p(w|t)$ for topic t into the top-N topic terms, and evaluate how well both humans and their model are able to detect the intruder.

The potential applications for automatic labelling of topics are many and varied. In document collection visualisation, e.g., the topic model can be used as the basis for generating a two-dimensional representation of the document collection (Newman et al., 2010a). Regions where documents have a high marginal probability $p(d_i|t_j)$ of being associated with a given topic can be explicitly labelled with the learned label, rather than just presented as an unlabelled region, or presented with a dense "term cloud" from the original topic. In topic model-based selectional preference learning (Ritter et al., 2010; Ò Séaghdha, 2010), the learned topics can be translated into semantic class labels (e.g. DAYS OF THE WEEK), and argument positions for individual predicates can be annotated with those labels for greater interpretability/portability. In dynamic topic models tracking the diachronic evolution of topics in time-sequenced document collections (Blei and Lafferty, 2006), labels can greatly enhance the interpretation of what topics are "trending" at any given point in time.

3 Methodology

The task of automatic labelling of topics is a natural progression from the best topic term selection task of Lau et al. (2010). In that work, the authors use a reranking framework to produce a ranking of the top-10 topic terms based on how well each term – in isolation – represents a topic. For example, in our *stock market investor fund trading ...* topic example, the term *trading* could be considered as a more representative term of the overall semantics of the topic than the top-ranked topic term *stock*.

While the best term could be used as a topic label, topics are commonly ideas or concepts that are better expressed with multiword terms (for example STOCK MARKET TRADING), or terms that might not be in the top-10 topic terms (for example, COLOURS

would be a good label for a topic of the form *red green blue cyan ...*).

In this paper, we propose a novel method for automatic topic labelling that first generates topic label candidates using English Wikipedia, and then ranks the candidates to select the best topic labels.

3.1 Candidate Generation

Given the size and diversity of English Wikipedia, we posit that the vast majority of (coherent) topics or concepts are encapsulated in a Wikipedia article. By making this assumption, the difficult task of generating potential topic labels is transposed to finding relevant Wikipedia articles, and using the title of each article as a topic label candidate.

We first use the top-10 topic terms (based on the marginal probabilities from the original topic model) to query Wikipedia, using: (a) Wikipedia's native search API; and (b) a site-restricted Google search. The combined set of top-8 article titles returned from the two search engines for each topic constitutes the initial set of *primary* candidates.

Next we chunk parse the primary candidates using the OpenNLP chunker,[2] and extract out all noun chunks. For each noun chunk, we generate all component n-grams (including the full chunk), out of which we remove all n-grams which are not in themselves article titles in English Wikipedia. For example, if the Wikipedia document title were the single noun chunk *United States Constitution*, we would generate the bigrams *United States* and *States Constitution*, and prune the latter; we would also generate the unigrams *United*, *States* and *Constitution*, all of which exist as Wikipedia articles and are preserved.

In this way, an average of 30–40 *secondary* labels are produced for each topic based on noun chunk n-grams. A good portion of these labels are commonly stopwords or unigrams that are only marginally related to the topic (an artifact of the n-gram generation process). To remove these outlier labels, we use the RACO lexical association method of Grieser et al. (2011).

RACO (Related Article Conceptual Overlap) uses Wikipedia's link structure and category membership to identify the strength of relationship between articles via their category overlap. The set of categories related to an article is defined as the union of the category membership of all outlinks in that article. The category overlap of two articles (a and b) is the intersection of the related category sets of each article. The formal definition of this measure is as follows:

$$|(\cup_{p \in O(a)} C(p)) \cap (\cup_{p \in O(b)} C(p))|$$

where $O(a)$ is the set of outlinks from article a, and $C(p)$ is the set of categories of which article p is a member. This is then normalised using Dice's coefficient to generate a similarity measure. In the instance that a term maps onto multiple Wikipedia articles via a disambiguation page, we return the best RACO score across article pairings for a given term pair. The final score for each secondary label candidate is calculated as the average RACO score with each of the primary label candidates. All secondary labels with an average RACO score of 0.1 and above are added to the label candidate set.

Finally, we add the top-5 topic terms to the set of candidates, based on the marginals from the original topic model. Doing this ensures that there are always label candidates for all topics (even if the Wikipedia searches fail), and also allows the possibility of labeling a topic using its own topic terms, which was demonstrated by Lau et al. (2010) to be a baseline source of topic label candidates.

3.2 Candidate Ranking

After obtaining the set of topic label candidates, the next step is to rank the candidates to find the best label for each topic. We will first describe the features that we use to represent label candidates.

3.2.1 Features

A good label should be strongly associated with the topic terms. To learn the association of a label candidate with the topic terms, we use several lexical association measures: pointwise mutual information (PMI), Student's t-test, Dice's coefficient, Pearson's χ^2 test, and the log likelihood ratio (Pecina, 2009). We also include conditional probability and reverse conditional probability measures, based on the work of Lau et al. (2010). To calculate the association measures, we parse the full collection of English Wikipedia articles using a sliding window of width

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

20, and obtain term frequencies for the label candidates and topic terms. To measure the association between a label candidate and a list of topic terms, we average the scores of the top-10 topic terms.

In addition to the association measures, we include two lexical properties of the candidate: the raw number of terms, and the relative number of terms in the label candidate that are top-10 topic terms.

We also include a search engine score for each label candidate, which we generate by querying a local copy of English Wikipedia with the top-10 topic terms, using the Zettair search engine (based on BM25 term similarity).[3] For a given label candidate, we return the average score for the Wikipedia article(s) associated with it.

3.2.2 Unsupervised and Supervised Ranking

Each of the proposed features can be used as the basis for an unsupervised model for label candidate selection, by ranking the label candidates for a given topic and selecting the top-N. Alternatively, they can be combined in a supervised model, by training over topics where we have gold-standard labelling of the label candidates. For the supervised method, we use a support vector regression (SVR) model (Joachims, 2006) over all of the features.

4 Datasets

We conducted topic labelling experiments using document collections constructed from four distinct domains/genres, to test the domain/genre independence of our method:

BLOGS : 120,000 blog articles dated from August to October 2008 from the Spinn3r blog dataset[4]

BOOKS : 1,000 English language books from the Internet Archive American Libraries collection

NEWS : 29,000 New York Times news articles dated from July to September 1999, from the English Gigaword corpus

PUBMED : 77,000 PubMed biomedical abstracts published in June 2010

[3] http://www.seg.rmit.edu.au/zettair/

[4] http://www.icwsm.org/data/

The BLOGS dataset contains blog posts that cover a diverse range of subjects, from product reviews to casual, conversational messages. The BOOKS topics, coming from public-domain out-of-copyright books (with publication dates spanning more than a century), relate to a wide range of topics including furniture, home decoration, religion and art, and have a more historic feel to them. The NEWS topics reflect the types and range of subjects one might expect in news articles such as health, finance, entertainment, and politics. The PUBMED topics frequently contain domain-specific terms and are sharply differentiated from the topics for the other corpora. We are particularly interested in the performance of the method over PUBMED, as it is a highly specialised domain where we may expect lower coverage of appropriate topic labels within Wikipedia.

We took a standard approach to topic modelling each of the four document collections: we tokenised, lemmatised and stopped each document,[5] and created a vocabulary of terms that occurred at least ten times. From this processed data, we created a bag-of-words representation of each document, and learned topic models with $T = 100$ topics in each case.

To focus our experiments on topics that were relatively more coherent and interpretable, we first used the method of Newman et al. (2010b) to calculate the average PMI-score for each topic, and filtered all topics that had an average PMI-score lower than 0.4. We additionally filtered any topics where less than 5 of the top-10 topic terms are default nominal in Wikipedia.[6] The filtering criteria resulted in 45 topics for BLOGS, 38 topics for BOOKS, 60 topics for NEWS, and 85 topics for PUBMED. Manual inspection of the discarded topics indicated that they were predominantly hard-to-label junk topics or mixed topics, with limited utility for document/term clustering.

Applying our label candidate generation methodology to these 228 topics produced approximately 6000 labels — an average of 27 labels per topic.

[5] OpenNLP is used for tokenization, Morpha for lemmatization (Minnen et al., 2001).

[6] As determined by POS tagging English Wikipedia with OpenNLP, and calculating the coarse-grained POS priors (noun, verb, etc.) for each term.

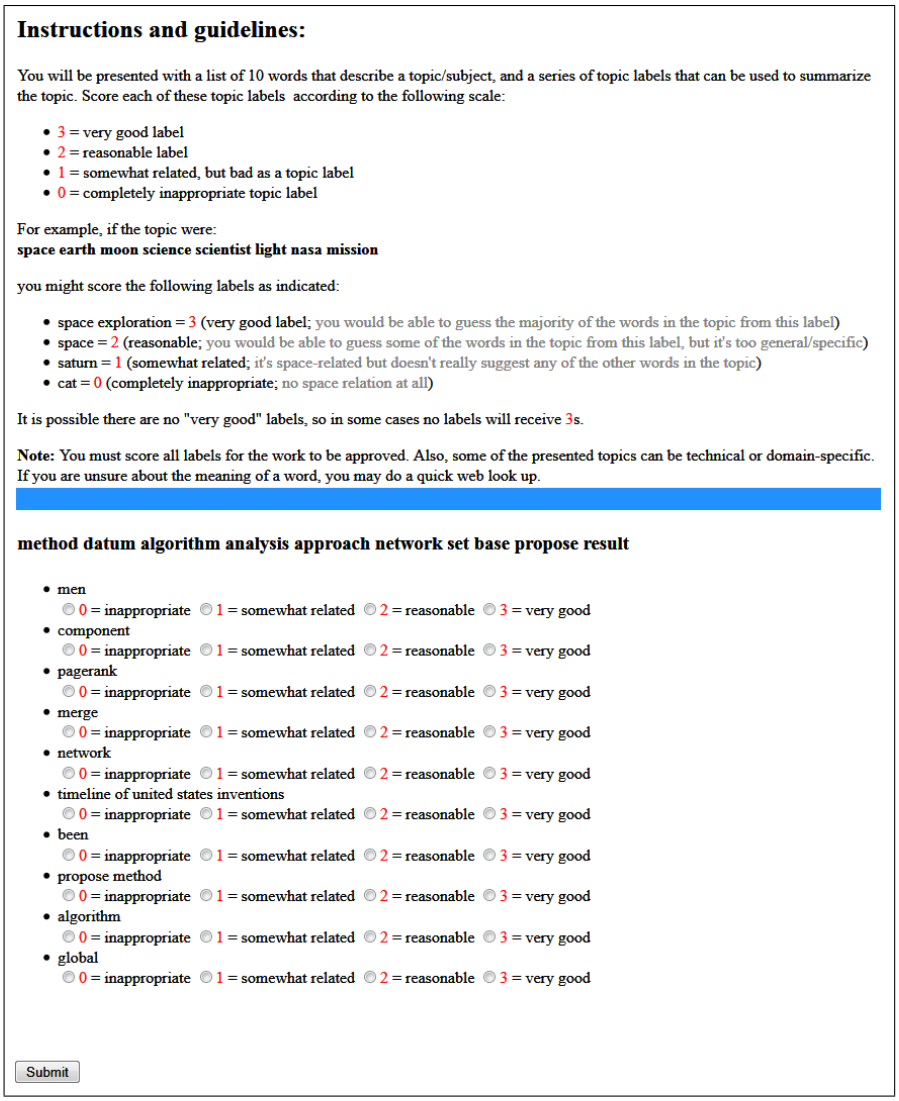


Figure 1: A screenshot of the topic label evaluation task on Amazon Mechanical Turk. This screen constitutes a *Human Intelligence Task* (HIT); it contains a topic followed by 10 suggested topic labels, which are to be rated. Note that *been* would be the stopword label in this example.

4.1 Topic Candidate Labelling

To evaluate our methods and train the supervised method, we require gold-standard ratings for the label candidates. To this end, we used Amazon Mechanical Turk to collect annotations for our labels.

In our annotation task, each topic was presented in the form of its top-10 terms, followed by 10 suggested labels for the topic. This constitutes a *Human Intelligence Task* (HIT); annotators are paid based on the number of HITs they have completed. A screenshot of a HIT seen by annotator is presented in Figure 1.

In each HIT, annotators were asked to rate the labels based on the following ordinal scale:

3: Very good label; a perfect description of the topic.

2: Reasonable label, but does not completely capture the topic.

1: Label is semantically related to the topic, but would not make a good topic label.

0: Label is completely inappropriate, and unrelated to the topic.

To filter annotations from workers who did not perform the task properly or from spammers, we ap-

Domain	Topic Terms	Label Candidate	Average Rating
BLOGS	china chinese olympics gold olympic team win beijing medal sport	2008 summer olympics	2.60
BOOKS	church arch wall building window gothic nave side vault tower	gothic architecture	2.40
NEWS	israel peace barak israeli minister palestinian agreement prime leader palestinians	israeli-palestinian conflict	2.63
PUBMED	cell response immune lymphocyte antigen cytokine t-cell induce receptor immunity	immune system	2.36

Table 1: A sample of topics and topic labels, along with the average rating for each label candidate

plied a few heuristics to automatically detect these workers. Additionally, we inserted a small number of stopwords as label candidates in each HIT and recorded workers who gave high ratings to these stopwords. Annotations from workers who failed to passed these tests are removed from the final set of gold ratings.

Each label candidate was rated in this way by at least 10 annotators, and ratings from annotators who passed the filter were combined by averaging them. A sample of topics, label candidates, and the average rating is presented in Table 1.[7]

Finally, we train the regression model over all the described features, using the human rating-based ranking.

5 Experiments

In this section we present our experimental results for the topic labelling task, based on both the unsupervised and supervised methods, and the methodology of Mei et al. (2007), which we denote MSZ for the remainder of the paper.

5.1 Evaluation

We use two basic measures to evaluate the performance of our predictions. **Top-1 average rating** is the average annotator rating given to the top-ranked system label, and has a maximum value of 3 (where annotators unanimously rated all top-ranked system labels with a 3). This is intended to give a sense of the *absolute* utility of the top-ranked candidates.

The second measure is normalized discounted cumulative gain (nDCG: Jarvelin and Kekalainen (2002), Croft et al. (2009)), computed for the top-1 (**nDCG-1**), top-3 (**nDCG-3**) and top-5 ranked system labels (**nDCG-5**). For a given ordered list of scores, this measure is based on the difference between the original order, and the order when the list is sorted by score. That is, if items are ranked optimally in descending order of score at position N, nDCG-N is equal to 1. nDCG is a normalised score, and indicates how close the candidate label ranking is to the optimal ranking within the set of annotated candidates, noting that an nDCG-N score of 1 tells us nothing about absolute values of the candidates. This second evaluation measure is thus intended to reflect the *relative* quality of the ranking, and complements the top-1 average rating.

Note that conventional precision- and recall-based evaluation is not appropriate for our task, as each label candidate has a real-valued rating.

As a baseline for the task, we use the unsupervised label candidate ranking method based on Pearson's χ^2 test, as it was overwhelmingly found to be the pick of the features for candidate ranking.

5.2 Results for the Supervised Method

For the supervised model, we present both in-domain results based on 10-fold cross-validation, and cross-domain results where we learn a model from the ratings for the topic model from a given domain, and apply it to a second domain. In each case, we learn an SVR model over the full set of features described in Section 3.2.1. In practical terms, in-domain results make the unreasonable assumption that we have labelled 90% of labels in order to be able to label the remaining 10%, and cross-domain results are thus the more interesting data point in terms of the expected results when applying our method to a novel topic model. It is valuable to compare the two, however, to gauge the relative impact of domain on the results.

We present the results for the supervised method in Table 2, including the unsupervised baseline and an upper bound estimate for comparison purposes. The upper bound is calculated by ranking the candi-

[7] The dataset is available for download from `http://www.csse.unimelb.edu.au/research/lt/resources/acl2011-topic/`.

Test Domain	Training	Top-1 Average Rating				nDCG-1	nDCG-3	nDCG-5
		All	1°	2°	Top5			
BLOGS	Baseline (unsupervised)	1.84	1.87	1.75	1.74	0.75	0.77	0.79
	In-domain	1.98	1.94	1.95	1.77	0.81	0.82	0.83
	Cross-domain: BOOKS	1.88	1.92	1.90	1.77	0.77	0.81	0.83
	Cross-domain: NEWS	1.97	1.94	1.92	1.77	0.80	0.83	0.83
	Cross-domain: PUBMED	1.95	1.95	1.93	1.82	0.80	0.82	0.83
	Upper bound	2.45	2.26	2.29	2.18	1.00	1.00	1.00
BOOKS	Baseline (unsupervised)	1.75	1.76	1.70	1.72	0.77	0.77	0.79
	In-domain	1.91	1.90	1.83	1.74	0.84	0.81	0.83
	Cross-domain: BLOGS	1.82	1.88	1.79	1.71	0.79	0.81	0.82
	Cross-domain: NEWS	1.82	1.87	1.80	1.75	0.79	0.81	0.83
	Cross-domain: PUBMED	1.87	1.87	1.80	1.73	0.81	0.82	0.83
	Upper bound	2.29	2.17	2.15	2.04	1.00	1.00	1.00
NEWS	Baseline (unsupervised)	1.96	1.76	1.87	1.70	0.80	0.79	0.78
	In-domain	2.02	1.92	1.90	1.82	0.82	0.82	0.84
	Cross-domain: BLOGS	2.03	1.92	1.89	1.85	0.83	0.82	0.84
	Cross-domain: BOOKS	2.01	1.80	1.93	1.73	0.82	0.82	0.83
	Cross-domain: PUBMED	2.01	1.93	1.94	1.80	0.82	0.82	0.83
	Upper bound	2.45	2.31	2.33	2.12	1.00	1.00	1.00
PUBMED	Baseline (unsupervised)	1.73	1.74	1.68	1.63	0.75	0.77	0.79
	In-domain	1.79	1.76	1.74	1.67	0.77	0.82	0.84
	Cross-domain: BLOGS	1.80	1.77	1.73	1.69	0.78	0.82	0.84
	Cross-domain: BOOKS	1.77	1.70	1.74	1.64	0.77	0.82	0.83
	Cross-domain: NEWS	1.79	1.76	1.73	1.65	0.77	0.82	0.84
	Upper bound	2.31	2.17	2.22	2.01	1.00	1.00	1.00

Table 2: Supervised results for all domains

dates based on the annotated human ratings. The upper bound for top-1 average rating is thus the highest average human rating of all label candidates for a given topic, while the upper bound for the nDCG measures will always be 1.

In addition to results for the combined candidate set, we include results for each of the three candidate subsets, namely the primary Wikipedia labels ("1°"), the secondary Wikipedia labels ("2°") and the top-5 topic terms ("Top5"); the nDCG results are over the full candidate set only, as the numbers aren't directly comparable over the different subsets (due to differences in the number of candidates and the distribution of ratings).

Comparing the in-domain and cross-domain results, we observe that they are largely comparable, with the exception of BOOKS, where there is a noticeable drop in both top-1 average rating and nDGC-1 when we use cross-domain training. We see an appreciable drop in scores when we train BOOKS against BLOGS (or vice versa), which we analyse as being due to incompatibility in document content and structure between these two domains. Overall though, the results are very encouraging, and point to the plausibility of using labelled topic models from independent domains to learn the best topic labels for a new domain.

Returning to the question of the suitability of label candidates for the highly specialised PUBMED document collection, we first notice that the upper bound top-1 average rating is comparable to the other domains, indicating that our method has been able to extract equivalent-quality label candidates from Wikipedia. The top-1 average ratings of the supervised method are lower than the other domains. We hypothesise that the cause of the drop is that the lexical association measures are trained over highly diverse Wikipedia data rather than biomedical-specific data, and predict that the results would improve if we trained our features over PubMed.

The results are uniformly better than the unsupervised baselines for all four corpora, although there is quite a bit of room for improvement relative to the upper bound. To better gauge the quality of these results, we carry out a direct comparison of our proposed method with the best-performing method of MSZ in Section 5.3.

Looking to the top-1 average score results over the different candidate sets, we observe first that the upper bound for the combined candidate set ("All") is higher than the scores for the candidate subsets in all cases, underlining the complementarity of the different candidate sets. We also observe that the top-5 topic term candidate set is the lowest performer out of the three subsets across all four corpora, in terms of both upper bound and the results for the supervised method. This reinforces our comments about the inferiority of the topic word selection method of Lau et al. (2010) for topic labelling purposes. For NEWS and PUBMED, there is a noticeable difference between the results of the supervised method over the full candidate set and each of the candidate subsets. In contrast, for BOOKS and BLOGS, the results for the primary candidate subset are at times actually higher than those over the full candidate set in most cases (but not for the upper bound). This is due to the larger search space in the full candidate set, and the higher median quality of candidates in the primary candidate set.

5.3 Comparison with MSZ

The best performing method out of the suite of approaches proposed by MSZ method exclusively uses bigrams extracted from the document collection, ranked based on Student's t-test. The potential drawbacks to this approach are: all labels must be bigrams, there must be explicit token instances of a given bigram in the document collection for it to be considered as a label candidate, and furthermore, there must be *enough* token instances in the document collection for it to have a high t score.

To better understand the performance difference of our approach to that of MSZ, we perform direct comparison of our proposed method with the benchmark method of MSZ.

5.3.1 Candidate Ranking

First, we compare the candidate ranking methodology of our method with that of MSZ, using the label candidates extracted by the MSZ method.

We first extracted the top-2000 bigrams using the N-gram Statistics Package (Banerjee and Pedersen, 2003). We then ranked the bigrams for each topic using the Student's t-test. We included the top-5 labels generated for each topic by the MSZ method in our Mechanical Turk annotation task, and use the annotations to directly compare the two methods.

To measure the performance of candidate ranking between our supervised method and MSZ's, we re-rank the top-5 labels extracted by MSZ using our SVR methodology (in-domain) and compare the top-1 average rating and nDCG scores. Results are shown in Table 3. We do not include results for the BOOKS domain because the text collection is much larger than the other domains, and the computation for the MSZ relevance score ranking is intractable due to the number of n-grams (a significant shortcoming of the method).

Looking at the results for the other domains, it is clear that our ranking system has the upper hand: it consistently outperforms MSZ over every evaluation metric.[8] Comparing the top-1 average rating results back to those in Table 2, we observe that for all three domains, the results for MSZ are below those of the unsupervised baseline, and well below those of our supervised method. The nDCG results are more competitive, and the nDCG-3 results are actually higher than our original results in Table 2. It is important to bear in mind, however, that the numbers are in each case relative to a different label candidate set. Additionally, the results in Table 3 are based on only 5 candidates, with a relatively flat gold-standard rating distribution, making it easier to achieve higher nDCG-5 scores.

5.3.2 Candidate Generation

The method of MSZ makes the implicit assumption that good bigram labels are discoverable within the document collection. In our method, on the other hand, we (efficiently) access the much larger and variable n-gram length set of English Wikipedia article titles, in addition to the top-5 topic terms. To better understand the differences in label candidate sets, and the relative coverage of the full label candidate set in each case, we conducted another survey where human users were asked to suggest one topic label for each topic presented.

The survey consisted, once again, of presenting annotators with a topic, but in this case, we gave them the open task of proposing the ideal label for

[8] Based on a single ANOVA, the difference in results is statistically significant at the 5% level for BLOGS, and 1% for NEWS and PUBMED.

Test Domain	Candidate Ranking System	Top-1 Avg. Rating	nDCG-1	nDCG-3	nDCG-5
BLOGS	MSZ	1.26	0.65	0.76	0.87
	SVR	1.41	0.75	0.85	0.92
	Upper bound	1.87	1.00	1.00	1.00
NEWS	MSZ	1.37	0.73	0.81	0.90
	SVR	1.66	0.88	0.90	0.95
	Upper bound	1.86	1.00	1.00	1.00
PUBMED	MSZ	1.53	0.77	0.85	0.93
	SVR	1.73	0.87	0.91	0.96
	Upper bound	1.98	1.00	1.00	1.00

Table 3: Comparison of results for our proposed supervised ranking method (SVR) and that of MSZ

the topic. In this, we did not enforce any restrictions on the type or size of label (e.g. the number of terms in the label).

Of the manually-generated gold-standard labels, approximately 36% were contained in the original document collection, but 60% were Wikipedia article titles. This indicates that our method has greater potential to generate a label of the quality of the ideal proposed by a human in a completely open-ended task.

6 Discussion

On the subject of suitability of using Amazon Mechanical Turk for natural language tasks, Snow et al. (2008) demonstrated that the quality of annotation is comparable to that of expert annotators. With that said, the PUBMED topics are still a subject of interest, as these topics often contain biomedical terms which could be difficult for the general populace to annotate.

As the number of annotators per topic and the number of annotations per annotator vary, there is no immediate way to calculate the inter-annotator agreement. Instead, we calculated the MAE score for each candidate, which is an average of the absolute difference between an annotator's rating and the average rating of a candidate, summed across all candidates to get the MAE score for a given corpus. The MAE scores for each corpus are shown in Table 4, noting that a smaller value indicates higher agreement.

As the table shows, the agreement for the PUBMED domain is comparable with the other datasets. BLOGS and NEWS have marginally better agreement, almost certainly because of the greater immediacy of the topics, covering everyday areas such as lifestyle and politics. BOOKS topics are occasionally difficult to label due to the breadth of the domain; e.g. consider a topic containing terms extracted from Shakespeare sonnets.

Corpus	MAE
BLOGS	0.50
BOOKS	0.56
NEWS	0.52
PUBMED	0.56

Table 4: Average MAE score for label candidate rating over each corpus

7 Conclusion

This paper has presented the task of topic labelling, that is the generation and scoring of labels for a given topic. We generate a set of label candidates from the top-ranking topic terms, titles of Wikipedia articles containing the top-ranking topic terms, and also a filtered set of sub-phrases extracted from the Wikipedia article titles. We rank the label candidates using a combination of association measures, lexical features and an Information Retrieval feature. Our method is shown to perform strongly over four independent sets of topics, and also significantly better than a competitor system.

Acknowledgements

NICTA is funded by the Australian government as represented by Department of Broadband, Communication and Digital Economy, and the Australian Research Council through the ICT centre of Excellence programme. DN has also been supported by a grant from the Institute of Museum and Library Services, and a Google Research Award.

References

S. Banerjee and T. Pedersen. 2003. The design, implementation, and use of the Ngram Statistic Package. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 370–381, Mexico City, February.

D.M. Blei and J.D. Lafferty. 2006. Dynamic topic models. In *ICML 2006*.

D.M. Blei, A.Y. Ng, and M.I. Jordan. 2003. Latent Dirichlet allocation. *JMLR*, 3:993–1022.

S. Brody and M. Lapata. 2009. Bayesian word sense induction. In *EACL 2009*, pages 103–111.

J. Chang, J. Boyd-Graber, S. Gerrish, C. Wang, and D. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *NIPS*, pages 288–296.

B. Croft, D. Metzler, and T. Strohman. 2009. *Search Engines: Information Retrieval in Practice*. Addison Wesley.

Y. Feng and M. Lapata. 2010. Topic models for image annotation and text illustration. In *Proceedings of Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2010)*, pages 831–839, Los Angeles, USA, June.

K. Grieser, T. Baldwin, F. Bohnert, and L. Sonenberg. 2011. Using ontological and document similarity to estimate museum exhibit relatedness. *ACM Journal on Computing and Cultural Heritage*, 3(3):1–20.

T. Griffiths and M. Steyvers. 2004. Finding scientific topics. In *PNAS*, volume 101, pages 5228–5235.

A. Haghighi and L. Vanderwende. 2009. Exploring content models for multi-document summarization. In *HLT: NAACL 2009*, pages 362–370.

K. Jarvelin and J. Kekalainen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4).

T. Joachims. 2006. Training linear svms in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 217–226, New York, NY, USA. ACM.

J.H. Lau, D. Newman, S. Karimi, and T. Baldwin. 2010. Best topic word selection for topic labelling. In *Coling 2010: Posters*, pages 605–613, Beijing, China.

D. Magatti, S. Calegari, D. Ciucci, and F. Stella. 2009. Automatic labeling of topics. In *ISDA 2009*, pages 1227–1232, Pisa, Italy.

Q. Mei, C. Liu, H. Su, and C. Zhai. 2006. A probabilistic approach to spatiotemporal theme pattern mining on weblogs. In *WWW 2006*, pages 533–542.

Q. Mei, X. Shen, and C. Zhai. 2007. Automatic labeling of multinomial topic models. In *SIGKDD*, pages 490–499.

G. Minnen, J. Carroll, and D. Pearce. 2001. Applied morphological processing of English. *Journal of Natural Language Processing*, 7(3):207–223.

D. Newman, T. Baldwin, L. Cavedon, S. Karimi, D. Martinez, and J. Zobel. 2010a. Visualizing document collections and search results using topic mapping. *Journal of Web Semantics*, 8(2-3):169–175.

D. Newman, J.H. Lau, K. Grieser, and T. Baldwin. 2010b. Automatic evaluation of topic coherence. In *Proceedings of Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2010)*, pages 100–108, Los Angeles, USA, June. Association for Computational Linguistics.

D. Ò Séaghdha. 2010. Latent variable models of selectional preference. In *ACL 2010*.

P. Pantel and D. Ravichandran. 2004. Automatically labeling semantic classes. In *HLT/NAACL-04*, pages 321–328.

P. Pecina. 2009. *Lexical Association Measures: Collocation Extraction*. Ph.D. thesis, Charles University.

A. Ritter, Mausam, and O. Etzioni. 2010. A latent Dirichlet allocation method for selectional preferences. In *ACL 2010*.

R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Morristown, NJ, USA.

I. Titov and R. McDonald. 2008. Modeling online reviews with multi-grain topic models. In *WWW '08*, pages 111–120.

X. Wang and A. McCallum. 2006. Topics over time: A non-Markov continuous-time model of topical trends. In *KDD*, pages 424–433.

S. Wei and W.B. Croft. 2006. LDA-based document models for ad-hoc retrieval. In *SIGIR '06*, pages 178–185.

Using Bilingual Information for Cross-Language Document Summarization

Xiaojun Wan
Institute of Compute Science and Technology, Peking University, Beijing 100871, China
Key Laboratory of Computational Linguistics (Peking University), MOE, China
wanxiaojun@icst.pku.edu.cn

Abstract

Cross-language document summarization is defined as the task of producing a summary in a target language (e.g. Chinese) for a set of documents in a source language (e.g. English). Existing methods for addressing this task make use of either the information from the original documents in the source language or the information from the translated documents in the target language. In this study, we propose to use the bilingual information from both the source and translated documents for this task. Two summarization methods (SimFusion and CoRank) are proposed to leverage the bilingual information in the graph-based ranking framework for cross-language summary extraction. Experimental results on the DUC2001 dataset with manually translated reference Chinese summaries show the effectiveness of the proposed methods.

1 Introduction

Cross-language document summarization is defined as the task of producing a summary in a different target language for a set of documents in a source language (Wan et al., 2010). In this study, we focus on English-to-Chinese cross-language summarization, which aims to produce Chinese summaries for English document sets. The task is very useful in the field of multilingual information access. For example, it is beneficial for most Chinese readers to quickly browse and understand English news documents or document sets by reading the corresponding Chinese summaries.

A few pilot studies have investigated the task in recent years and exiting methods make use of either the information in the source language or the information in the target language after using machine translation. In particular, for the task of English-to-Chinese cross-language summarization, one method is to directly extract English summary sentences based on English features extracted from the English documents, and then automatically translate the English summary sentences into Chinese summary sentences. The other method is to automatically translate the English sentences into Chinese sentences, and then directly extract Chinese summary sentences based on Chinese features. The two methods make use of the information from only one language side.

However, it is not very reliable to use only the information in one language, because the machine translation quality is far from satisfactory, and thus the translated Chinese sentences usually contain some errors and noises. For example, the English sentence "*Many destroyed power lines are thought to be uninsured, as are trees and shrubs uprooted across a wide area.*" is automatically translated into the Chinese sentence "*许多破坏电源线被认为是保险的，因为是连根拔起的树木和灌木，在广泛的领域。*" by using Google Translate[1], but the Chinese sentence contains a few translation errors. Therefore, on the one side, if we rely only on the English-side information to extract Chinese

[1] http://translate.google.com/. Note that the translation service is updated frequently and the current translation results may be different from that presented in this paper.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1546–1555,
Portland, Oregon, June 19-24, 2011.

summary sentences, we cannot guarantee that the automatically translated Chinese sentences for salient English sentences are really salient when these sentences may contain many translation errors and other noises. On the other side, if we rely only on the Chinese-side information to extract Chinese summary sentences, we cannot guarantee that the selected sentences are really salient because the features for sentence ranking based on the incorrectly translated sentences are not very reliable, either.

In this study, we propose to leverage both the information in the source language and the information in the target language for cross-language document summarization. In particular, we propose two graph-based summarization methods (SimFusion and CoRank) for using both English-side and Chinese-side information in the task of English-to-Chinese cross-document summarization. The SimFusion method linearly fuses the English-side similarity and the Chinese-side similarity for measuring Chinese sentence similarity. The CoRank method adopts a co-ranking algorithm to simultaneously rank both English sentences and Chinese sentences by incorporating mutual influences between them.

We use the DUC2001 dataset with manually translated reference Chinese summaries for evaluation. Experimental results based on the ROUGE metrics show the effectiveness of the proposed methods. Three important conclusions for this task are summarized below:

1) The Chinese-side information is more beneficial than the English-side information.
2) The Chinese-side information and the English-side information can complement each other.
3) The proposed CoRank method is more reliable and robust than the proposed SimFusion method.

The rest of this paper is organized as follows: Section 2 introduces related work. In Section 3, we present our proposed methods. Evaluation results are shown in Section 4. Lastly, we conclude this paper in Section 5.

2 Related Work

2.1 General Document Summarization

Document summarization methods can be extraction-based, abstraction-based or hybrid methods. We focus on extraction-based methods in this study, and the methods directly extract summary sentences from a document or document set by ranking the sentences in the document or document set.

In the task of single document summarization, various features have been investigated for ranking sentences in a document, including term frequency, sentence position, cue words, stigma words, and topic signature (Luhn 1969; Lin and Hovy, 2000). Machine learning techniques have been used for sentence ranking (Kupiec et al., 1995; Amini and Gallinari, 2002). Litvak et al. (2010) present a language-independent approach for extractive summarization based on the linear optimization of several sentence ranking measures using a genetic algorithm. In recent years, graph-based methods have been proposed for sentence ranking (Erkan and Radev, 2004; Mihalcea and Tarau, 2004). Other methods include mutual reinforcement principle (Zha 2002; Wan et al., 2007).

In the task of multi-document summarization, the centroid-based method (Radev et al., 2004) ranks the sentences in a document set based on such features as cluster centroids, position and TFIDF. Machine Learning techniques have also been used for feature combining (Wong et al., 2008). Nenkova and Louis (2008) investigate the influences of input difficulty on summarization performance. Pitler et al. (2010) present a systematic assessment of several diverse classes of metrics designed for automatic evaluation of linguistic quality of multi-document summaries. Celikyilmaz and Hakkani-Tur (2010) formulate extractive summarization as a two-step learning problem by building a generative model for pattern discovery and a regression model for inference. Aker et al. (2010) propose an A* search algorithm to find the best extractive summary up to a given length, and they propose a discriminative training algorithm for directly maximizing the quality of the best summary. Graph-based methods have also been used to rank sentences for multi-document summarization (Mihalcea and Tarau, 2005; Wan and Yang, 2008).

2.2 Cross-Lingual Document Summarization

Several pilot studies have investigated the task of cross-language document summarization. The existing methods use only the information in either language side. Two typical translation schemes are document translation or summary translation. The document translation scheme first translates the source documents into the corresponding documents in the target language, and then extracts summary sentences based only on the information on the target side. The summary translation scheme first extracts summary sentences from the source documents based only on the information on the source side, and then translates the summary sentences into the corresponding summary sentences in the target language.

For example Leuski et al. (2003) use machine translation for English headline generation for Hindi documents. Lim et al. (2004) propose to generate a Japanese summary by using Korean summarizer. Chalendar et al. (2005) focus on semantic analysis and sentence generation techniques for cross-language summarization. Orasan and Chiorean (2008) propose to produce summaries with the MMR method from Romanian news articles and then automatically translate the summaries into English. Cross language query based summarization has been investigated in (Pingali et al., 2007), where the query and the documents are in different languages. Wan et al. (2010) adopt the summary translation scheme for the task of English-to-Chinese cross-language summarization. They first extract English summary sentences by using English-side features and the machine translation quality factor, and then automatically translate the English summary into Chinese summary. Other related work includes multilingual summarization (Lin et al., 2005; Siddharthan and McKeown, 2005), which aims to create summaries from multiple sources in multiple languages.

3 Our Proposed Methods

As mentioned in Section 1, existing methods rely only on one-side information for sentence ranking, which is not very reliable. In order to leveraging both-side information for sentence ranking, we propose the following two methods to incorporate the bilingual information in different ways.

3.1 SimFusion

This method uses the English-side information for Chinese sentence ranking in the graph-based framework. The sentence similarities in the two languages are fused in the method. In other words, when we compute the similarity value between two Chinese sentences, the similarity value between the corresponding two English sentences is used by linear fusion. Since sentence similarity evaluation plays a very important role in the graph-based ranking algorithm, this method can leverage both-side information through similarity fusion.

Formally, given the Chinese document set D^{cn} translated from an English document set, let $G^{cn}=(V^{cn}, E^{cn})$ be an undirected graph to reflect the relationships between the sentences in the Chinese document set. V^{cn} is the set of vertices and each vertex s^{cn}_i in V^{cn} represents a Chinese sentence. E^{cn} is the set of edges. Each edge e^{cn}_{ij} in E^{cn} is associated with an affinity weight $f(s^{cn}_i, s^{cn}_j)$ between sentences s^{cn}_i and s^{cn}_j $(i \neq j)$. The weight is computed by linearly combining the similarity value $sim_{cosine}(s^{cn}_i, s^{cn}_j)$ between the Chinese sentences and the similarity value $sim_{cosine}(s^{en}_i, s^{en}_j)$ between the corresponding English sentences.

$$f(s_i^{cn}, s_j^{cn}) = \lambda \cdot sim_{\cos ine}(s_i^{cn}, s_j^{cn}) + (1-\lambda) \cdot sim_{\cos ine}(s_i^{en}, s_j^{en})$$

where s^{en}_j and s^{en}_i are the source English sentences for s^{cn}_j and s^{cn}_i. $\lambda \in [0, 1]$ is a parameter to control the relative contributions of the two similarity values. The similarity values $sim_{cosine}(s^{cn}_i, s^{cn}_j)$ and $sim_{cosine}(s^{en}_i, s^{en}_j)$ are computed by using the standard cosine measure. The weight for each term is computed based on the TFIDF formula. For Chinese similarity computation, Chinese word segmentation is performed. Here, we have $f(s^{cn}_i, s^{cn}_j)=f(s^{cn}_j, s^{cn}_i)$ and let $f(s^{cn}_i, s^{cn}_i)=0$ to avoid self transition. We use an affinity matrix M^{cn} to describe G^{cn} with each entry corresponding to the weight of an edge in the graph. $M^{cn}=(M^{cn}_{ij})_{|Vcn|\times|Vcn|}$ is defined as $M^{cn}_{ij}=f(s^{cn}_i,s^{cn}_j)$. Then M^{cn} is normalized to $\tilde{M}^{cn}$ to make the sum of each row equal to 1.

Based on matrix $\tilde{M}^{cn}$, the saliency score $InfoScore(s^{cn}_i)$ for sentence s^{cn}_i can be deduced from those of all other sentences linked with it and it can be formulated in a recursive form as in the PageRank algorithm:

$$InfoScore(s_i^{cn}) = \mu \cdot \sum_{all\ j \neq i} InfoScore(s_j^{cn}) \cdot \widetilde{M}_{ji}^{cn} + \frac{(1-\mu)}{n}$$

where n is the sentence number, i.e. $n= |V^{cn}|$. μ is the damping factor usually set to 0.85, as in the PageRank algorithm.

For numerical computation of the saliency scores, we can iteratively run the above equation until convergence.

For multi-document summarization, some sentences are highly overlapping with each other, and thus we apply the same greedy algorithm in Wan et al. (2006) to penalize the sentences highly overlapping with other highly scored sentences, and finally the salient and novel Chinese sentences are directly selected as summary sentences.

3.2 CoRank

This method leverages both the English-side information and the Chinese-side information in a co-ranking way. The source English sentences and the translated Chinese sentences are simultaneously ranked in a unified graph-based algorithm. The saliency of each English sentence relies not only on the English sentences linked with it, but also on the Chinese sentences linked with it. Similarly, the saliency of each Chinese sentence relies not only on the Chinese sentences linked with it, but also on the English sentences linked with it. More specifically, the proposed method is based on the following assumptions:

Assumption 1: A Chinese sentence would be salient if it is heavily linked with other salient Chinese sentences; and an English sentence would be salient if it is heavily linked with other salient English sentences.

Assumption 2: A Chinese sentence would be salient if it is heavily linked with salient English sentences; and an English sentence would be salient if it is heavily linked with salient Chinese sentences.

The first assumption is similar to PageRank which makes use of mutual "recommendations" between the sentences in the same language to rank sentences. The second assumption is similar to HITS if the English sentences and the Chinese sentences are considered as authorities and hubs, respectively. In other words, the proposed method aims to fuse the ideas of PageRank and HITS in a unified framework. The mutual influences between the Chinese sentences and the English sentences are incorporated in the method.

Figure 1 gives the graph representation for the method. Three kinds of relationships are exploited: the CN-CN relationships between Chinese sentences, the EN-EN relationships between English sentences, and the EN-CN relationships between English sentences and Chinese sentences.

Formally, given an English document set D^{en} and the translated Chinese document set D^{cn}, let $G=(V^{en}, V^{cn}, E^{en}, E^{cn}, E^{encn})$ be an undirected graph to reflect all the three kinds of relationships between the sentences in the two document sets. $V^{en} =\{s^{en}_i \mid 1\le i\le n\}$ is the set of English sentences. $V^{cn}=\{s^{cn}_i \mid 1\le i\le n\}$ is the set of Chinese sentences. s^{cn}_i is the corresponding Chinese sentence translated from s^{en}_i. n is the number of the sentences. E^{en} is the edge set to reflect the relationships between the English sentences. E^{cn} is the edge set to reflect the relationships between the Chinese sentences. E^{encn} is the edge set to reflect the relationships between the English sentences and the Chinese sentences. Based on the graph representation, we compute the following three affinity matrices to reflect the three kinds of sentence relationships:

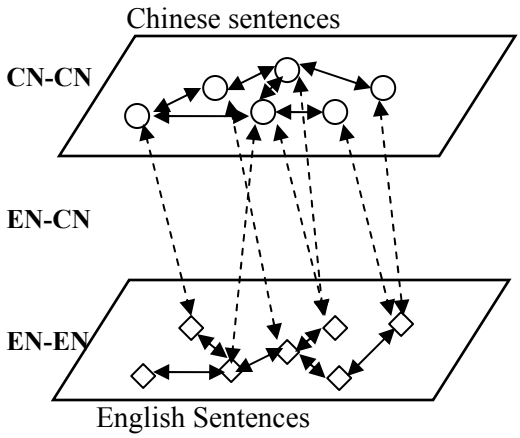


Figure 1. The three kinds of sentence relationships

1) $\boldsymbol{M}^{cn}=(M^{cn}_{ij})_{n\times n}$: This affinity matrix aims to reflect the relationships between the Chinese sentences. Each entry in the matrix corresponds to the cosine similarity between the two Chinese sentences.

$$M_{ij}^{cn} = \begin{cases} sim_{\cos ine}(s_i^{cn}, s_j^{cn}), & if\ i \neq j \\ 0, & otherwise \end{cases}$$

Then $\boldsymbol{M}^{cn}$ is normalized to $\tilde{\boldsymbol{M}}^{\boldsymbol{cn}}$ to make the sum of each row equal to 1.

2) $\boldsymbol{M}^{en}=(M^{en}{}_{i,j})_{n\times n}$: This affinity matrix aims to reflect the relationships between the English sentences. Each entry in the matrix corresponds to the cosine similarity between the two English sentences.

$$M_{ij}^{en}=\begin{cases} sim_{\cos ine}(s_i^{en},s_j^{en}), & if\ i\neq j \\ 0, & otherwise \end{cases}$$

Then $\boldsymbol{M}^{en}$ is normalized to $\tilde{\boldsymbol{M}}^{\boldsymbol{en}}$ to make the sum of each row equal to 1.

3) $\boldsymbol{M}^{encn}=(M^{encn}{}_{ij})_{n\times n}$: This affinity matrix aims to reflect the relationships between the English sentences and the Chinese sentences. Each entry $M^{encn}{}_{ij}$ in the matrix corresponds to the similarity between the English sentence $s^{en}{}_i$ and the Chinese sentence $s^{cn}{}_j$. It is hard to directly compute the similarity between the sentences in different languages. In this study, the similarity value is computed by fusing the following two similarity values: the cosine similarity between the sentence $s^{en}{}_i$ and the corresponding source English sentence $s^{en}{}_j$ for $s^{cn}{}_j$, and the cosine similarity between the corresponding translated Chinese sentence $s^{cn}{}_i$ for $s^{en}{}_i$ and the sentence $s^{cn}{}_j$. We use the geometric mean of the two values as the affinity weight.

$$M_{ij}^{encn}=\sqrt{sim_{\cos ine}(s_i^{en},s_j^{en})\times sim_{\cos ine}(s_i^{cn},s_j^{cn})}$$

Note that we have $M^{encn}{}_{ij}=M^{encn}{}_{ji}$ and $\boldsymbol{M}^{encn}=(\boldsymbol{M}^{encn})^{\mathrm{T}}$. Then $\boldsymbol{M}^{encn}$ is normalized to $\tilde{\boldsymbol{M}}^{\boldsymbol{encn}}$ to make the sum of each row equal to 1.

We use two column vectors $\boldsymbol{u}=[u(s^{cn}{}_i)]_{n\times 1}$ and $\boldsymbol{v}=[v(s^{en}{}_j)]_{n\times 1}$ to denote the saliency scores of the Chinese sentences and the English sentences, respectively. Based on the three kinds of relationships, we can get the following four assumptions:

$$u(s_i^{cn})\propto\sum\nolimits_j \tilde{M}_{ji}^{cn}u(s_j^{cn})$$

$$v(s_j^{en})\propto\sum\nolimits_i \tilde{M}_{ij}^{en}v(s_i^{en})$$

$$u(s_i^{cn})\propto\sum\nolimits_j \tilde{M}_{ji}^{encn}v(s_j^{en})$$

$$v(s_j^{en})\propto\sum\nolimits_i \tilde{M}_{ij}^{encn}u(s_i^{cn})$$

After fusing the above equations, we can obtain the following iterative forms:

$$u(s_i^{cn})=\alpha\sum\nolimits_j \tilde{M}_{ji}^{cn}u(s_j^{cn})+\beta\sum\nolimits_j \tilde{M}_{ji}^{encn}v(s_j^{en})$$

$$v(s_j^{en})=\alpha\sum\nolimits_i \tilde{M}_{ij}^{en}v(s_i^{en})+\beta\sum\nolimits_i \tilde{M}_{ij}^{encn}u(s_i^{cn})$$

And the matrix form is:

$$\boldsymbol{u}=\alpha(\tilde{\boldsymbol{M}}^{\boldsymbol{cn}})^T\boldsymbol{u}+\beta(\tilde{\boldsymbol{M}}^{encn})^T\boldsymbol{v}$$

$$\boldsymbol{v}=\alpha(\tilde{\boldsymbol{M}}^{\boldsymbol{en}})^T\boldsymbol{v}+\beta(\tilde{\boldsymbol{M}}^{encn})^T\boldsymbol{u}$$

where α and β specify the relative contributions to the final saliency scores from the information in the same language and the information in the other language and we have $\alpha+\beta=1$.

For numerical computation of the saliency scores, we can iteratively run the two equations until convergence. Usually the convergence of the iteration algorithm is achieved when the difference between the scores computed at two successive iterations for any sentences and words falls below a given threshold. In order to guarantee the convergence of the iterative form, $\boldsymbol{u}$ and $\boldsymbol{v}$ are normalized after each iteration.

After we get the saliency scores $\boldsymbol{u}$ for the Chinese sentences, we apply the same greedy algorithm for redundancy removing. Finally, a few highly ranked sentences are selected as summary sentences.

4 Experimental Evaluation

4.1 Evaluation Setup

There is no benchmark dataset for English-to-Chinese cross-language document summarization, so we built our evaluation dataset based on the DUC2001 dataset by manually translating the reference summaries.

DUC2001 provided 30 English document sets for generic multi-document summarization. The average document number per document set was 10. The sentences in each article have been separated and the sentence information has been stored into files. Three or two generic reference English summaries were provided by NIST annotators for each document set. Three graduate students were employed to manually translate the reference English summaries into reference Chinese summaries. Each student manually translated one third of the reference summaries. It was much easier and more reliable to provide the reference Chinese summaries by manual translation than by manual summarization.

	ROUGE-2 Average_F	ROUGE-W Average_F	ROUGE-L Average_F	ROUGE-SU4 Average_F
Baseline(EN)	0.03723	0.05566	0.13259	0.07177
Baseline(CN)	0.03805	0.05886	0.13871	0.07474
SimFusion	0.04017	0.06117	0.14362	0.07645
CoRank	0.04282	0.06158	0.14521	0.07805

Table 1: Comparison Results

All the English sentences in the document set were automatically translated into Chinese sentences by using *Google Translate*, and the Stanford Chinese Word Segmenter[2] was used for segmenting the Chinese documents and summaries into words. For comparative study, the summary length was limited to five sentences, i.e. each Chinese summary consisted of five sentences.

We used the ROUGE-1.5.5 (Lin and Hovy, 2003) toolkit for evaluation, which has been widely adopted by DUC and TAC for automatic summarization evaluation. It measured summary quality by counting overlapping units such as the n-gram, word sequences and word pairs between the candidate summary and the reference summary. We showed three of the ROUGE F-measure scores in the experimental results: ROUGE-2 (bigram-based), ROUGE-W (based on weighted longest common subsequence, weight=1.2), ROUGE-L (based on longest common subsequences), and ROUGE-SU4 (based on skip bigram with a maximum skip distance of 4). Note that the ROUGE toolkit was performed for Chinese summaries after using word segmentation.

Two graph-based baselines were used for comparison.

Baseline(EN): This baseline adopts the summary translation scheme, and it relies on the English-side information for English sentence ranking. The extracted English summary is finally automatically translated into the corresponding Chinese summary. The same sentence ranking algorithm with the SimFusion method is adopted, and the affinity weight is computed based only on the cosine similarity between English sentences.

Baseline(CN): This baseline adopts the document translation scheme, and it relies on the Chinese-side information for Chinese sentence ranking. The Chinese summary sentences are directly extracted from the translated Chinese documents. The same sentence ranking algorithm with the SimFusion method is adopted, and the affinity weight is computed based only on the cosine similarity between Chinese sentences.

For our proposed methods, the parameter values are empirically set as λ=0.8 and α=0.5.

4.2 Results and Discussion

Table 1 shows the comparison results for our proposed methods and the baseline methods. Seen from the tables, Baseline(CN) performs better than Baseline(EN) over all the metrics. The results demonstrate that the Chinese-side information is more beneficial than the English-side information for cross-document summarization, because the summary sentences are finally selected from the Chinese side. Moreover, our proposed two methods can outperform the two baselines over all the metrics. The results demonstrate the effectiveness of using bilingual information for cross-language document summarization. It is noteworthy that the ROUGE scores in the table are not high due to the following two reasons: 1) The use of machine translation may introduce many errors and noises in the peer Chinese summaries; 2) The use of Chinese word segmentation may introduce more noises and mismatches in the ROUGE evaluation based on Chinese words.

We can also see that the CoRank method can outperform the SimFusion method over all metrics. The results show that the CoRank method is more suitable for the task by incorporating the bilingual information into a unified ranking framework.

In order to show the influence of the value of the combination parameter λ on the performance of the SimFusion method, we present the performance curves over the four metrics in Figures 2 through 5, respectively. In the figures, λ ranges from 0 to 1, and λ=1 means that SimFusion is the same with Baseline(CN), and λ=0 means that only English-side information is used for Chinese sentence ranking. We can see that when λ is set to a value larger than 0.5, SimFusion can outperform the two baselines over most metrics. The results show that λ can be set in a relatively wide range. Note that

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

λ>0.5 means that SimFusion relies more on the Chinese-side information than on the English-side information. Therefore, the Chinese-side information is more beneficial than the English-side information.

In order to show the influence of the value of the combination parameter α on the performance of the CoRank method, we present the performance curves over the four metrics in Figures 6 through 9, respectively. In the figures, α ranges from 0.1 to 0.9, and a larger value means that the information from the same language side is more relied on, and a smaller value means that the information from the other language side is more relied on. We can see that CoRank can always outperform the two baselines over all metrics with different value of α. The results show that α can be set in a very wide range. We also note that a very large value or a very small value of α can lower the performance values. The results demonstrate that CoRank relies on both the information from the same language side and the information from the other language side for sentence ranking. Therefore, both the Chinese-side information and the English-side information can complement each other, and they are beneficial to the final summarization performance.

Comparing Figures 2 through 5 with Figures 6 through 9, we can further see that the CoRank method is more stable and robust than the SimFusion method. The CoRank method can outperform the SimFusion method with most parameter settings. The bilingual information can be better incorporated in the unified ranking framework of the CoRank method.

Finally, we show one running example for the document set D59 in the DUC2001 dataset. The four summaries produced by the four methods are listed below:

Baseline(EN): *周日的崩溃是 24 年来第一次乘客在涉及西北飞机事故中丧生。有乘客和观察员的报告，这架飞机的右翼引擎也坠毁前失败。在坠机现场联邦航空局官员表示不会揣测关于崩溃或在飞机上的发动机评论的原因。美国联邦航空局的记录显示，除了那些涉及的飞机坠毁，与 JT8D 涡轮路段-200 系列发动机问题的三个共和国在过去四年的航班发生的事件。1988 年 7 月，一个联合国的 DC-10 坠毁的苏城，艾奥瓦州后，发动机在飞行中发生外，造成 112 人。*

Baseline(CN): *第二，在美国历史上最严重的事故是 1987 年 8 月 16 日，坠毁，造成 156 人时，美国西北航空公司飞机上的底特律都市机场起飞时坠毁。据美国联邦航空管理局的纪录，麦道公司的 MD-82 飞机在 1985 年和 1986 年紧急降落后，在其两个引擎之一是失去权力。周日的崩溃是 24 年来第一次乘客在涉及西北飞机事故中丧生。今年 4 月，国家运输安全委员会敦促美国联邦航空局后进行一些危险，发动机故障，飞机的一个发动机的 200 系列 JT8D 安全调查。目前，机组人员发现了一个黑人师谁说，他可以引导飞机在附近的人们听到了他们的区域。*

SimFusion: *第二，在美国历史上最严重的事故是 1987 年 8 月 16 日，坠毁，造成 156 人时，美国西北航空公司飞机上的底特律都市机场起飞时坠毁。周日的崩溃是 24 年来第一次乘客在涉及西北飞机事故中丧生。在坠机现场联邦航空局官员表示不会揣测关于崩溃或在飞机上的发动机评论的原因。有乘客和观察员的报告，这架飞机的右翼引擎也坠毁前失败。据美国联邦航空管理局的纪录，麦道公司的 MD-82 飞机在 1985 年和 1986 年紧急降落后，在其两个引擎之一是失去权力。*

CoRank：*周日的崩溃是 24 年来第一次乘客在涉及西北飞机事故中丧生。第二，在美国历史上最严重的事故是 1987 年 8 月 16 日，坠毁，造成 156 人时，美国西北航空公司飞机上的底特律都市机场起飞时坠毁。在坠机现场联邦航空局官员表示不会揣测关于崩溃或在飞机上的发动机评论的原因。最严重的航空事故不断，在美国是一个在芝加哥的美国航空公司客机 1979 年崩溃。有乘客和观察员的报告，这架飞机的右翼引擎也坠毁前失败。*

5 Conclusion and Future Work

In this paper, we propose two methods (SimFusion and CoRank) to address the cross-language document summarization task by leveraging the bilingual information in both the source and target language sides. Evaluation results demonstrate the effectiveness of the proposed methods. The Chinese-side information is validated to be more beneficial than the English-side information, and the CoRank method is more robust than the SimFusion method.

In future work, we will investigate to use the machine translation quality factor to further improve the fluency of the Chinese summary, as in Wan et al. (2010). Though our attempt to use GIZA++ for evaluating the similarity between Chinese sentences and English sentences failed, we will exploit more advanced measures based on statistical alignment model for cross-language similarity computation.

Acknowledgments

This work was supported by NSFC (60873155), Beijing Nova Program (2008B03) and NCET (NCET-08-0006). We thank the three students for translating the reference summaries. We also thank the anonymous reviewers for their useful comments.

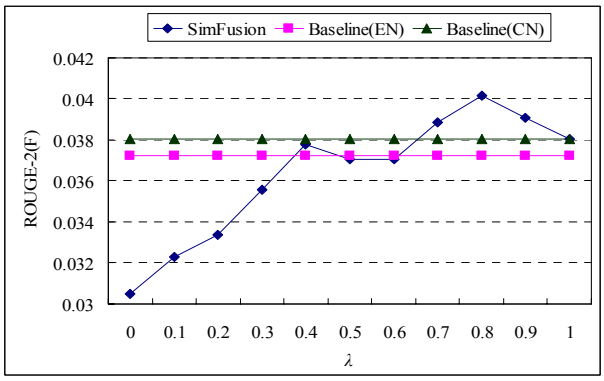


Figure 2. ROUGE-2(F) vs. λ for SimFusion

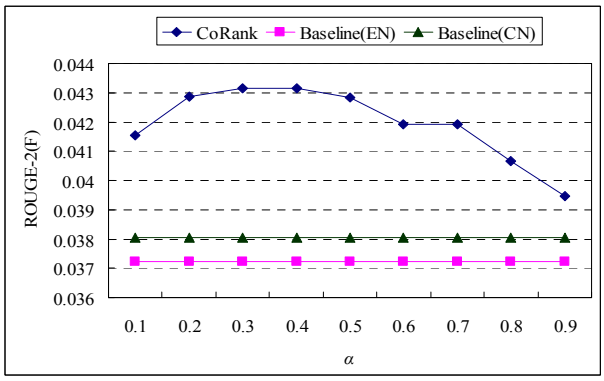


Figure 6. ROUGE-2(F) vs. α for CoRank

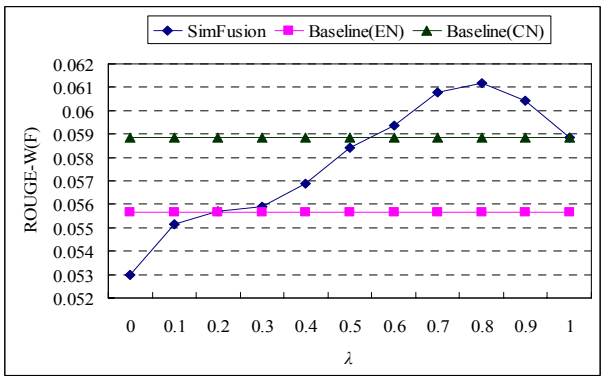


Figure 3. ROUGE-W(F) vs. λ for SimFusion

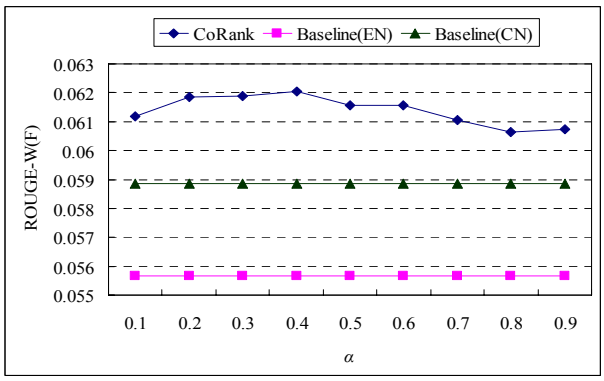


Figure 7. ROUGE-W(F) vs. α for CoRank

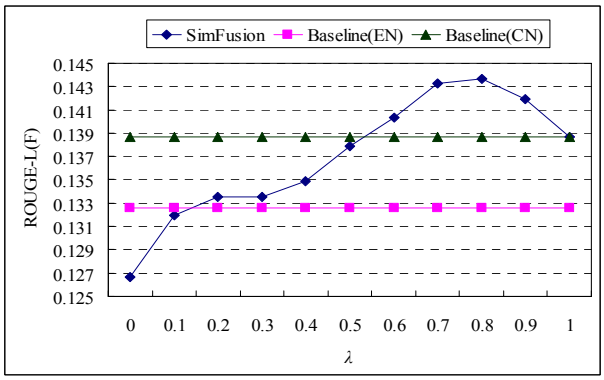


Figure 4. ROUGE-L(F) vs. λ for SimFusion

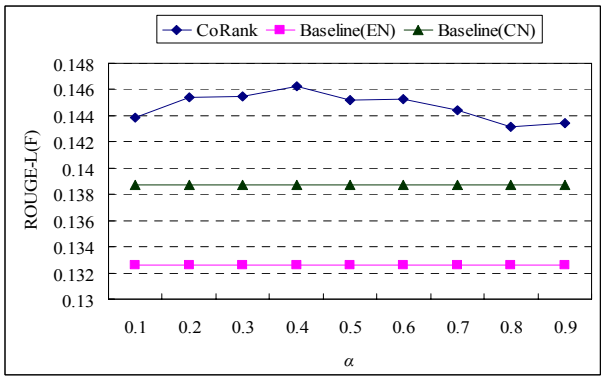


Figure 8. ROUGE-L(F) vs. α for CoRank

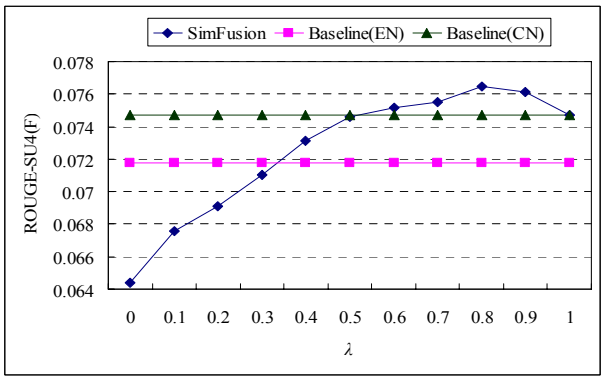


Figure 5. ROUGE-SU4(F) vs. λ for SimFusion

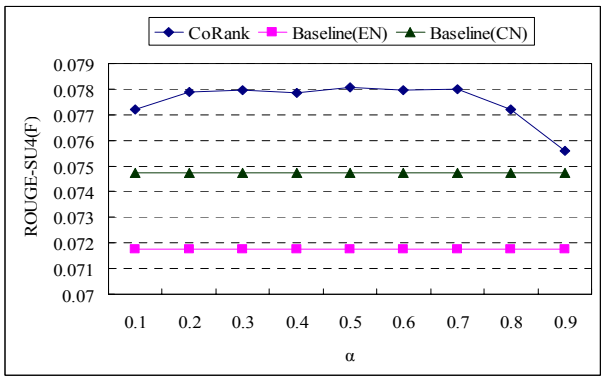


Figure 9. ROUGE-SU4(F) vs. α for CoRank

References

A. Aker, T. Cohn, and R. Gaizauskas. 2010. Multi-document summarization using A* search and discriminative training. In *Proceedings of EMNLP2010*.

M. R. Amini, P. Gallinari. 2002. The Use of Unlabeled Data to Improve Supervised Learning for Text Summarization. In *Proceedings of SIGIR2002*.

G. de Chalendar, R. Besançon, O. Ferret, G. Grefenstette, and O. Mesnard. 2005. Crosslingual summarization with thematic extraction, syntactic sentence simplification, and bilingual generation. In *Workshop on Crossing Barriers in Text Summarization Research, 5th International Conference on Recent Advances in Natural Language Processing (RANLP2005)*.

A. Celikyilmaz and D. Hakkani-Tur. 2010. A hybrid hierarchical model for multi-document summarization. In *Proceedings of ACL2010*.

G. ErKan, D. R. Radev. LexPageRank. 2004. Prestige in Multi-Document Text Summarization. In *Proceedings of EMNLP2004*.

D. Klein and C. D. Manning. 2002. Fast Exact Inference with a Factored Model for Natural Language Parsing. In *Proceedings of NIPS2002*.

J. Kupiec, J. Pedersen, F. Chen. 1995. A.Trainable Document Summarizer. In *Proceedings of SIGIR1995*.

A. Leuski, C.-Y. Lin, L. Zhou, U. Germann, F. J. Och, E. Hovy. 2003. Cross-lingual C*ST*RD: English access to Hindi information. *ACM Transactions on Asian Language Information Processing*, 2(3): 245-269.

J.-M. Lim, I.-S. Kang, J.-H. Lee. 2004. Multi-document summarization using cross-language texts. In *Proceedings of NTCIR-4*.

C. Y. Lin, E. Hovy. 2000. The Automated Acquisition of Topic Signatures for Text Summarization. In *Proceedings of the 17th Conference on Computational Linguistics*.

C.-Y. Lin and E.H. Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In *Proceedings of HLT-NAACL -03*.

C.-Y. Lin, L. Zhou, and E. Hovy. 2005. Multilingual summarization evaluation 2005: automatic evaluation report. In *Proceedings of MSE (ACL-2005 Workshop)*.

M. Litvak, M. Last, and M. Friedman. 2010. A new approach to improving multilingual summarization using a genetic algorithm. In *Proceedings of ACL2010*.

H. P. Luhn. 1969. The Automatic Creation of literature Abstracts. *IBM Journal of Research and Development*, 2(2).

R. Mihalcea, P. Tarau. 2004. TextRank: Bringing Order into Texts. In *Proceedings of EMNLP2004*.

R. Mihalcea and P. Tarau. 2005. A language independent algorithm for single and multiple document summarization. In *Proceedings of IJCNLP-05*.

A. Nenkova and A. Louis. 2008. Can you summarize this? Identifying correlates of input difficulty for generic multi-document summarization. In *Proceedings of ACL-08:HLT*.

A. Nenkova, R. Passonneau, and K. McKeown. 2007. The Pyramid method: incorporating human content selection variation in summarization evaluation. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(2).

C. Orasan, and O. A. Chiorean. 2008. Evaluation of a Crosslingual Romanian-English Multi-document Summariser. In *Proceedings of 6th Language Resources and Evaluation Conference (LREC2008)*.

P. Pingali, J. Jagarlamudi and V. Varma. 2007. Experiments in cross language query focused multi-document summarization. In *Workshop on Cross Lingual Information Access Addressing the Information Need of Multilingual Societies in IJCAI2007*.

E. Pitler, A. Louis, and A. Nenkova. 2010. Automatic evaluation of linguistic quality in multi-document summarization. In *Proceedings of ACL2010*.

D. R. Radev, H. Y. Jing, M. Stys and D. Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40: 919-938.

A. Siddharthan and K. McKeown. 2005. Improving multilingual summarization: using redundancy in the input to correct MT errors. In *Proceedings of HLT/EMNLP-2005*.

X. Wan, H. Li and J. Xiao. 2010. Cross-language document summarization based on machine translation quality prediction. In *Proceedings of ACL2010*.

X. Wan, J. Yang and J. Xiao. 2006. Using cross-document random walks for topic-focused multi-documetn summarization. In *Proceedings of WI2006*.

X. Wan and J. Yang. 2008. Multi-document summarization using cluster-based link analysis. In *Proceedings of SIGIR-08*.

X. Wan, J. Yang and J. Xiao. 2007. Towards an Iterative Reinforcement Approach for Simultaneous Document Summarization and Keyword Extraction. In *Proceedings of ACL2007*.

K.-F. Wong, M. Wu and W. Li. 2008. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of COLING-08*.

H. Y. Zha. 2002. Generic Summarization and Keyphrase Extraction Using Mutual Reinforcement Principle and Sentence Clustering. In *Proceedings of SIGIR2002*.

Exploiting Web-Derived Selectional Preference to Improve Statistical Dependency Parsing

Guangyou Zhou, Jun Zhao*, Kang Liu, and Li Cai
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
95 Zhongguancun East Road, Beijing 100190, China
{gyzhou,jzhao,kliu,lcai}@nlpr.ia.ac.cn

Abstract

In this paper, we present a novel approach which incorporates the web-derived selectional preferences to improve statistical dependency parsing. Conventional selectional preference learning methods have usually focused on word-to-class relations, e.g., a verb selects as its subject a given nominal class. This paper extends previous work to word-to-word selectional preferences by using web-scale data. Experiments show that web-scale data improves statistical dependency parsing, particularly for long dependency relationships. There is no data like more data, performance improves log-linearly with the number of parameters (unique N-grams). More importantly, when operating on new domains, we show that using web-derived selectional preferences is essential for achieving robust performance.

1 Introduction

Dependency parsing is the task of building dependency links between words in a sentence, which has recently gained a wide interest in the natural language processing community. With the availability of large-scale annotated corpora such as Penn Treebank (Marcus et al., 1993), it is easy to train a high-performance dependency parser using supervised learning methods.

However, current state-of-the-art statistical dependency parsers (McDonald et al., 2005; McDonald and Pereira, 2006; Hall et al., 2006) tend to have lower accuracies for longer dependencies (McDonald and Nivre, 2007). The length of a dependency from word w_i to word w_j is simply equal to $|i-j|$. Longer dependencies typically represent the modifier of the root or the main verb, internal dependencies of longer NPs or PP-attachment in a sentence. Figure 1 shows the F_1 score[1] relative to the dependency length on the development set by using the graph-based dependency parsers (McDonald et al., 2005; McDonald and Pereira, 2006). We note that the parsers provide very good results for adjacent dependencies (96.89% for dependency length =1), while the dependency length increases, the accuracies degrade sharply. These longer dependencies are therefore a major opportunity to improve the overall performance of dependency parsing. Usually, these longer dependencies can be parsed dependent on the specific words involved due to the limited range of features (e.g., a verb and its modifiers). Lexical statistics are therefore needed for resolving ambiguous relationships, yet the lexicalized statistics are sparse and difficult to estimate directly. To solve this problem, some information with different granularity has been investigated. Koo et al. (2008) proposed a semi-supervised dependency parsing by introducing lexical intermediaries at a coarser level than words themselves via a cluster method. This approach, however, ignores the selectional preference for word-to-word interactions, such as head-modifier relationship. Extra resources

*Correspondence author: jzhao@nlpr.ia.ac.cn

[1] Precision represents the percentage of predicted arcs of length d that are correct, and recall measures the percentage of gold-standard arcs of length d that are correctly predicted. $F_1 = 2 \times precision \times recall/(precision + recall)$

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1556–1565,
Portland, Oregon, June 19-24, 2011.

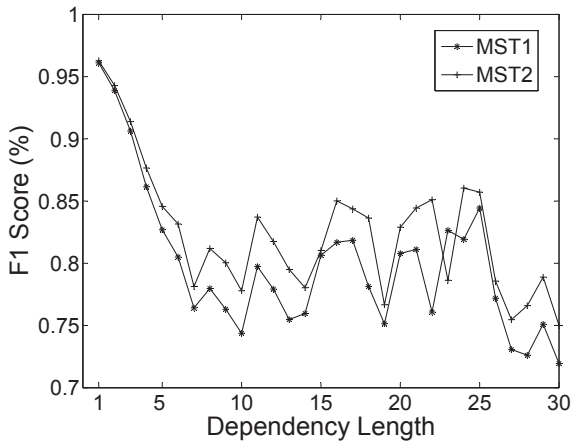


Figure 1: F score relative to dependency length.

beyond the annotated corpora are needed to capture the bi-lexical relationship at the word-to-word level.

Our purpose in this paper is to exploit web-derived selectional preferences to improve the supervised statistical dependency parsing. All of our lexical statistics are derived from two kinds of web-scale corpus: one is the web, which is the largest data set that is available for NLP (Keller and Lapata, 2003). Another is a web-scale N-gram corpus, which is a N-gram corpus with N-grams of length 1-5 (Brants and Franz, 2006), we call it **Google V1** in this paper. The idea is very simple: web-scale data have large coverage for word pair acquisition. By leveraging some assistant data, the dependency parsing model can directly utilize the additional information to capture the word-to-word level relationships. We address two natural and related questions which some previous studies leave open:

Question I: Is there a benefit in incorporating web-derived selectional preference features for statistical dependency parsing, especially for longer dependencies?

Question II: How well do web-derived selectional preferences perform on new domains?

For Question I, we systematically assess the value of using web-scale data in state-of-the-art supervised dependency parsers. We compare dependency parsers that include or exclude selectional preference features obtained from web-scale corpus. To the best of our knowledge, none of the existing studies directly address long dependencies of dependency parsing by using web-scale data.

Most statistical parsers are highly domain dependent. For example, the parsers trained on WSJ text perform poorly on Brown corpus. Some studies have investigated domain adaptation for parsers (McClosky et al., 2006; Daumé III, 2007; McClosky et al., 2010). These approaches assume that the parsers know which domain it is used, and that it has access to representative data in that domain. However, in practice, these assumptions are unrealistic in many real applications, such as when processing the heterogeneous genre of web texts. In this paper we incorporate the web-derived selectional preference features to design our parsers for robust open-domain testing.

We conduct the experiments on the English Penn Treebank (PTB) (Marcus et al., 1993). The results show that web-derived selectional preference can improve the statistical dependency parsing, particularly for long dependency relationships. More importantly, when operating on new domains, the web-derived selectional preference features show great potential for achieving robust performance (Section 4.3).

The remainder of this paper is divided as follows. Section 2 gives a brief introduction of dependency parsing. Section 3 describes the web-derived selectional preference features. Experimental evaluation and results are reported in Section 4. Finally, we discuss related work and draw conclusion in Section 5 and Section 6, respectively.

2 Dependency Parsing

In dependency parsing, we attempt to build head-modifier (or head-dependent) relations between words in a sentence. The discriminative parser we used in this paper is based on the *part-factored* model and features of the MSTParser (McDonald et al., 2005; McDonald and Pereira, 2006; Carreras, 2007). The parsing model can be defined as a conditional distribution $p(y|\mathbf{x};\mathbf{w})$ over each projective parse tree y for a particular sentence $\mathbf{x}$, parameterized by a vector $\mathbf{w}$. The probability of a parse tree is

$$p(y|\mathbf{x};\mathbf{w}) = \frac{1}{Z(\mathbf{x};\mathbf{w})}\exp\Big\{\sum_{\rho\in y}\mathbf{w}\cdot\mathbf{\Phi}(\mathbf{x},\rho)\Big\} \quad (1)$$

where $Z(\mathbf{x};\mathbf{w})$ is the partition function and $\mathbf{\Phi}$ are *part-factored* feature functions that include *head-*

modifier parts, *sibling* parts and *grandchild* parts. Given the training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, parameter estimation for log-linear models generally resolve around optimization of a regularized conditional log-likelihood objective $\mathbf{w}^* = \arg\min_{\mathbf{w}} L(\mathbf{w})$ where

$$L(\mathbf{w}) = -C \sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i; \mathbf{w}) + \frac{1}{2}||\mathbf{w}||^2 \quad (2)$$

The parameter $C > 0$ is a constant dictating the level of regularization in the model. Since objective function $L(\mathbf{w})$ is smooth and convex, which is convenient for standard gradient-based optimization techniques. In this paper we use the dual exponentiated gradient (EG)[2] descent, which is a particularly effective optimization algorithm for log-linear models (Collins et al., 2008).

3 Web-Derived Selectional Preference Features

In this paper, we employ two different feature sets: a baseline feature set[3] which draw upon "normal" information source, such as word forms and part-of-speech (POS) without including the web-derived selectional preference[4] features, a feature set conjoins the baseline features and the web-derived selectional preference features.

3.1 Web-scale resources

All of our selectional preference features described in this paper rely on probabilities derived from unlabeled data. To use the largest amount of data possible, we exploit web-scale resources. one is web, N-gram counts are approximated by **Google hits**. Another we use is **Google V1** (Brants and Franz, 2006). This N-gram corpus records how often each unique sequence of words occurs. N-grams appearing 40 times or more (1 in 25 billion) are kept, and appear in the n-gram tables. All n-grams with lower counts are discarded. Co-occurrence probabilities can be calculated directly from the N-gram counts.

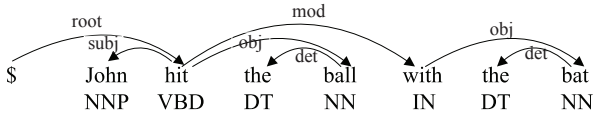


Figure 2: An example of a labeled dependency tree. The tree contains a special token "$" which is always the root of the tree. Each arc is directed from head to modifier and has a label describing the function of the attachment.

3.2 Web-derived N-gram features

3.2.1 PMI

Previous work on noun compounds bracketing has used *adjacency model* (Resnik, 1993) and *dependency model* (Lauer, 1995) to compute association statistics between pairs of words. In this paper we generalize the adjacency and dependency models by including the pointwise mutual information (Church and Hanks, 1900) between all pairs of words in the dependency tree:

$$\text{PMI}(x, y) = \log \frac{p(\text{“x y”})}{p(\text{“x”})p(\text{“y”})} \quad (3)$$

where $p(\text{“x y”})$ is the co-occurrence probabilities. When use the **Google V1** corpus, this probabilities can be calculated directly from the N-gram counts, while using the **Google hits**, we send the queries to the search engine *Google*[5] and all the search queries are performed as exact matches by using quotation marks.[6]

The value of these features is the PMI, if it is defined. If the PMI is undefined, following the work of (Pitler et al., 2010), we include one of two binary features:

$$p(\text{“x y”}) = 0 \text{ or } p(\text{“x”}) \vee p(\text{“y”}) = 0$$

Besides, we also consider the trigram features be-

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

[3] This kind of feature sets are similar to other feature sets in the literature (McDonald et al., 2005; Carreras, 2007), so we will not attempt to give a exhaustive description.

[4] Selectional preference tells us which arguments are plausible for a particular predicate, one way to determine the selectional preference is from co-occurrences of predicates and arguments in text (Bergsma et al., 2008). In this paper, the selectional preferences have the same meaning with N-grams, which model the word-to-word relationships, rather than only considering the predicates and arguments relationships.

[5] http://www.google.com/

[6] Google only allows automated querying through the Google Web API, this involves obtaining a license key, which then restricts the number of queries to a daily quota of 1000. However, we obtained a quota of 20,000 queries per day by sending a request to api-support@google.com for research purposes.

PMI("hit with")
x_i-word="hit", x_j-word="with", PMI("hit with")
x_i-word="hit", x_j-word="with", x_j-pos="IN", PMI("hit with")
x_i-word="hit", x_i-pos="VBD", x_j-word="with", PMI("hit with")
x_i-word="hit", b-pos="ball", x_j-word="with", PMI("hit with")
x_i-word="hit", x_j-word="with", PMI("hit with"), dir=R, dist=3
...

Table 1: An example of the N-gram PMI features and the conjoin features with the baseline.

tween the three words in the dependency tree:

$$\text{PMI}(x, y, z) = \log \frac{p(\text{"x y z"})}{p(\text{"x y"})p(\text{"y z"})} \quad (4)$$

This kinds of trigram features, for example in MSTParser, which can directly capture the sibling and grandchild features.

We illustrate the PMI features with an example of dependency parsing tree in Figure 2. In deciding the dependency between the main verb *hit* and its argument headed preposition *with*, an example of the N-gram PMI features and the conjoin features with the baseline are shown in Table 1.

3.2.2 PP-attachment

Propositional phrase (PP) attachment is one of the hardest problems in English dependency parsing. An English sentence consisting of a subject, a verb, and a nominal object followed by a prepositional phrase is often ambiguous. Ambiguity resolution reflects the selectional preference between the verb and noun with their prepositional phrase. For example, considering the following two examples:

(1) John **hit** the ball **with** the *bat*.

(2) John hit the **ball with** the red *stripe*.

In sentence (1), the preposition **with** depends on the main verb **hit**; but in sentence (2), the prepositional phrase is a noun attribute and the preposition **with** needs to depends on the word **ball**. To resolve this kind of ambiguity, there needs to measure the attachment preference. We thus have PP-attachment features that determine the PMI association across the preposition word "*IN*"[7]:

$$\text{PMI}_{IN}(x, z) = \log \frac{p(\text{"x } IN \text{ z"})}{p(\text{x})} \quad (5)$$

[7]Here, the preposition word "*IN*" (e.g., "with", "in", ...) is any token whose part-of-speech is IN

N-gram feature templates
hw, mw, PMI(hw,mw)
hw, ht, mw, PMI(hw,mw)
hw, mw, mt, PMI(hw,mw)
hw, ht, mw, mt, PMI(hw,mw)
...
hw, mw, sw
hw, mw, sw, PMI(hw, mw, sw)
hw, mw, gw
hw, mw, gw, PMI(hw, mw, gw)

Table 2: Examples of N-gram feature templates. Each entry represents a class of indicator for tuples of information. For example, "hw, mw" reprsents a class of indicator features with one feature for each possible combination of head word and modifier word. Abbreviations: hw=head word, ht= head POS. st, gt=likewise for sibling and grandchild.

$$\text{PMI}_{IN}(y, z) = \log \frac{p(\text{"y } IN \text{ z"})}{p(\text{y})} \quad (6)$$

where the word x and y are usually verb and noun, z is a noun which directly depends on the preposition word "*IN*". For example in sentence (1), we would include the features $\text{PMI}_{with}(hit, bat)$ and $\text{PMI}_{with}(ball, bat)$. If both PMI features exist and $\text{PMI}_{with}(hit, bat) > \text{PMI}_{with}(ball, bat)$, indicating to our dependency parsing model that the preposition word *with* depends on the verb *hit* is a good choice. While in sentence (2), the features include $\text{PMI}_{with}(hit, stripe)$ and $\text{PMI}_{with}(ball, stripe)$.

3.3 N-gram feature templates

We generate N-gram features by mimicking the template structure of the original baseline features. For example, the baseline feature set includes indicators for word-to-word and tag-to-tag interactions between the head and modifier of a dependency. In the N-gram feature set, we correspondingly introduce N-gram PMI for word-to-word interactions.

The N-gram feature set for MSTParser is shown in Table 2. Following McDonald et al. (2005), all features are conjoined with the direction of attachment as well as the distance between the two words creating the dependency. In between N-gram features, we include the form of word trigrams and PMI of the trigrams. The surrounding word N-gram features represent the local context of the selectional preference. Besides, we also present the second-order feature templates, including the sibling and grandchild features. These features are designed to disambiguate cases like coordinating conjunctions and prepositional attachment. Consider the examples we have shown in section 3.2.2, for sentence (1), the dependency graph path feature $ball \rightarrow with \rightarrow bat$ should have a lower weight since *ball* rarely is modified by *bat*, but is often seen through them (e.g., a higher weight should be associated with $hit \rightarrow with \rightarrow bat$). In contrast, for sentence (2), our N-gram features will tell us that the prepositional phrase is much more likely to attach to the noun since the dependency graph path feature $ball \rightarrow with \rightarrow stripe$ should have a high weight due to the high strength of selectional preference between *ball* and *stripe*.

Web-derived selectional preference features based on PMI values are trickier to incorporate into the dependency parsing model because they are continuous rather than discrete. Since all the baseline features used in the literature (McDonald et al., 2005; Carreras, 2007) take on binary values of 0 or 1, there is a "mis-match" between the continuous and binary features. Log-linear dependency parsing model is sensitive to inappropriately scaled feature. To solve this problem, we transform the PMI values into a more amenable form by replacing the PMI values with their $z\text{-}score$. The $z\text{-}score$ of a PMI value x is $\frac{x-\mu}{\sigma}$, where μ and σ are the mean and standard deviation of the PMI distribution, respectively.

4 Experiments

In order to evaluate the effectiveness of our proposed approach, we conducted dependency parsing experiments in English. The experiments were performed on the Penn Treebank (PTB) (Marcus et al., 1993), using a standard set of head-selection rules (Yamada and Matsumoto, 2003) to convert the phrase structure syntax of the Treebank into a dependency tree representation, dependency labels were obtained via the "Malt" hard-coded setting.[8] We split the Treebank into a training set (Sections 2-21), a development set (Section 22), and several test sets (Sections 0,[9] 1, 23, and 24). The part-of-speech tags for the development and test set were automatically assigned by the MXPOST tagger[10], where the tagger was trained on the entire training corpus.

Web page hits for word pairs and trigrams are obtained using a simple heuristic query to the search engine *Google*.[11] Inflected queries are performed by expanding a bigram or trigram into all its morphological forms. These forms are then submitted as literal queries, and the resulting hits are summed up. John Carroll's suite of morphological tools[12] is used to generate inflected forms of verbs and nouns. All the search terms are performed as exact matches by using quotation marks and submitted to the search engines in lower case.

We measured the performance of the parsers using the following metrics: unlabeled attachment score (UAS), labeled attachment score (LAS) and complete match (CM), which were defined by Hall et al. (2006). All the metrics are calculated as mean scores per word, and punctuation tokens are consistently excluded.

4.1 Main results

There are some clear trends in the results of Table 3. First, performance increases with the order of the parser: *edge-factored* model (dep1) has the lowest performance, adding sibling and grandchild relationships (dep2) significantly increases performance. Similar observations regarding the effect of model order have also been made by Carreras (2007) and Koo et al. (2008).

Second, note that the parsers incorporating the N-gram feature sets consistently outperform the models using the baseline features in all test data sets, regardless of model order or label usage. Another

[8] http://w3.msi.vxu.se/ nivre/research/MaltXML.html

[9] We removed a single 249-word sentence from Section 0 for computational reasons.

[10] http://www.inf.ed.ac.uk/resources/nlp/local_doc/MXPOST.html

[11] http://www.google.com/

[12] http://www.cogs.susx.ac.uk/lab/nlp/carroll/morph.html.

Sec	dep1	+hits	+V1	dep2	+hits	+V1	dep1-L	+hits-L	+V1-L	dep2-L	+hits-L	+V1-L
00	90.39	**90.94**	90.91	91.56	**92.16**	**92.16**	90.11	**90.69**	90.67	91.94	**92.47**	92.42
01	91.01	**91.60**	**91.60**	92.27	**92.89**	92.86	90.77	**91.39**	**91.39**	91.81	**92.38**	92.37
23	90.82	**91.46**	91.39	91.98	**92.64**	92.59	90.30	**90.98**	90.92	91.24	**91.83**	91.77
24	89.53	**90.15**	90.13	90.81	**91.44**	91.41	89.42	**90.03**	90.02	90.30	**90.91**	90.89

Table 3: Unlabeled accuracies (UAS) and labeled accuracies (LAS) on Section 0, 1, 23, 24. Abbreviation: dep1/dep2=first-order parser and second-order parser with the baseline features; +hits=N-gram features derived from the Google hits; +V1=N-gram features derived from the Google V1; suffix-L=labeled parser. Unlabeled parsers are scored using unlabeled parent predictions, and labeled parsers are scored using labeled parent predictions.

finding is that the N-gram features derived from Google hits are slightly better than Google V1 due to the large N-gram coverage, we will discuss later. As a final note, all the comparisons between the integration of N-gram features and the baseline features in Table 3 are mildly significant using the Z-test of Collins et al. (2005) ($p < 0.08$).

Type	Systems	UAS	CM
D	Yamada and Matsumoto (2003)	90.3	38.7
	McDonald et al. (2005)	90.9	37.5
	McDonald and Pereira (2006)	91.5	42.1
	Corston-Oliver et al. (2006)	90.9	37.5
	Hall et al. (2006)	89.4	36.4
	Wang et al. (2007)	89.2	34.4
	Carreras et al. (2008)	**93.5**	-
	GoldBerg and Elhadad (2010)†	91.32	40.41
	Ours	*92.64*	46.61
C	Nivre and McDonald (2008)†	92.12	44.37
	Martins et al. (2008)†	**92.87**	45.51
	Zhang and Clark (2008)	92.1	45.4
S	Koo et al. (2008)	93.16	-
	Suzuki et al. (2009)	**93.79**	-
	Chen et al. (2009)	93.16	47.15

Table 4: Comparison of our final results with other best-performing systems on the whole Section 23. Type D, C and S denote discriminative, combined and semi-supervised systems, respectively. † These papers were not directly reported the results on this data set, we implemented the experiments in this paper.

To put our results in perspective, we also compare them with other best-performing systems in Table 4. To facilitate comparisons with previous work, we only use Section 23 as the test data. The results show that our second order model incorporating the N-gram features (92.64) performs better than most previously reported discriminative systems trained on the Treebank. Carreras et al. (2008) reported a very high accuracy using information of constituent structure of TAG grammar formalism, while in our system, we do not use such knowledge. When compared to the combined systems, our system is better than Nivre and McDonald (2008) and Zhang and Clark (2008), but a slightly worse than Martins et al. (2008). We also compare our method with the semi-supervised approaches, the semi-supervised approaches achieved very high accuracies by leveraging on large unlabeled data directly into the systems for joint learning and decoding, while in our method, we only explore the N-gram features to further improve supervised dependency parsing performance.

Table 5 shows the details of some other N-gram sources, where **NEWS**: created from a large set of news articles including the Reuters and Gigword (Graff, 2003) corpora. For a given number of unique N-gram, using any of these sources does not have significant difference in Figure 3. Google hits is the largest N-gram data and shows the best performance. The other two are smaller ones, accuracies increase linearly with the log of the number of types in the auxiliary data set. Similar observations have been made by Pitler et al. (2010). We see that the relationship between accuracy and the number of N-gram is not monotonic for Google V1. The reason may be that Google V1 does not make detailed preprocessing, containing many mistakes in the corpus. Although Google hits is noisier, it has very much larger coverage of bigrams or trigrams.

Some previous studies also found a log-linear relationship between unlabeled data (Suzuki and Isozaki, 2008; Suzuki et al., 2009; Bergsma et al., 2010; Pitler et al., 2010). We have shown that this trend continues well for dependency parsing by using web-scale data (NEWS and Google V1).

[13]Google indexes about more than 8 billion pages and each contains about 1,000 words on average.

Corpus	# of tokens	θ	# of types
NEWS	3.2B	1	3.7B
Google V1	1,024.9B	40	3.4B
Google hits[13]	8,000B	100	-

Table 5: N-gram data, with total number of words in the original corpus (in billions, B). Following (Brants and Franz, 2006; Pitler et al., 2010), we set the frequency threshold to filter the data θ, and total number of unique N-gram (types) remaining in the data.

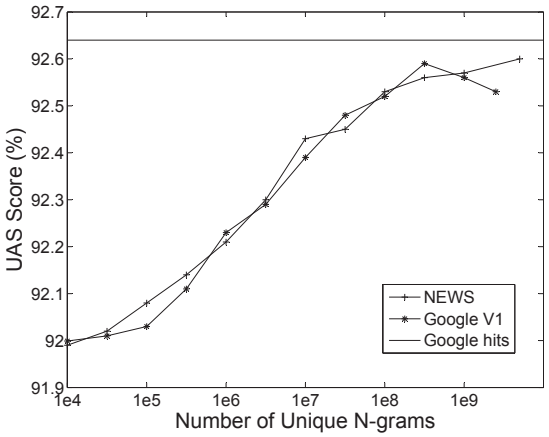


Figure 3: There is no data like more data. UAS accuracy improves with the number of unique N-grams but still lower than the Google hits.

4.2 Improvement relative to dependency length

The experiments in (McDonald and Nivre, 2007) showed a negative impact on the dependency parsing performance from too long dependencies. For our proposed approach, the improvement relative to dependency length is shown in Figure 4. From the Figure, it is seen that our method gives observable better performance when dependency lengths are larger than 3. The results here show that the proposed approach improves the dependency parsing performance, particularly for long dependency relationships.

4.3 Cross-genre testing

In this section, we present the experiments to validate the robustness the web-derived selectional preferences. The intent is to understand how well the web-derived selectional preferences transfer to other sources.

The English experiment evaluates the performance of our proposed approach when it is trained

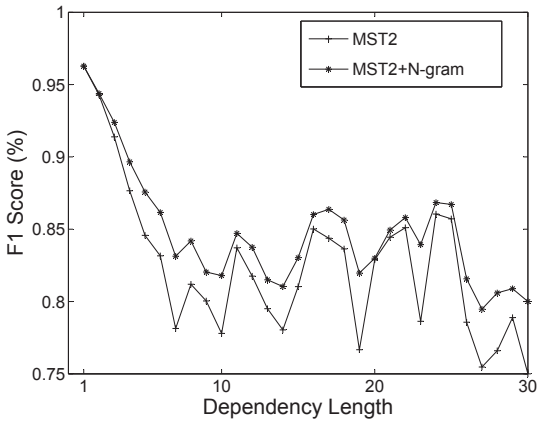


Figure 4: Dependency length vs. F_1 score.

on annotated data from one genre of text (WSJ) and is used to parse a test set from a different genre: the biomedical domain related to cancer (PennBioIE., 2005) with 2,600 parsed sentences. We divided the data into 500 for training, 100 for development and others for testing. We created five sets of training data with 100, 200, 300, 400, and 500 sentences respectively. Figure 5 plots the UAS accuracy as function of training instances. *WSJ* is the performance of our second-order dependency parser trained on section 2-21; WSJ+N-gram is the performance of our proposed approach trained on section 2-21; *WSJ+BioMed* is the performance of the parser trained on WSJ and biomedical data. *WSJ+BioMed+N-gram* is the performance of our proposed approach trained on WSJ and biomedical data. The results show that incorporating the web-scale N-gram features can significantly improve the dependency parsing performance, and the improvement is much larger than the in-domain testing presented in Section 4.1, the reason may be that web-derived N-gram features do not depend directly on training data and thus work better on new domains.

4.4 Discussion

In this paper, we present a novel method to improve dependency parsing by using web-scale data. Despite the success, there are still some problems which should be discussed.

(1) Google hits is less sparse than Google V1 in modeling the word-to-word relationships, but Google hits are likely to be noisier than Google V1. It is very appealing to carry out a correlation anal-

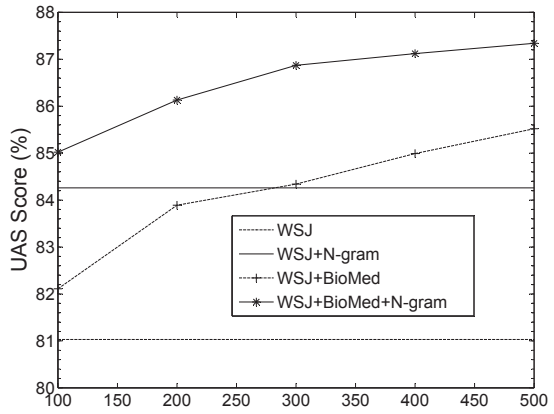


Figure 5: Adapting a WSJ parser to biomedical text. *WSJ*: performance of parser trained only on WSJ; *WSJ+N-gram*: performance of our proposed approach trained only on WSJ; *WSJ+BioMed*: parser trained on WSJ and biomedical text; *WSJ+BioMed+N-gram*: our approach trained on WSJ and biomedical text.

ysis to determine whether Google hits and Google V1 are highly correlated. We will leave it for future research.

(2) Veronis (2005) pointed out that there had been a debate about reliability of Google hits due to the inconsistencies of page hits estimates. However, this estimate is scale-invariant. Assume that when the number of pages indexed by Google grows, the number of pages containing a given search term goes to a fixed fraction. This means that if pages indexed by Google doubles, then so do the bigrams or trigrams frequencies. Therefore, the estimate becomes stable when the number of indexed pages grows unboundedly. Some details are presented in Cilibrasi and Vitanyi (2007).

5 Related Work

Our approach is to exploit web-derived selectional preferences to improve the dependency parsing. The idea of this paper is inspired by the work of Suzuki et al. (2009) and Pitler et al. (2010). The former uses the web-scale data explicitly to create more data for training the model; while the latter explores the web-scale N-grams data (Lin et al., 2010) for compound bracketing disambiguation. Our research, however, applies the web-scale data (**Google hits** and **Google V1**) to model the word-to-word dependency relationships rather than compound bracketing disambiguation.

Several previous studies have exploited the web-scale data for word pair acquisition. Keller and Lapata (2003) evaluated the utility of using web search engine statistics for unseen bigram. Nakov and Hearst (2005) demonstrated the effectiveness of using search engine statistics to improve the noun compound bracketing. Volk (2001) exploited the WWW as a corpus to resolve PP attachment ambiguities. Turney (2007) measured the semantic orientation for sentiment classification using co-occurrence statistics obtained from the search engines. Bergsma et al. (2010) created robust supervised classifiers via web-scale N-gram data for adjective ordering, spelling correction, noun compound bracketing and verb part-of-speech disambiguation. Our approach, however, extends these techniques to dependency parsing, particularly for long dependency relationships, which involves more challenging tasks than the previous work.

Besides, there are some work exploring the word-to-word co-occurrence derived from the web-scale data or a fixed size of corpus (Calvo and Gelbukh, 2004; Calvo and Gelbukh, 2006; Yates et al., 2006; Drabek and Zhou, 2000; van Noord, 2007) for PP attachment ambiguities or shallow parsing. Johnson and Riezler (2000) incorporated the lexical selectional preference features derived from British National Corpus (Graff, 2003) into a stochastic unification-based grammar. Abekawa and Okumura (2006) improved Japanese dependency parsing by using the co-occurrence information derived from the results of automatic dependency parsing of large-scale corpora. However, we explore the web-scale data for dependency parsing, the performance improves log-linearly with the number of parameters (unique N-grams). To the best of our knowledge, web-derived selectional preference has not been successfully applied to dependency parsing.

6 Conclusion

In this paper, we present a novel method which incorporates the web-derived selectional preferences to improve statistical dependency parsing. The results show that web-scale data improves the dependency parsing, particularly for long dependency relationships. There is no data like more data, performance improves log-linearly with the num-

ber of parameters (unique N-grams). More importantly, when operating on new domains, the web-derived selectional preferences show great potential for achieving robust performance.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 60875041 and No. 61070106), and CSIDM project (No. CSIDM-200805) partially funded by a grant from the National Research Foundation (NRF) administered by the Media Development Authority (MDA) of Singapore. We thank the anonymous reviewers for their insightful comments.

References

T. Abekawa and M. Okumura. 2006. Japanese dependency parsing using co-occurrence information and a combination of case elements. In *Proceedings of ACL-COLING*.

S. Bergsma, D. Lin, and R. Goebel. 2008. Discriminative learning of selectional preference from unlabeled text. In *Proceedings of EMNLP*, pages 59-68.

S. Bergsma, E. Pitler, and D. Lin. 2010. Creating robust supervised classifier via web-scale N-gram data. In *Proceedings of ACL*.

T. Brants and Alex Franz. 2006. The Google Web 1T 5-gram Corpus Version 1.1. LDC2006T13.

H. Calvo and A. Gelbukh. 2004. Acquiring selectional preferences from untagged text for prepositional phrase attachment disambiguation. In *Proceedings of VLDB*.

H. Calvo and A. Gelbukh. 2006. DILUCT: An open-source Spanish dependency parser based on rules, heuristics, and selectional preferences. In *Lecture Notes in Computer Science 3999*, pages 164-175.

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

X. Carreras, M. Collins, and T. Koo. 2008. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of CoNLL*.

E. Charniak, D. Blaheta, N. Ge, K. Hall, and M. Johnson. 2000. BLLIP 1987-89 WSJ Corpus Release 1, LDC No. LDC2000T43.Linguistic Data Consortium.

W. Chen, D. Kawahara, K. Uchimoto, and Torisawa. 2009. Improving dependency parsing with subtrees from auto-parsed data. In *Proceedings of EMNLP*, pages 570-579.

K. W. Church and P. Hanks. 1900. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22-29.

R. L. Cilibrasi and P. M. B. Vitanyi. 2007. The Google similarity distance. IEEE Transaction on Knowledge and Data Engineering, 19(3):2007. pages 370-383.

M. Collins, A. Globerson, T. Koo, X. Carreras, and P. L. Bartlett. 2008. Exponentiated gradient algorithm for conditional random fields and max-margin markov networks. *Journal of Machine Learning Research*, pages 1775–1822.

M. Collins, P. Koehn, and I. Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*, pages 531-540.

S. Corston-Oliver, A. Aue, Kevin. Duh, and E. Ringger. 2006. Multilingual dependency parsing using bayes point machines. In *Proceedings of NAACL*.

H. Daumé III. 2007. Frustrating easy domain adaptation. In *Proceedings of ACL*.

E. F. Drabek and Q. Zhou. 2000. Using co-occurrence statistics as an information source for partial parsing of Chinese. In *Proceedings of Second Chinese Language Processing Workshop, ACL*, pages 22-28.

Y. GoldBerg and M. Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Proceedings of NAACL*, pages 742-750.

D. Graff. 2003. English Gigaword, LDC2003T05.

J. Hall, J. Nivre, and J. Nilsson. 2006. Discriminative classifier for deterministic dependency parsing. In *Proceedings of ACL*, pages 316-323.

M. Johnson and S. Riezler. 2000. Exploiting auxiliary distribution in stochastic unification-based garmmars. In *Proceedings of NAACL*.

T. Koo, X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL*, pages 595-603.

F. Keller and M. Lapata. 2003. Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29(3):459-484.

M. Lapata and F. Keller. 2005. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*, 2(1), pages 1-30.

M. Lauer. 1995. Corpus statistics meet the noun compound: some empirical results. In *Proceedings of ACL*.

D. K. Lin, H. Church, S. Ji, S. Sekine, D. Yarowsky, S. Bergsma, K. Patil, E. Pitler, E. Lathbury, V Rao, K. Dalwani, and S. Narsale. 2010. New tools for web-scale n-grams. In *Proceedings of LREC*.

M.P. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*.

A. F. T. Martins, D. Das, N. A. Smith, and E. P. Xing. 2008. Stacking dependency parsers. In *Proceedings of EMNLP*, pages 157-166.

D. McClosky, E. Charniak, and M. Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of ACL*.

D. McClosky, E. Charniak, and M. Johnson. 2010. Automatic Domain Adapatation for Parsing. In *Proceedings of NAACL-HLT*.

R. McDonald and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP-CoNLL*.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81-88.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91-98.

P. Nakov and M. Hearst. 2005. Search engine statistics beyond the n-gram: application to noun compound bracketing. In *Proceedings of CoNLL*.

J. Nivre and R. McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL*, pages 950-958.

G. van Noord. 2007. Using self-trained bilexical preferences to improve disambiguation accuracy. In *Proceedings of IWPT*, pages 1-10.

PennBioIE. 2005. Mining the bibliome project, 2005. http:bioie.ldc.upenn.edu/.

E. Pitler, S. Bergsma, D. Lin, and K. Church. 2010. Using web-scale N-grams to improve base NP parsing performance. In *Proceedings of COLING*, pages 886-894.

P. Resnik. 1993. *Selection and information: a class-based approach to lexical relationships*. Ph.D. thesis, University of Pennsylvania.

J. Suzuki, H. Isozaki, X. Carreras, and M. Collins. 2009. An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proceedings of EMNLP*, pages 551-560.

J. Suzuki and H. Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *Proceedings of ACL*, pages 665-673.

P. D. Turney. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*, 21(4).

J. Veronis. 2005. Web: Google adjusts its counts. Jean Veronis' blog: http://aixtal.blogsplot.com/2005/03/web-google-adjusts-its-count.html.

M. Volk. 2001. Exploiting the WWW as corpus to resolve PP attachment ambiguities. In *Proceedings of the Corpus Linguistics*.

Q. I. Wang, D. Lin, and D. Schuurmans. 2007. Simple training of dependency parsers via structured boosting. In *Proceedings of IJCAI*, pages 1756-1762.

Yamada and Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, pages 195-206.

A. Yates, S. Schoenmackers, and O. Etzioni. 2006. Detecting parser errors using web-based semantic filters. In *Proceedings of EMNLP*, pages 27-34.

Y. Zhang and S. Clark. 2008. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of EMNLP*, pages 562-571.

Effective Measures of Domain Similarity for Parsing

Barbara Plank
University of Groningen
The Netherlands
b.plank@rug.nl

Gertjan van Noord
University of Groningen
The Netherlands
G.J.M.van.Noord@rug.nl

Abstract

It is well known that parsing accuracy suffers when a model is applied to out-of-domain data. It is also known that the most beneficial data to parse a given domain is data that matches the domain (Sekine, 1997; Gildea, 2001). Hence, an important task is to select appropriate domains. However, most previous work on domain adaptation relied on the implicit assumption that domains are somehow given. As more and more data becomes available, automatic ways to select data that is beneficial for a new (unknown) target domain are becoming attractive. This paper evaluates various ways to automatically acquire related training data for a given test set. The results show that an unsupervised technique based on topic models is effective – it outperforms random data selection on both languages examined, English and Dutch. Moreover, the technique works better than manually assigned labels gathered from meta-data that is available for English.

1 Introduction and Motivation

Previous research on domain adaptation has focused on the task of adapting a system trained on one domain, say newspaper text, to a particular new domain, say biomedical data. Usually, some amount of (labeled or unlabeled) data from the new domain was given – which has been determined by a human.

However, with the growth of the web, more and more data is becoming available, where each document "is potentially its own domain" (McClosky et al., 2010). It is not straightforward to determine which data or model (in case we have several source domain models) will perform best on a new (unknown) target domain. Therefore, an important issue that arises is *how to measure domain similarity*, i.e. whether we can find a simple yet effective method to determine which model or data is most beneficial for an arbitrary piece of new text. Moreover, if we had such a measure, a related question is whether it can tell us something more about what is actually meant by "domain". So far, it was mostly arbitrarily used to refer to some kind of coherent unit (related to topic, style or genre), e.g.: newspaper text, biomedical abstracts, questions, fiction.

Most previous work on domain adaptation, for instance Hara et al. (2005), McClosky et al. (2006), Blitzer et al. (2006), Daumé III (2007), sidestepped this problem of automatic domain selection and adaptation. For parsing, to our knowledge only one recent study has started to examine this issue (McClosky et al., 2010) – we will discuss their approach in Section 2. Rather, an implicit assumption of all of these studies is that domains are given, i.e. that they are represented by the respective corpora. Thus, a corpus has been considered a homogeneous unit. As more data is becoming available, it is unlikely that domains will be 'given'. Moreover, a given corpus might not always be as homogeneous as originally thought (Webber, 2009; Lippincott et al., 2010). For instance, recent work has shown that the well-known Penn Treebank (PT) Wall Street Journal (WSJ) actually contains a variety of genres, including letters, wit and short verse (Webber, 2009).

In this study we take a different approach. Rather than viewing a given corpus as a monolithic entity,

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1566–1576,
Portland, Oregon, June 19-24, 2011.

we break it down to the article-level and disregard corpora boundaries. Given the resulting set of documents (articles), we evaluate various ways to automatically acquire related training data for a given test set, to find answers to the following questions:

- Given a pool of data (a collection of articles from unknown domains) and a test article, is there a way to automatically select data that is relevant for the new domain? If so:
- Which similarity measure is good for parsing?
- How does it compare to human-annotated data?
- Is the measure also useful for other languages and/or tasks?

To this end, we evaluate measures of domain similarity and feature representations and their impact on dependency parsing accuracy. Given a collection of annotated articles, and a new article that we want to parse, we want to select the most similar articles to train the best parser for that new article.

In the following, we will first compare automatic measures to human-annotated labels by examining parsing performance within subdomains of the Penn Treebank WSJ. Then, we extend the experiments to the domain adaptation scenario. Experiments were performed on two languages: English and Dutch. The empirical results show that a simple measure based on topic distributions is effective for both languages and works well also for Part-of-Speech tagging. As the approach is based on plain surface-level information (words) and it finds related data in a completely unsupervised fashion, it can be easily applied to other tasks or languages for which annotated (or automatically annotated) data is available.

2 Related Work

The work most related to ours is McClosky et al. (2010). They try to find the best combination of source models to parse data from a new domain, which is related to Plank and Sima'an (2008). In the latter, unlabeled data was used to create several parsers by weighting trees in the WSJ according to their similarity to the subdomain. McClosky et al. (2010) coined the term *multiple source domain adaptation*. Inspired by work on parsing accuracy prediction (Ravi et al., 2008), they train a linear regression model to predict the best (linear interpolation) of source domain models. Similar to us, McClosky et al. (2010) regard a target domain as mixture of source domains, but they focus on phrase-structure parsing. Furthermore, our approach differs from theirs in two respects: we do not treat source corpora as one entity and try to mix models, but rather consider articles as base units and try to find subsets of related articles (the most similar articles); moreover, instead of creating a supervised model (in their case to predict parsing accuracy), our approach is 'simplistic': we apply measures of domain similarity directly (in an unsupervised fashion), without the necessity to train a supervised model.

Two other related studies are (Lippincott et al., 2010; Van Asch and Daelemans, 2010). Van Asch and Daelemans (2010) explore a measure of domain difference (Renyi divergence) between pairs of domains and its correlation to Part-of-Speech tagging accuracy. Their empirical results show a linear correlation between the measure and the performance loss. Their goal is different, but related: rather than finding related data for a new domain, they want to estimate the loss in accuracy of a PoS tagger when applied to a new domain. We will briefly discuss results obtained with the Renyi divergence in Section 5.1. Lippincott et al. (2010) examine subdomain variation in biomedicine corpora and propose awareness of NLP tools to such variation. However, they did not yet evaluate the effect on a practical task, thus our study is somewhat complementary to theirs.

The issue of data selection has recently been examined for Language Modeling (Moore and Lewis, 2010). A subset of the available data is automatically selected as training data for a Language Model based on a scoring mechanism that compares cross-entropy scores. Their approach considerably outperformed random selection and two previous proposed approaches both based on perplexity scoring.[1]

3 Measures of Domain Similarity

3.1 Measuring Similarity Automatically

Feature Representations A similarity function may be defined over any set of events that are con-

[1] We tested data selection by perplexity scoring, but found the Language Models too small to be useful in our setting.

sidered to be relevant for the task at hand. For parsing, these might be words, characters, n-grams (of words or characters), Part-of-Speech (PoS) tags, bilexical dependencies, syntactic rules, etc. However, to obtain more abstract types such as PoS tags or dependency relations, one would first need to gather respective labels. The necessary tools for this are again trained on particular corpora, and will suffer from domain shifts, rendering labels noisy.

Therefore, we want to gauge the effect of the simplest representation possible: plain surface characteristics (unlabeled text). This has the advantage that we do not need to rely on additional supervised tools; moreover, it is interesting to know how far we can get with this level of information only.

We examine the following feature representations: relative frequencies of words, relative frequencies of character tetragrams, and topic models. Our motivation was as follows. Relative frequencies of words are a simple and effective representation used e.g. in text classification (Manning and Schütze, 1999), while character n-grams have proven successful in genre classification (Wu et al., 2010). Topic models (Blei et al., 2003; Steyvers and Griffiths, 2007) can be considered an advanced model over word distributions: every article is represented by a topic distribution, which in turn is a distribution over words. Similarity between documents can be measured by comparing topic distributions.

Similarity Functions There are many possible similarity (or distance) functions. They fall broadly into two categories: probabilistically-motivated and geometrically-motivated functions. The similarity functions examined in this study will be described in the following.

The *Kullback-Leibler (KL) divergence* $D(q||r)$ is a classical measure of 'distance'[2] between two probability distributions, and is defined as: $D(q||r) = \sum_y q(y) \log \frac{q(y)}{r(y)}$. It is a non-negative, additive, asymmetric measure, and 0 iff the two distributions are identical. However, the KL-divergence is undefined if there exists an event y such that $q(y) > 0$ but $r(y) = 0$, which is a property that "makes it unsuitable for distributions derived via maximum-likelihood estimates" (Lee, 2001).

[2]It is not a proper distance metric since it is asymmetric.

One option to overcome this limitation is to apply smoothing techniques to gather non-zero estimates for all y. The alternative, examined in this paper, is to consider an approximation to the KL divergence, such as the Jensen-Shannon (JS) divergence (Lin, 1991) and the skew divergence (Lee, 2001).

The *Jensen-Shannon divergence*, which is symmetric, computes the KL-divergence between q, r, and the average between the two. We use the JS divergence as defined in Lee (2001): $JS(q,r) = \frac{1}{2}[D(q||\text{avg}(q,r)) + D(r||\text{avg}(q,r))]$. The asymmetric *skew divergence* s_α, proposed by Lee (2001), mixes one distribution with the other by a degree defined by $\alpha \in [0,1)$: $s_\alpha(q,r,\alpha) = D(q||\alpha r + (1-\alpha)q)$. As α approaches 1, the skew divergence approximates the KL-divergence.

An alternative way to measure similarity is to consider the distributions as vectors and apply geometrically-motivated distance functions. This family of similarity functions includes the *cosine* $cos(q,r) = q(y) \cdot r(y)/||q(y)||||r(y)||$, *euclidean* $euc(q,r) = \sqrt{\sum_y (q(y) - r(y))^2}$ and *variational* (also known as L1 or Manhattan) distance function, defined as $var(q,r) = \sum_y |q(y) - r(y)|$.

3.2 Human-annotated data

In contrast to the automatic measures devised in the previous section, we might have access to human annotated data. That is, use label information such as topic or genre to define the set of similar articles.

Genre For the Penn Treebank (PT) Wall Street Journal (WSJ) section, more specifically, the subset available in the Penn Discourse Treebank, there exists a partition of the data by *genre* (Webber, 2009). Every article is assigned one of the following genre labels: news, letters, highlights, essays, errata, wit and short verse, quarterly progress reports, notable and quotable. This classification has been made on the basis of meta-data (Webber, 2009). It is well-known that there is no meta-data directly associated with the individual WSJ files in the Penn Treebank. However, meta-data can be obtained by looking at the articles in the ACL/DCI corpus (LDC99T42), and a mapping file that aligns document numbers of DCI (DOCNO) to WSJ keys (Webber, 2009). An example document is given in Figure 1. The meta-data field `HL` contains headlines, `SO` source info, and

the IN field includes topic markers.

```
<DOC><DOCNO>  891102-0186. </DOCNO>
<WSJKEY> wsj_0008 </WSJKEY>
<AN> 891102-0186. </AN>
<HL> U.S. Savings Bonds Sales
@  Suspended by Debt Limit </HL>
<DD> 11/02/89 </DD>
<SO> WALL STREET JOURNAL (J) </SO>
<IN> FINANCIAL, ACCOUNTING, LEASING (FIN)
BOND MARKET NEWS (BON) </IN>
<GV> TREASURY DEPARTMENT (TRE) </GV>
<DATELINE> WASHINGTON  </DATELINE>
<TXT>
<p><s>
The federal government suspended sales of U.S.
savings bonds because Congress hasn't lifted
the ceiling on government debt.</s></p> [...]
```

Figure 1: Example of ACL/DCI article. We have augmented it with the WSJ filename (WSJKEY).

Topic On the basis of the same meta-data, we devised a classification of the Penn Treebank WSJ by *topic*. That is, while the genre division has been mostly made on the basis of headlines, we use the information of the IN field. Every article is assigned one, more than one or none of a predefined set of keywords. While their origin remains unclear,[3] these keywords seem to come from a controlled vocabulary. There are 76 distinct topic markers. The three most frequent keywords are: TENDER OFFERS, MERGERS, ACQUISITIONS (TNM), EARNINGS (ERN), STOCK MARKET, OFFERINGS (STK). This reflects the fact that a lot of articles come from the financial domain. But the corpus also contains articles from more distant domains, like MARKETING, ADVERTISING (MKT), COMPUTERS AND INFORMATION TECHNOLOGY (CPR), HEALTH CARE PROVIDERS, MEDICINE, DENTISTRY (HEA), PETROLEUM (PET).

4 Experimental Setup

4.1 Tools & Evaluation

The parsing system used in this study is the MST parser (McDonald et al., 2005), a state-of-the-art data-driven graph-based dependency parser. It is a system that can be trained on a variety of languages given training data in CoNLL format (Buchholz and Marsi, 2006). Additionally, the parser implements both projective and non-projective parsing algorithms. The projective algorithm is used for the experiments on English, while the non-projective variant is used for Dutch. We train the parser using default settings. MST takes PoS-tagged data as input; we use gold-standard tags in the experiments.

We estimate topic models using Latent Dirichlet Allocation (Blei et al., 2003) implemented in the MALLET[4] toolkit. Like Lippincott et al. (2010), we set the number of topics to 100, and otherwise use standard settings (no further optimization). We experimented with the removal of stopwords, but found no deteriorating effect while keeping them. Thus, all experiments are carried out on data where stopwords were not removed.

We implemented the similarity measures presented in Section 3.1. For skew divergence, that requires parameter α, we set $\alpha = .99$ (close to KL divergence) since that has shown previously to work best (Lee, 2001). Additionally, we evaluate the approach on English PoS tagging using two different taggers: MXPOST, the MaxEnt tagger of Ratnaparkhi[5] and Citar,[6] a trigram HMM tagger.

In all experiments, parsing performance is measured as *Labeled Attachment Score* (LAS), the percentage of tokens with correct dependency edge and label. To compute LAS, we use the CoNLL 2007 evaluation script[7] with punctuation tokens excluded from scoring (as was the default setting in CoNLL 2006). PoS tagging accuracy is measured as the percentage of correctly labeled words out of all words. Statistical significance is determined by *Approximate Randomization Test* (Noreen, 1989; Yeh, 2000) with 10,000 iterations.

4.2 Data

English - WSJ For English, we use the portion of the Penn Treebank Wall Street Journal (WSJ) that has been made available in the CoNLL 2008 shared

[3] It is not known what IN stands for, as also stated in Mark Liberman's notes in the readme of the ACL/DCI corpus. However, a reviewer suggested that IN might stand for "index terms" which seems plausible.

[4] http://mallet.cs.umass.edu/

[5] ftp://ftp.cis.upenn.edu/pub/adwait/jmx/

[6] Citar has been implemented by Daniël de Kok and is available at: https://github.com/danieldk/citar

[7] http://nextens.uvt.nl/depparse-wiki/

task. This data has been automatically converted[8] into dependency structure, and contains three files: the training set (sections 02-21), development set (section 24) and test set (section 23).

Since we use articles as basic units, we actually split the data to get back original article boundaries.[9] This led to a total of 2,034 articles (1 million words). Further statistics on the datasets are given in Table 1. In the first set of experiments on WSJ subdomains, we consider articles from section 23 and 24 that contain at least 50 sentences as test sets (target domains). This amounted to 22 test articles.

	EN: WSJ	WSJ+G+B	Dutch
articles	2,034	3,776	51,454
sentences	43,117	77,422	1,663,032
words	1,051,997	1,784,543	20,953,850

Table 1: Overview of the datasets for English and Dutch.

To test whether we have a reasonable system, we performed a sanity check and trained the MST parser on the training section (02-21). The result on the standard test set (section 23) is identical to previously reported results (excluding punctuation tokens: LAS 87.50, Unlabeled Attachment Score (UAS) 90.75; with punctuation tokens: LAS 87.07, UAS 89.95). The latter has been reported in (Surdeanu and Manning, 2010).

English - Genia (G) & Brown (B) For the Domain Adaptation experiments, we added 1,552 articles from the GENIA[10] treebank (biomedical abstracts from Medline) and 190 files from the Brown corpus to the pool of data. We converted the data to CoNLL format with the LTH converter (Johansson and Nugues, 2007). The size of the test files is, respectively: Genia 1,360 sentences with an average number of 26.20 words per sentence; the Brown test set is the same as used in the CoNLL 2008 shared task and contains 426 sentences with a mean of 16.80 words.

[8]Using the LTH converter: `http://nlp.cs.lth.se/software/treebank_converter/`

[9]This was a non-trivial task, as we actually noticed that some sentences have been omitted from the CoNLL 2008 shared task.

[10]We use the GENIA distribution in Penn Treebank format available at `http://bllip.cs.brown.edu/download/genia1.0-division-rel1.tar.gz`

5 Experiments on English

5.1 Experiments within the WSJ

In the first set of experiments, we focus on the WSJ and evaluate the similarity functions to gather related data for a given test article. We have 22 WSJ articles as test set, sampled from sections 23 and 24. Regarding feature representations, we examined three possibilities: relative frequencies of words, relative frequencies of character tetragrams (both unsmoothed) and document topic distributions.

In the following, we only discuss representations based on words or topic models as we found character tetragrams less stable; they performed sometimes like their word-based counterparts but other times, considerably worse.

Results of Similarity Measures Table 2 compares the effect of the different ways to select related data in comparison to the random baseline for increasing amounts of training data. The table gives the average over 22 test articles (rather than showing individual tables for the 22 articles). We select articles up to various thresholds that specify the total number of sentences selected in each round (e.g. 0.3k, 1.2k, etc.).[11] In more detail, Table 2 shows the result of applying various similarity functions (introduced in Section 3.1) over the two different feature representations (w: words; tm: topic model) for increasing amounts of data. We additionally provide results of using the Renyi divergence.[12]

Clearly, as more and more data is selected, the differences become smaller, because we are close to the data limit. However, for all data points less than 38k (97%), selection by jensen-shannon, variational and cosine similarity outperform random data selection significantly for both types of feature representations (words and topic model). For selection by topic models, this additionally holds for the euclidean measure.

From the various measures we can see that selection by jensen-shannon divergence and variational distance perform best, followed by cosine similarity, skew divergence, euclidean and renyi.

[11]Rather than choosing k articles, as article length may differ.

[12]The *Renyi divergence* (Rényi, 1961), also used by Van Asch and Daelemans (2010), is defined as $D_\alpha(q,r) = 1/(\alpha - 1)\log(\sum q^\alpha r^{1-\alpha})$.

	1% (0.3k)	**3%** (1.2k)	**25%** (9.6k)	**49%** (19.2k)	**97%** (38k)
random	70.61	77.21	82.98	84.48	85.51
w-js	74.07⋆	79.41⋆	83.98⋆	84.94⋆	85.68
w-var	74.07⋆	**79.60**⋆	83.82⋆	84.94⋆	85.45
w-skw	74.20⋆	78.95⋆	83.68⋆	84.60	85.55
w-cos	73.77⋆	79.30⋆	83.87⋆	84.96⋆	85.59
w-euc	73.85⋆	78.90⋆	83.52⋆	84.68	85.57
w-ryi	73.41⋆	78.31	83.76⋆	84.46	85.46
tm-js	74.23⋆	79.49⋆	84.04⋆	85.01⋆	85.45
tm-var	**74.29**⋆	79.59⋆	83.93⋆	84.94⋆	85.43
tm-skw	74.13⋆	79.42⋆	84.13⋆	84.82	**85.73**
tm-cos	74.04⋆	79.27⋆	**84.14**⋆	84.99⋆	85.42
tm-euc	74.27⋆	79.53⋆	83.93⋆	**85.15**⋆	85.62
tm-ryi	71.26	78.64⋆	83.79⋆	84.85	85.58

Table 2: Comparison of similarity measures based on words (w) and topic model (tm): parsing accuracy for increasing amounts of training data as average over 22 WSJ articles (js=jensen-shannon; cos=cosine; skw=skew; var=variational; euc=euclidean; ryi=renyi). Best score (per representation) underlined, best overall score bold; ⋆ indicates significantly better ($p < 0.05$) than random.

Renyi divergence does not perform as well as other probabilistically-motivated functions. Regarding feature representations, the representation based on topic models works slightly better than the respective word-based measure (cf. Table 2) and often achieves the overall best score (boldface).

Overall, the differences in accuracy between the various similarity measures are small; but interestingly, the overlap between them is not that large. Table 3 and Table 4 show the overlap (in terms of proportion of identically selected articles) between pairs of similarity measures. As shown in Table 3, for all measures there is only a small overlap with the random baseline (around 10%-14%). Despite similar performance, topic model selection has interestingly no substantial overlap with any other word-based similarity measures: their overlap is at most 41.6%. Moreover, Table 4 compares the overlap of the various similarity functions within a certain feature representation (here x stands for either topic model – left value – or words – right value). The table shows that there is quite some overlap between jensen-shannon, variational and skew divergence on one side, and cosine and euclidean on the other side, i.e. between probabilistically- and geometrically-motivated functions. Variational has a higher overlap with the probabilistic functions. Interestingly, the 'peaks' in Table 4 (underlined, i.e. the highest pair-wise overlaps) are the same for the different feature representations.

In the following we analyze selection by topic model and words, as they are relatively different from each other, despite similar performance. For the word-based model, we use jensen-shannon as similarity function, as it turned out to be the best measure. For topic model, we use the simpler variational metric. However, very similar results were achieved using jensen-shannon. Cosine and euclidean did not perform as well.

	ran	w-js	w-var	w-skw	w-cos	w-euc
ran	–	10.3	10.4	10.0	10.4	10.2
tm-js	12.1	41.6	39.6	36.0	29.3	28.6
tm-var	12.3	40.8	39.3	34.9	29.3	28.5
tm-skw	11.8	40.9	39.7	36.8	30.0	30.1
tm-cos	14.0	31.7	30.7	27.3	24.1	23.2
tm-euc	14.6	27.5	27.2	23.4	22.6	22.1

Table 3: Average overlap (in %) of similarity measure: random selection (ran) vs. measures based on words (w) and topic model (tm).

x=tm/w	x-js	x-var	x-skw	x-cos	x-euc
tm/w-var	76/74	–	60/63	55/48	49/47
tm/w-skw	69/72	60/63	–	48/41	42/42
tm/w-cos	57/42	55/48	48/41	–	62/71
tm/w-euc	47/41	49/47	42/42	62/71	–

Table 4: Average overlap (in %) for different feature representations x as tm/w, where tm=topic model and w=words. Highest pair-wise overlap is underlined.

Automatic Measure vs. Human labels The next question is how these automatic measures compare to human-annotated data. We compare word-based and topic model selection (by using jensen-shannon and variational, respectively) to selection based on human-given labels: genre and topic. For genre, we randomly select larger amounts of training data for a given test article from the same genre. For topic, the approach is similar, but as an article might have

several topic markers (keywords in the IN field), we rank articles by proportion of overlapping keywords.

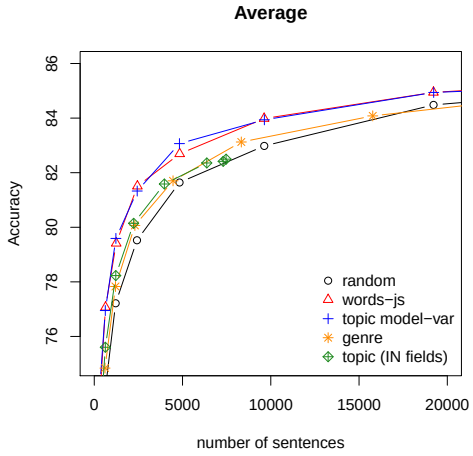


Figure 2: Comparison of automatic measures (words using jensen-shannon and topic model using variational) with human-annotated labels (genre/topic). Automatic measures outperform human labels ($p < 0.05$).

Figure 2 shows that human-labels do actually not perform better than the automatic measures. Both are close to random selection. Moreover, the line of selection by topic marker (IN fields) stops early – we believe the reason for this is that the IN fields are too fine-grained, which limits the number of articles that are considered relevant for a given test article. However, manually aggregating articles on similar topics did not improve topic-based selection either. We conclude that the automatic selection techniques perform significantly better than human-annotated data, at least within the WSJ domain considered here.

5.2 Domain Adaptation Results

Until now, we compared similarity measures by restricting ourselves to articles from the WSJ. In this section, we extend the experiments to the domain adaptation scenario. We augment the pool of WSJ articles with articles coming from two other corpora: Genia and Brown. We want to gauge the effectiveness of the domain similarity measures in the multi-domain setting, where articles are selected from the pool of data without knowing their identity (which corpus the articles came from).

The test sets are the standard evaluation sets from the three corpora: the standard WSJ (section 23) and Brown test set from CoNLL 2008 (they contain 2,399 and 426 sentences, respectively) and the Genia test set (1,370 sentences). As a reference, we give results of models trained on the respective corpora (*per-corpus* models; i.e. if we consider corpora boundaries and train a model on the respective domain – this model is 'supervised' in the sense that it knows from which corpus the test article came from) as well as a baseline model trained on all data, i.e. the *union* of all three corpora (wsj+genia+brown), which is a standard baseline in domain adaptation (Daumé III, 2007; McClosky et al., 2010).

	WSJ (38k)	**Brown** (28k)	**Genia** (19k)
random	86.58	73.81	83.77
per-corpus	87.50	81.55	86.63
union	87.05	79.12	81.57
topic model (var)	87.11⋆	81.76♢	86.77♢
words (js)	86.30	81.47♢	86.44♢

Table 5: Domain Adaptation Results on English (significantly better: ⋆ than random; ♢ than random and union).

The learning curves are shown in Figure 3, the scores for a specific amount of data are given in Table 5. The performance of the reference models (per-corpus and union in Table 5) are indicated in Figure 3 with horizontal lines: the dashed line represents the per-corpus performance ('supervised' model); the solid line shows the performance of the union baseline trained on all available data (77k sentences). For the former, the vertical dashed lines indicate the amount of data the model was trained on (e.g. 23k sentences for Brown).

Simply taking all available data has a deteriorating effect: on all three test sets, the performance of the union model is below the presumably best performance of a model trained on the respective corpus (per-corpus model).

The empirical results show that automatic data selection by topic model outperforms random selection on all three test sets and the union baseline in two out of three cases. More specifically, selection by topic model outperforms random selection significantly on all three test sets and all points in the graph ($p < 0.001$). Selection by the word-based measure (words-js) achieves a significant improve-

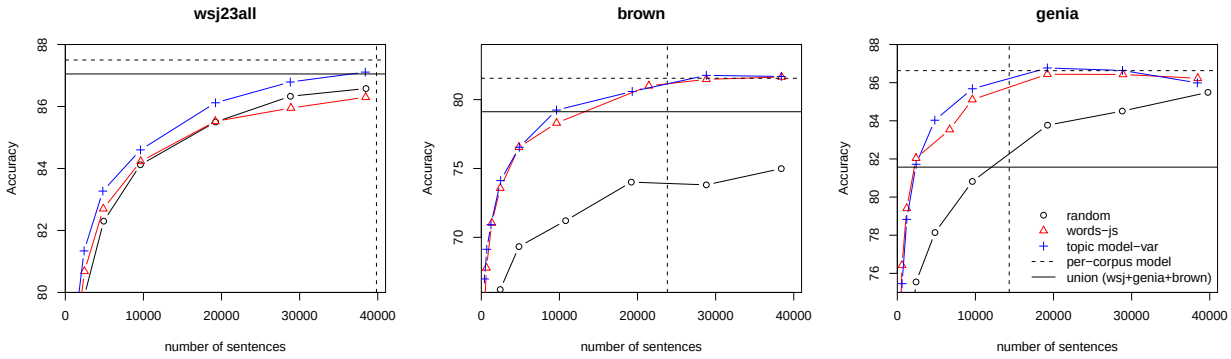


Figure 3: Domain Adaptation Results for English Parsing with Increasing Amounts of Training Data. The vertical line represents the amount of data the per-corpus model is trained on.

ment over the random baseline on two out of the three test sets – it falls below the random baseline on the WSJ test set. Thus, selection by topic model performs best – it achieves better performance than the union baseline with comparatively little data (Genia: 4k; Brown: 19k – in comparison: union has 77k). Moreover, it comes very close to the supervised per-corpus model performance[13] with a similar amount of data (cf. vertical dashed line). This is a very good result, given that the technique disregards the origin of the articles and just uses plain words as information. It automatically finds data that is beneficial for an unknown target domain.

So far we examined domain similarity measures for parsing, and concluded that selection by topic model performs best, closely followed by word-based selection using the jensen-shannon divergence. The question that remains is whether the measure is more widely applicable: How does it perform on another language and task?

PoS tagging We perform similar Domain Adaptation experiments on WSJ, Genia and Brown for PoS tagging. We use two taggers (HMM and MaxEnt) and the same three test articles as before. The results are shown in Figure 4 (it depicts the average over the three test sets, WSJ, Genia, Brown, for space reasons). The left figure shows the performance of the HMM tagger; on the right is the MaxEnt tagger. The graphs show that automatic training data selection outperforms random data selection, and again topic model selection performs best, closely followed by words-js. This confirms previous findings and shows that the domain similarity measures are effective also for this task.

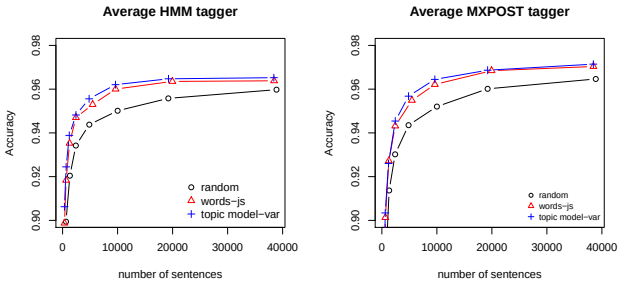


Figure 4: PoS tagging results, average over 3 test sets.

6 Experiments on Dutch

For Dutch, we evaluate the approach on a bigger and more varied dataset. It contains in total over 50k articles and 20 million words (cf. Table 1). In contrast to the English data, only a small portion of the dataset is manually annotated: 281 articles.[14]

Since we want to evaluate the performance of different similarity measures, we want to keep the influence of noise as low as possible. Therefore, we annotated the remaining articles with a parsing system that is more accurate (Plank and van Noord, 2010), the Alpino parser (van Noord, 2006). Note that using a more accurate parsing system to train another parser has recently also been proposed by Petrov et al. (2010) as *uptraining*. Alpino is a

[13] On Genia and Brown (cf. Table 5) there is no significant difference between topic model and per-corpus model.

[14] http://www.let.rug.nl/vannoord/Lassy/

parser tailored to Dutch, that has been developed over the last ten years, and reaches an accuracy level of 90% on general newspaper text. It uses a conditional MaxEnt model as parse selection component. Details of the parser are given in (van Noord, 2006).

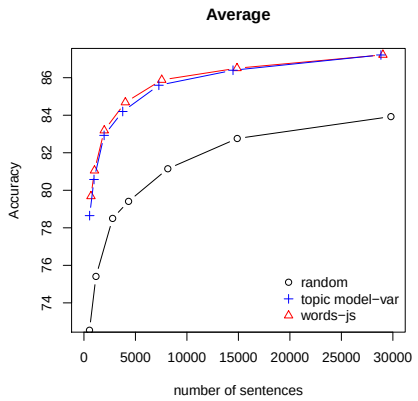


Figure 5: Result on Dutch; average over 30 articles.

Data and Results The Dutch dataset contains articles from a variety of sources: Wikipedia[15], EMEA[16] (documents from the European Medicines Agency) and the Dutch parallel corpus[17] (DPC), that covers a variety of subdomains. The Dutch articles were parsed with Alpino and automatically converted to CoNLL format with the treebank conversion software from CoNLL 2006, where PoS tags have been replaced with more fine-grained Alpino tags as that had a positive effect on MST. The 281 annotated articles come from all three sources. As with English, we consider as test set articles with at least 50 sentences, from which 30 are randomly sampled.

The results on Dutch are shown in Figure 5. Domain similarity measures clearly outperform random data selection also in this setting with another language and a considerably larger pool of data (20 million words; 51k articles).

7 Discussion

In this paper we have shown the effectiveness of a simple technique that considers only plain words as domain selection measure for two tasks, dependency parsing and PoS tagging. Interestingly, human-annotated labels did not perform better than the automatic measures. The best technique is based on topic models, and compares document topic distributions estimated by LDA (Blei et al., 2003) using the variational metric (very similar results were obtained using jensen-shannon). Topic model selection significantly outperforms random data selection on both examined languages, English and Dutch, and has a positive effect on PoS tagging. Moreover, it outperformed a standard Domain Adaptation baseline (union) on two out of three test sets. Topic model is closely followed by the word-based measure using jensen-shannon divergence. By examining the overlap between word-based and topic model-based techniques, we found that despite similar performance their overlap is rather small. Given these results and the fact that no optimization has been done for the topic model itself, results are encouraging: there might be an even better measure that exploits the information from both techniques. So far, we tested a simple combination of the two by selecting half of the articles by a measure based on words and the other half by a measure based on topic models (by testing different metrics). However, this simple combination technique did not improve results yet – topic model alone still performed best.

Overall, plain surface characteristics seem to carry important information of what kind of data is relevant for a given domain. Undoubtedly, parsing accuracy will be influenced by more factors than lexical information. Nevertheless, as we have seen, lexical differences constitute an important factor.

Applying divergence measures over syntactic patterns, adding additional articles to the pool of data (by uptraining (Petrov et al., 2010), selftraining (McClosky et al., 2006) or active learning (Hwa, 2004)), gauging the effect of weighting instances according to their similarity to the test data (Jiang and Zhai, 2007; Plank and Sima'an, 2008), as well as analyzing differences between gathered data are venues for further research.

Acknowledgments

The authors would like to thank Bonnie Webber and the three anonymous reviewers for their valuable comments on earlier drafts of this paper.

[15] http://ilps.science.uva.nl/WikiXML/
[16] http://urd.let.rug.nl/tiedeman/OPUS/EMEA.php
[17] http://www.kuleuven-kortrijk.be/DPC

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain Adaptation with Structural Correspondence Learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City.

Hal Daumé III. 2007. Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics*, Prague, Czech Republic.

Daniel Gildea. 2001. Corpus Variation and Parser Performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, Pittsburgh, PA.

Tadayoshi Hara, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Adapting a Probabilistic Disambiguation Model of an HPSG Parser to a New Domain. In Robert Dale, Kam-Fai Wong, Jian Su, and Oi Yee Kwong, editors, *Natural Language Processing IJCNLP 2005*, volume 3651 of *Lecture Notes in Computer Science*, pages 199–210. Springer Berlin / Heidelberg.

Rebecca Hwa. 2004. Sample Selection for Statistical Parsing. *Compututational Linguistics*, 30:253–276, September.

Jing Jiang and ChengXiang Zhai. 2007. Instance Weighting for Domain Adaptation in NLP. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics*, pages 264–271, Prague, Czech Republic, June. Association for Computational Linguistics.

Richard Johansson and Pierre Nugues. 2007. Extended Constituent-to-dependency Conversion for English. In *Proceedings of NODALIDA*, Tartu, Estonia.

Lillian Lee. 2001. On the Effectiveness of the Skew Divergence for Statistical Language Analysis. In *In Artificial Intelligence and Statistics 2001*, pages 65–72, Key West, Florida.

J. Lin. 1991. Divergence measures based on the Shannon entropy. *Information Theory, IEEE Transactions on*, 37(1):145 –151, January.

Tom Lippincott, Diarmuid Ó Séaghdha, Lin Sun, and Anna Korhonen. 2010. Exploring variation across biomedical subdomains. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 689–697, Beijing, China, August.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge Mass.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective Self-Training for Parsing. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 152–159, Brooklyn, New York. Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic Domain Adaptation for Parsing. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 28–36, Los Angeles, California, June. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Robert C. Moore and William Lewis. 2010. Intelligent Selection of Language Model Training Data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224, Uppsala, Sweden, July. Association for Computational Linguistics.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. Wiley-Interscience.

Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. 2010. Uptraining for Accurate Deterministic Question Parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 705–713, Cambridge, MA, October. Association for Computational Linguistics.

Barbara Plank and Khalil Sima'an. 2008. Subdomain Sensitive Statistical Parsing using Raw Corpora. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, Marrakech, Morocco, May.

Barbara Plank and Gertjan van Noord. 2010. Grammar-Driven versus Data-Driven: Which Parsing System Is More Affected by Domain Shifts? In *Proceedings of the 2010 Workshop on NLP and Linguistics: Finding the Common Ground*, pages 25–33, Uppsala, Sweden, July. Association for Computational Linguistics.

Sujith Ravi, Kevin Knight, and Radu Soricut. 2008. Automatic Prediction of Parser Accuracy. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 887–

896, Morristown, NJ, USA. Association for Computational Linguistics.

A. Rényi. 1961. On measures of information and entropy. In *Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability*, pages 547–561, Berkeley.

Satoshi Sekine. 1997. The Domain Dependence of Parsing. In *In Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 96–102, Washington D.C.

Mark Steyvers and Tom Griffiths, 2007. *Probabilistic Topic Models*. Lawrence Erlbaum Associates.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble Models for Dependency Parsing: Cheap and Good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 649–652, Los Angeles, California, June. Association for Computational Linguistics.

Vincent Van Asch and Walter Daelemans. 2010. Using Domain Similarity for Performance Estimation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 31–36, Uppsala, Sweden, July. Association for Computational Linguistics.

Gertjan van Noord. 2006. At Last Parsing Is Now Operational. In *TALN 2006 Verbum Ex Machina, Actes De La 13e Conference sur Le Traitement Automatique des Langues naturelles*, pages 20–42, Leuven.

Bonnie Webber. 2009. Genre distinctions for Discourse in the Penn TreeBank. In *Proceedings of the 47th Meeting of the Association for Computational Linguistics*, pages 674–682, Suntec, Singapore, August. Association for Computational Linguistics.

Zhili Wu, Katja Markert, and Serge Sharoff. 2010. Fine-Grained Genre Classification Using Structural Learning Algorithms. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 749–759, Uppsala, Sweden, July. Association for Computational Linguistics.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics*, pages 947–953, Morristown, NJ, USA. Association for Computational Linguistics.

Efficient CCG Parsing: A* versus Adaptive Supertagging

Michael Auli
School of Informatics
University of Edinburgh
m.auli@sms.ed.ac.uk

Adam Lopez
HLTCOE
Johns Hopkins University
alopez@cs.jhu.edu

Abstract

We present a systematic comparison and combination of two orthogonal techniques for efficient parsing of Combinatory Categorial Grammar (CCG). First we consider adaptive supertagging, a widely used approximate search technique that prunes most lexical categories from the parser's search space using a separate sequence model. Next we consider several variants on A*, a classic exact search technique which to our knowledge has not been applied to more expressive grammar formalisms like CCG. In addition to standard hardware-independent measures of parser effort we also present what we believe is the first evaluation of A* parsing on the more realistic but more stringent metric of CPU time. By itself, A* substantially reduces parser effort as measured by the number of edges considered during parsing, but we show that for CCG this does not always correspond to improvements in CPU time over a CKY baseline. Combining A* with adaptive supertagging decreases CPU time by 15% for our best model.

1 Introduction

Efficient parsing of Combinatorial Categorial Grammar (CCG; Steedman, 2000) is a longstanding problem in computational linguistics. Even with practical CCG that are strongly context-free (Fowler and Penn, 2010), parsing can be much harder than with Penn Treebank-style context-free grammars, since the number of nonterminal categories is generally much larger, leading to increased grammar constants. Where a typical Penn Treebank grammar may have fewer than 100 nonterminals (Hockenmaier and Steedman, 2002), we found that a CCG grammar derived from CCGbank contained nearly 1600. The same grammar assigns an average of 26 lexical categories per word, resulting in a very large space of possible derivations.

The most successful strategy to date for efficient parsing of CCG is to first prune the set of lexical categories considered for each word, using the output of a *supertagger*, a sequence model over these categories (Bangalore and Joshi, 1999; Clark, 2002). Variations on this approach drive the widely-used, broad coverage C&C parser (Clark and Curran, 2004; Clark and Curran, 2007). However, pruning means approximate search: if a lexical category used by the highest probability derivation is pruned, the parser will not find that derivation (§2). Since the supertagger enforces no grammaticality constraints it may even prefer a sequence of lexical categories that cannot be combined into *any* derivation (Figure 1). Empirically, we show that supertagging improves efficiency by an order of magnitude, but the tradeoff is a significant loss in accuracy (§3).

Can we improve on this tradeoff? The line of investigation we pursue in this paper is to consider more efficient *exact* algorithms. In particular, we test different variants of the classical A* algorithm (Hart et al., 1968), which has met with success in Penn Treebank parsing with context-free grammars (Klein and Manning, 2003; Pauls and Klein, 2009a; Pauls and Klein, 2009b). We can substitute A* for standard CKY on either the unpruned set of lexical categories, or the pruned set resulting from su-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1577–1585,
Portland, Oregon, June 19-24, 2011.

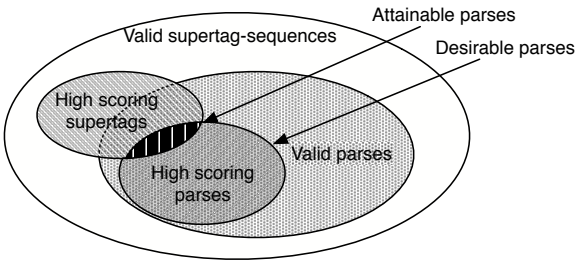
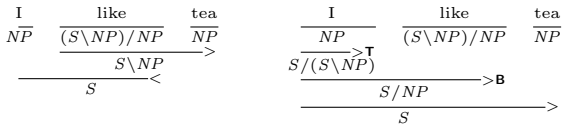


Figure 1: The relationship between supertagger and parser search spaces based on the intersection of their corresponding tag sequences.

pertagging. Our empirical results show that on the unpruned set of lexical categories, heuristics employed for context-free grammars show substantial speedups in hardware-independent metrics of parser effort (§4). To understand how this compares to the CKY baseline, we conduct a carefully controlled set of timing experiments. Although our results show that improvements on hardware-independent metrics do not always translate into improvements in CPU time due to increased processing costs that are hidden by these metrics, they also show that when the lexical categories are pruned using the output of a supertagger, then we can still improve efficiency by 15% with A* techniques (§5).

2 CCG and Parsing Algorithms

CCG is a lexicalized grammar formalism encoding for each word lexical categories which are either basic (eg. NN, JJ) or complex. Complex lexical categories specify the number and directionality of arguments. For example, one lexical category (of over 100 in our model) for the transitive verb *like* is $(S\backslash NP_2)/NP_1$, specifying the first argument as an NP to the right and the second as an NP to the left. In parsing, adjacent spans are combined using a small number of binary combinatory rules like forward application or composition on the spanning categories (Steedman, 2000; Fowler and Penn, 2010). In the first derivation below, $(S\backslash NP)/NP$ and NP combine to form the spanning category $S\backslash NP$, which only requires an NP to its left to form a complete sentence-spanning S. The second derivation uses type-raising to change the category type of *I*.



Because of the number of lexical categories and their complexity, a key difficulty in parsing CCG is that the number of analyses for each span of the sentence quickly becomes extremely large, even with efficient dynamic programming.

2.1 Adaptive Supertagging

Supertagging (Bangalore and Joshi, 1999) treats the assignment of lexical categories (or *supertags*) as a sequence tagging problem. Once the supertagger has been run, lexical categories that apply to each word in the input sentence are pruned to contain only those with high posterior probability (or other figure of merit) under the supertagging model (Clark and Curran, 2004). The posterior probabilities are then discarded; it is the extensive pruning of lexical categories that leads to substantially faster parsing times.

Pruning the categories in advance this way has a specific failure mode: sometimes it is not possible to produce a sentence-spanning derivation from the tag sequences preferred by the supertagger, since it does not enforce grammaticality. A workaround for this problem is the *adaptive supertagging* (AST) approach of Clark and Curran (2004). It is based on a step function over supertagger beam ratios, relaxing the pruning threshold for lexical categories whenever the parser fails to find an analysis. The process either succeeds and returns a parse after some iteration or gives up after a predefined number of iterations. As Clark and Curran (2004) show, most sentences can be parsed with a very small number of supertags per word. However, the technique is inherently approximate: it will return a lower probability parse under the parsing model if a higher probability parse can only be constructed from a supertag sequence returned by a subsequent iteration. In this way it prioritizes speed over accuracy, although the tradeoff can be modified by adjusting the beam step function.

2.2 A* Parsing

Irrespective of whether lexical categories are pruned in advance using the output of a supertagger, the CCG parsers we are aware of all use some vari-

ant of the CKY algorithm. Although CKY is easy to implement, it is *exhaustive*: it explores all possible analyses of all possible spans, irrespective of whether such analyses are likely to be part of the highest probability derivation. Hence it seems natural to consider exact algorithms that are more efficient than CKY.

A* search is an agenda-based best-first graph search algorithm which finds the lowest cost parse exactly without necessarily traversing the entire search space (Klein and Manning, 2003). In contrast to CKY, items are not processed in topological order using a simple control loop. Instead, they are processed from a priority queue, which orders them by the product of their inside probability and a heuristic estimate of their outside probability. Provided that the heuristic never underestimates the true outside probability (i.e. it is *admissible*) the solution is guaranteed to be exact. Heuristics are model specific and we consider several variants in our experiments based on the CFG heuristics developed by Klein and Manning (2003) and Pauls and Klein (2009a).

3 Adaptive Supertagging Experiments

Parser. For our experiments we used the generative CCG parser of Hockenmaier and Steedman (2002). Generative parsers have the property that all edge weights are non-negative, which is required for A* techniques.[1] Although not quite as accurate as the discriminative parser of Clark and Curran (2007) in our preliminary experiments, this parser is still quite competitive. It is written in Java and implements the CKY algorithm with a global pruning threshold of 10^{-4} for the models we use. We focus on two parsing models: `PCFG`, the baseline of Hockenmaier and Steedman (2002) which treats the grammar as a PCFG (Table 1); and `HWDep`, a headword dependency model which is the best performing model of the parser. The PCFG model simply generates a tree top down and uses very simple structural probabilities while the `HWDep` model conditions node expansions on headwords and their lexical categories.

Supertagger. For supertagging we used Dennis Mehay's implementation, which follows Clark (2002).[2] Due to differences in smoothing of the supertagging and parsing models, we occasionally drop supertags returned by the supertagger because they do not appear in the parsing model [3].

Evaluation. All experiments were conducted on CCGBank (Hockenmaier and Steedman, 2007), a right-most normal-form CCG version of the Penn Treebank. Models were trained on sections 2-21, tuned on section 00, and tested on section 23. Parsing accuracy is measured using labelled and unlabelled predicate argument structure recovery (Clark and Hockenmaier, 2002); we evaluate on *all* sentences and thus penalise lower coverage. All timing experiments reported in the paper were run on a 2.5 GHz Xeon machine with 32 GB memory and are averaged over ten runs[4].

3.1 Results

Supertagging has been shown to improve the speed of a generative parser, although little analysis has been reported beyond the speedups (Clark, 2002) We ran experiments to understand the time/accuracy tradeoff of adaptive supertagging, and to serve as baselines.

Adaptive supertagging is parametrized by a beam size β and a dictionary cutoff k that bounds the number of lexical categories considered for each word (Clark and Curran, 2007). Table 3 shows both the standard beam levels (`AST`) used for the C&C parser and looser beam levels: `AST-covA`, a simple extension of `AST` with increased coverage and `AST-covB`, also increasing coverage but with better performance for the `HWDep` model.

Parsing results for the AST settings (Tables 4 and 5) confirm that it improves speed by an order of magnitude over a baseline parser without AST. Perhaps surprisingly, the number of parse failures decreases with AST in some cases. This is because the parser prunes more aggressively as the search space increases.[5]

[1] Indeed, all of the past work on A* parsing that we are aware of uses generative parsers (Pauls and Klein, 2009b, *inter alia*).

[2] http://code.google.com/p/statopenccg

[3] Less than 2% of supertags are affected by this.

[4] The timing results reported differ from an earlier draft since we used a different machine

[5] Hockenmaier and Steedman (2002) saw a similar effect.

Expansion probability	$p(exp\|P)$	exp $\in$ {leaf, unary, left-head, right-head}
Head probability	$p(H\|P, exp)$	H is the head daughter
Non-head probability	$p(S\|P, exp, H)$	S is the non-head daughter
Lexical probability	$p(w\|P)$	

Table 1: Factorisation of the PCFG model. H,P, and S are categories, and w is a word.

Expansion probability	$p(exp\|P, c_P\#w_P)$	exp $\in$ {leaf, unary, left-head, right-head}
Head probability	$p(H\|P, exp, c_P\#w_P)$	H is the head daughter
Non-head probability	$p(S\|P, exp, H\#c_P\#w_P)$	S is the non-head daughter
Lexcat probability	$p(c_S\|S\#P, H, S)$	$p(c_{TOP}\|P{=}TOP)$
Headword probability	$p(w_S\|c_S\#P, H, S, w_P)$	$p(w_{TOP}\|c_{TOP})$

Table 2: Headword dependency model factorisation, backoff levels are denoted by '#' between conditioning variables: A # B # C indicates that $\hat{P}(\ldots|A, B, C)$ is interpolated with $\hat{P}(\ldots|A, B)$, which is an interpolation of $\hat{P}\ldots|A, B)$ and $\hat{P}(\ldots|A)$. Variables c_P and w_P represent, respectively, the head lexical category and headword of category P.

Condition	Parameter	Iteration 1	2	3	4	5	6
AST	β (beam width)	0.075	0.03	0.01	0.005	0.001	
	k (dictionary cutoff)	20	20	20	20	150	
AST-covA	β	0.075	0.03	0.01	0.005	0.001	0.0001
	k	20	20	20	20	150	150
AST-covB	β	0.03	0.01	0.005	0.001	0.0001	0.0001
	k	20	20	20	20	20	150

Table 3: Beam step function used for standard (AST) and high-coverage (AST-covA and AST-covB) supertagging.

	Time(sec)	Sent/sec	Cats/word	Fail	UP	UR	UF	LP	LR	LF
PCFG	290	6.6	26.2	5	86.4	86.5	86.5	77.2	77.3	77.3
PCFG (AST)	**65**	**29.5**	1.5	14	**87.4**	85.9	86.6	**79.5**	78.0	78.8
PCFG (AST-covA)	67	28.6	1.5	6	87.3	**86.9**	**87.1**	79.1	**78.8**	**78.9**
PCFG (AST-covB)	69	27.7	1.7	5	87.3	86.2	86.7	79.1	78.1	78.6
HWDep	1512	1.3	26.2	5	**90.2**	**90.1**	**90.2**	**83.2**	**83.0**	**83.1**
HWDep (AST)	**133**	**14.4**	1.5	16	89.8	88.0	88.9	82.6	80.9	81.8
HWDep (AST-covA)	139	13.7	1.5	9	89.8	88.3	89.0	82.6	81.1	81.9
HWDep (AST-covB)	155	12.3	1.7	7	90.1	88.7	89.4	83.0	81.8	82.4

Table 4: Results on CCGbank section 00 when applying adaptive supertagging (AST) to two models of a generative CCG parser. Performance is measured in terms of parse failures, labelled and unlabelled precision (LP/UP), recall (LR/UR) and F-score (LF/UF). Evaluation is based only on sentences for which each parser returned an analysis.

3.2 Efficiency versus Accuracy

The most interesting result is the effect of the speedup on accuracy. As shown in Table 6, the vast majority of sentences are actually parsed with a very tight supertagger beam, raising the question of whether many higher-scoring parses are pruned.[6] Despite this, labeled F-score improves by up to 1.6 F-measure for the PCFG model, although it harms accuracy for HWDep as expected.

In order to understand this effect, we filtered section 00 to include only sentences of between 18 and 26 words (resulting in 610 sentences) for which

[6]Similar results are reported by Clark and Curran (2007).

	Time(sec)	Sent/sec	Cats/word	Fail	UP	UR	UF	LP	LR	LF
PCFG	326	7.4	25.7	29	85.9	85.4	85.7	76.6	76.2	76.4
PCFG (AST)	**82**	**29.4**	1.5	34	**86.7**	84.8	85.7	**78.6**	76.9	77.7
PCFG (AST-covA)	85	28.3	1.6	15	86.6	85.5	86.0	78.5	**77.5**	**78.0**
PCFG (AST-covB)	86	27.9	1.7	14	86.6	**85.6**	**86.1**	78.1	77.3	77.7
HWDep	1754	1.4	25.7	30	**90.2**	**89.3**	**89.8**	**83.5**	**82.7**	**83.1**
HWDep (AST)	**167**	**14.4**	1.5	27	89.5	87.6	88.5	82.3	80.6	81.5
HWDep (AST-covA)	177	13.6	1.6	14	89.4	88.1	88.8	82.2	81.1	81.7
HWDep (AST-covB)	188	12.8	1.7	14	89.7	88.5	89.1	82.5	81.4	82.0

Table 5: Results on CCGbank section 23 when applying adaptive supertagging (AST) to two models of a CCG parser.

β	Cats/word	Parses	%
0.075	1.33	1676	87.6
0.03	1.56	114	6.0
0.01	1.97	60	3.1
0.005	2.36	15	0.8
$0.001_{k=150}$	3.84	32	1.7
Fail		16	0.9

Table 6: Breakdown of the number of sentences parsed for the HWDep (AST) model (see Table 4) at each of the supertagger beam levels from the most to the least restrictive setting.

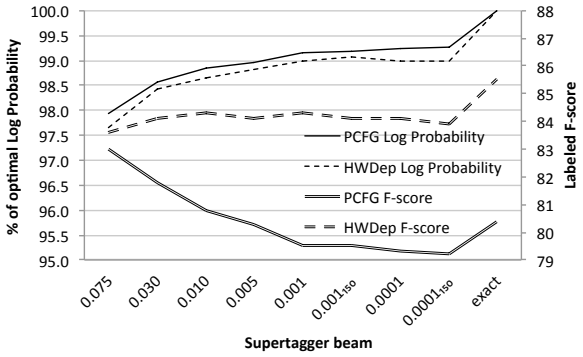


Figure 2: Log-probability of parses relative to exact solution vs. labelled F-score at each supertagging beam-level.

we can perform exhaustive search without pruning[7], and for which we could parse without failure at all of the tested beam settings. We then measured the log probability of the highest probability parse found under a variety of beam settings, relative to the log probability of the unpruned exact parse, along with the labeled F-Score of the Viterbi parse under these settings (Figure 2). The results show that PCFG actually finds worse results as it considers more of the search space. In other words, the supertagger can actually "fix" a bad parsing model by restricting it to a small portion of the search space. With the more accurate HWDep model, this does not appear to be a problem and there is a clear opportunity for improvement by considering the larger search space. The next question is whether we can exploit this larger search space without paying as high a cost in efficiency.

[7]The fact that only a subset of short sentences could be exhaustively parsed demonstrates the need for efficient search algorithms.

4 A* Parsing Experiments

To compare approaches, we extended our baseline parser to support A* search. Following (Klein and Manning, 2003) we restrict our experiments to sentences on which we can perform exact search via using the same subset of section 00 as in §3.2. Before considering CPU time, we first evaluate the amount of work done by the parser using three hardware-independent metrics. We measure the number of *edges pushed* (Pauls and Klein, 2009a) and *edges popped*, corresponding to the insert/decrease-key operations and remove operation of the priority queue, respectively. Finally, we measure the number of *traversals*, which counts the number of edge weights computed, regardless of whether the weight is discarded due to the prior existence of a better weight. This latter metric seems to be the most accurate account of the work done by the parser.

Due to differences in the PCFG and HWDep models, we considered different A* variants: for the PCFG model we use a simple A* with a precom-

puted heuristic, while for the the more complex HWDep model, we used a hierarchical A* algorithm (Pauls and Klein, 2009a; Felzenszwalb and McAllester, 2007) based on a simple grammar projection that we designed.

4.1 Hardware-Independent Results: PCFG

For the PCFG model, we compared three agenda-based parsers: EXH prioritizes edges by their span length, thereby simulating the exhaustive CKY algorithm; NULL prioritizes edges by their inside probability; and SX is an A* parser that prioritizes edges by their inside probability times an admissible outside probability estimate.[8] We use the SX estimate devised by Klein and Manning (2003) for CFG parsing, where they found it offered very good performance for relatively little computation. It gives a bound on the outside probability of a nonterminal P with i words to the right and j words to the left, and can be computed from a grammar using a simple dynamic program.

The parsers are tested with and without adaptive supertagging where the former can be seen as performing exact search (via A*) over the pruned search space created by AST.

Table 7 shows that A* with the SX heuristic decreases the number of edges pushed by up to 39% on the unpruned search space. Although encouraging, this is not as impressive as the 95% speedup obtained by Klein and Manning (2003) with this heuristic on their CFG. On the other hand, the NULL heuristic works better for CCG than for CFG, with speedups of 29% and 11%, respectively. These results carry over to the AST setting which shows that A* can improve search even on the highly pruned search graph. Note that A* only saves work in the final iteration of AST, since for earlier iterations it must process the entire agenda to determine that there is no spanning analysis.

Since there are many more categories in the CCG grammar we might have expected the SX heuristic to work better than for a CFG. Why doesn't it? We can measure how well a heuristic bounds the true cost in terms of *slack*: the difference between the true and estimated outside cost. Lower slack means that the heuristic bounds the true cost better and guides us to the exact solution more quickly. Figure 3 plots the average slack for the SX heuristic against the number of words in the outside context. Comparing this with an analysis of the same heuristic when applied to a CFG by Klein and Manning (2003), we find that it is less effective in our setting[9]. There is a steep increase in slack for outside contexts with size more than one. The main reason for this is because a single word in the outside context is in many cases the full stop at the end of the sentence, which is very predictable. However for longer spans the flexibility of CCG to analyze spans in many different ways means that the outside estimate for a nonterminal can be based on many high probability outside derivations which do not bound the true probability very well.

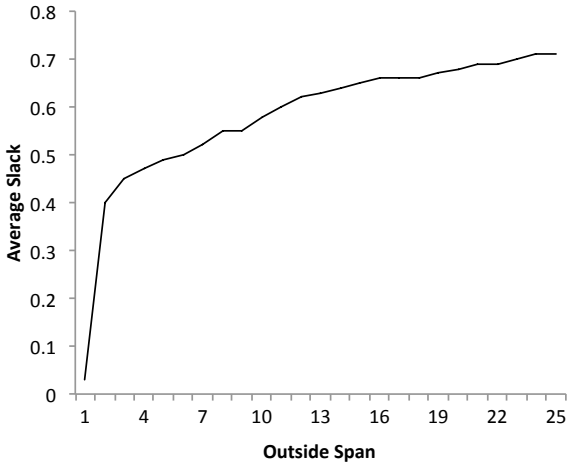


Figure 3: Average slack of the SX heuristic. The figure aggregates the ratio of the difference between the estimated outside cost and true outside costs relative to the true cost across the development set.

4.2 Hardware-Independent Results: HWDep

Lexicalization in the HWDep model makes the precomputed SX estimate impractical, so for this model we designed two hierarchical A* (HA*) variants based on simple grammar projections of the model. The basic idea of HA* is to compute Viterbi inside probabilities using the easier-to-parse projected

[8] The NULL parser is a special case of A*, also called uniform cost search, which in the case of parsing corresponds to Knuth's algorithm (Knuth, 1977; Klein and Manning, 2001), the extension of Dijkstra's algorithm to hypergraphs.

[9] Specifically, we refer to Figure 9 of their paper which uses a slightly different representation of estimate sharpness

Parser	Edges pushed				Edges popped				Traversals			
	Std	%	AST	%	Std	%	AST	%	Std	%	AST	%
EXH	34	100	6.3	100	15.7	100	4.2	100	133.4	100	13.3	100
NULL	24.3	71	4.9	78	13.5	86	3.5	83	113.8	85	11.1	83
SX	**20.9**	**61**	**4.3**	**68**	**10.0**	**64**	**2.6**	**62**	**96.5**	**72**	**9.7**	**73**

Table 7: Exhaustive search (EXH), A* with no heuristic (NULL) and with the SX heuristic in terms of millions of edges pushed, popped and traversals computed using the PCFG grammar with and without adaptive supertagging.

grammar, use these to compute Viterbi outside probabilities for the simple grammar, and then use these as outside estimates for the true grammar; all computations are prioritized in a single agenda following the algorithm of Felzenszwalb and McAllester (2007) and Pauls and Klein (2009a). We designed two simple grammar projections, each simplifying the HWDep model: PCFGProj completely removes lexicalization and projects the grammar to a PCFG, while as LexcatProj removes only the headwords but retains the lexical categories.

Figure 4 compares exhaustive search, A* with no heuristic (NULL), and HA*. For HA*, parsing effort is broken down into the different edge types computed at each stage: We distinguish between the work carried out to compute the inside and outside edges of the projection, where the latter represent the heuristic estimates, and finally, the work to compute the edges of the target grammar. We find that A* NULL saves about 44% of edges pushed which makes it slightly more effective than for the PCFG model. However, the effort to compute the grammar projections outweighs their benefit. We suspect that this is due to the large difference between the target grammar and the projection: The PCFG projection is a simple grammar and so we improve the probability of a traversal less often than in the target grammar.

The Lexcat projection performs worst, for two reasons. First, the projection requires about as much work to compute as the target grammar without a heuristic (NULL). Second, the projection itself does not save a large amount of work as can be seen in the statistics for the target grammar.

5 CPU Timing Experiments

Hardware-independent metrics are useful for understanding agenda-based algorithms, but what we actually care about is CPU time. We were not aware of any past work that measures A* parsers in terms of CPU time, but as this is the real objective we feel that experiments of this type are important. This is especially true in real implementations because the savings in edges processed by an agenda parser come at a cost: operations on the priority queue data structure can add significant runtime.

Timing experiments of this type are very implementation-dependent, so we took care to implement the algorithms as cleanly as possible and to reuse as much of the existing parser code as we could. An important implementation decision for agenda-based algorithms is the data structure used to implement the priority queue. Preliminary experiments showed that a Fibonacci heap implementation outperformed several alternatives: Brodal queues (Brodal, 1996), binary heaps, binomial heaps, and pairing heaps.[10]

We carried out timing experiments on the best A* parsers for each model (SX and NULL for PCFG and HWDep, respectively), comparing them with our CKY implementation and the agenda-based CKY simulation EXH; we used the same data as in §3.2. Table 8 presents the cumulative running times with and without adaptive supertagging average over ten runs, while Table 9 reports F-scores.

The results (Table 8) are striking. Although the timing results of the agenda-based parsers track the hardware-independent metrics, they start at a significant disadvantage to exhaustive CKY with a simple control loop. This is most evident when looking at the timing results for EXH, which in the case of the full PCFG model requires more than twice the time than the CKY algorithm that it simulates. A*

[10] We used the Fibonacci heap implementation at http://www.jgrapht.org

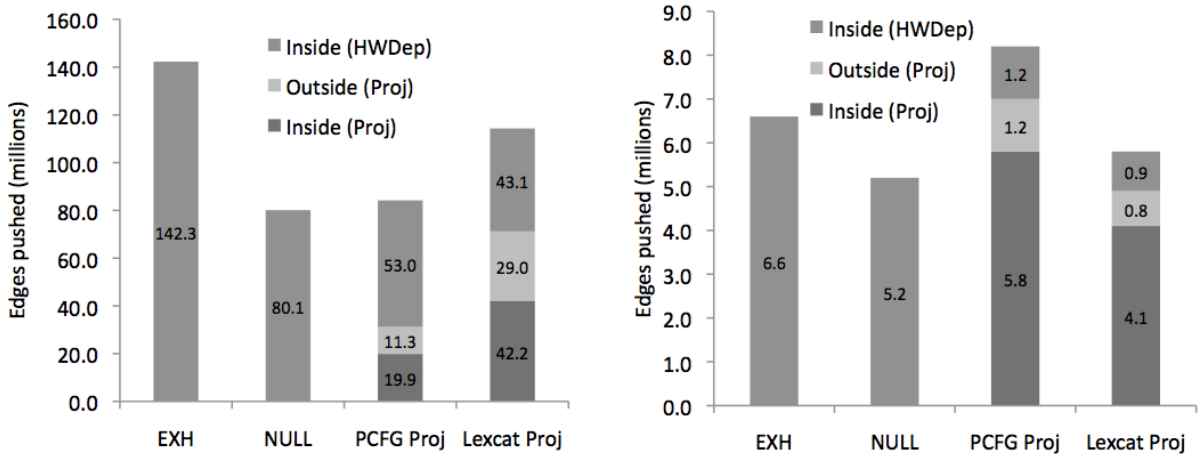


Figure 4: Comparsion between a CKY simulation (EXH), A* with no heuristic (NULL), hierarchical A* (HA*) using two grammar projections for standard search (left) and AST (right). The breakdown of the inside/outside edges for the grammar projection as well as the target grammar shows that the projections, serving as the heuristic estimates for the target grammar, are costly to compute.

	Standard		AST	
	PCFG	HWDep	PCFG	HWDep
CKY	536	24489	34	143
EXH	1251	26889	41	155
A* NULL	1032	21830	36	121
A* SX	889	-	34	-

Table 8: Parsing time in seconds of CKY and agenda-based parsers with and without adaptive supertagging.

	Standard		AST	
	PCFG	HWDep	PCFG	HWDep
CKY	80.4	85.5	81.7	83.8
EXH	79.4	85.5	80.3	83.8
A* NULL	79.6	85.5	80.7	83.8
A* SX	79.4	-	80.4	-

Table 9: Labelled F-score of exact CKY and agenda-based parsers with/without adaptive supertagging. All parses have the same probabilities, thus variances are due to implementation-dependent differences in tiebreaking.

makes modest CPU-time improvements in parsing the full space of the HWDep model. Although this decreases the time required to obtain the highest accuracy, it is still a substantial tradeoff in speed compared with AST.

On the other hand, the AST tradeoff improves significantly: by combining AST with A* we observe a decrease in running time of 15% for the A* NULL parser of the HWDep model over CKY. As the CKY baseline with AST is very strong, this result shows that A* holds real promise for CCG parsing.

6 Conclusion and Future Work

Adaptive supertagging is a strong technique for efficient CCG parsing. Our analysis confirms tremendous speedups, and shows that for weak models, it can even result in improved accuracy. However, for better models, the efficiency gains of adaptive supertagging come at the cost of accuracy. One way to look at this is that the supertagger has good precision with respect to the parser's search space, but low recall. For instance, we might combine both parsing and supertagging models in a principled way to exploit these observations, eg. by making the supertagger output a soft constraint on the parser rather than a hard constraint. Principled, efficient search algorithms will be crucial to such an approach.

To our knowledge, we are the first to measure A* parsing speed both in terms of running time and commonly used hardware-independent metrics. It is clear from our results that the gains from A* do not come as easily for CCG as for CFG, and that agenda-based algorithms like A* must make very large reductions in the number of edges processed to result in realtime savings, due to the added expense of keeping a priority queue. However, we

have shown that A* can yield real improvements even over the highly optimized technique of adaptive supertagging: in this pruned search space, a 44% reduction in the number of edges pushed results in a 15% speedup in CPU time. Furthermore, just as A* can be combined with adaptive supertagging, it should also combine easily with other search-space pruning methods, such as those of Djordjevic et al. (2007), Kummerfeld et al. (2010), Zhang et al. (2010) and Roark and Hollingshead (2009). In future work we plan to examine better A* heuristics for CCG, and to look at principled approaches to combine the strengths of A*, adaptive supertagging, and other techniques to the best advantage.

Acknowledgements

We would like to thank Prachya Boonkwan, Juri Ganitkevitch, Philipp Koehn, Tom Kwiatkowski, Matt Post, Mark Steedman, Emily Thomforde, and Luke Zettlemoyer for helpful discussion related to this work and comments on previous drafts; Julia Hockenmaier for furnishing us with her parser; and the anonymous reviewers for helpful commentary. We also acknowledge funding from EPSRC grant EP/P504171/1 (Auli); the EuroMatrixPlus project funded by the European Commission, 7th Framework Programme (Lopez); and the resources provided by the Edinburgh Compute and Data Facility (`http://www.ecdf.ed.ac.uk/`). The ECDF is partially supported by the eDIKT initiative (`http://www.edikt.org.uk/`).

References

S. Bangalore and A. K. Joshi. 1999. Supertagging: An Approach to Almost Parsing. *Computational Linguistics*, 25(2):238–265, June.

G. S. Brodal. 1996. Worst-case efficient priority queues. In *Proc. of SODA*, pages 52–58.

S. Clark and J. R. Curran. 2004. The importance of supertagging for wide-coverage CCG parsing. In *Proc. of COLING*.

S. Clark and J. R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

S. Clark and J. Hockenmaier. 2002. Evaluating a wide-coverage CCG parser. In *Proc. of LREC Beyond Parseval Workshop*, pages 60–66.

S. Clark. 2002. Supertagging for Combinatory Categorial Grammar. In *Proc. of TAG+6*, pages 19–24.

B. Djordjevic, J. R. Curran, and S. Clark. 2007. Improving the efficiency of a wide-coverage CCG parser. In *Proc. of IWPT*.

P. F. Felzenszwalb and D. McAllester. 2007. The Generalized A* Architecture. In *Journal of Artificial Intelligence Research*, volume 29, pages 153–190.

T. A. D. Fowler and G. Penn. 2010. Accurate context-free parsing with combinatory categorial grammar. In *Proc. of ACL*.

P. Hart, N. Nilsson, and B. Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *Transactions on Systems Science and Cybernetics*, 4, Jul.

J. Hockenmaier and M. Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proc. of ACL*, pages 335–342. Association for Computational Linguistics.

J. Hockenmaier and M. Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

D. Klein and C. D. Manning. 2001. Parsing and hypergraphs. In *Proc. of IWPT*.

D. Klein and C. D. Manning. 2003. A* parsing: Fast exact Viterbi parse selection. In *Proc. of HLT-NAACL*, pages 119–126, May.

D. E. Knuth. 1977. A generalization of Dijkstra's algorithm. *Information Processing Letters*, 6:1–5.

J. K. Kummerfeld, J. Roesner, T. Dawborn, J. Haggerty, J. R. Curran, and S. Clark. 2010. Faster parsing by supertagger adaptation. In *Proc. of ACL*.

A. Pauls and D. Klein. 2009a. Hierarchical search for parsing. In *Proc. of HLT-NAACL*, pages 557–565, June.

A. Pauls and D. Klein. 2009b. k-best A* Parsing. In *Proc. of ACL-IJCNLP*, ACL-IJCNLP '09, pages 958–966.

B. Roark and K. Hollingshead. 2009. Linear complexity context-free parsing pipelines via chart constraints. In *Proc. of HLT-NAACL*.

M. Steedman. 2000. *The syntactic process*. MIT Press, Cambridge, MA.

Y. Zhang, B.-G. Ahn, S. Clark, C. V. Wyk, J. R. Curran, and L. Rimell. 2010. Chart pruning for fast lexicalised-grammar parsing. In *Proc. of COLING*.

Improving Arabic Dependency Parsing with Form-based and Functional Morphological Features

Yuval Marton
T.J. Watson Research Center
IBM
yymarton@us.ibm.com

Nizar Habash and **Owen Rambow**
Center for Computational Learning Systems
Columbia University
{habash,rambow}@ccls.columbia.edu

Abstract

We explore the contribution of morphological features – both lexical and inflectional – to dependency parsing of Arabic, a morphologically rich language. Using controlled experiments, we find that definiteness, person, number, gender, and the undiacritzed lemma are most helpful for parsing on automatically tagged input. We further contrast the contribution of form-based and functional features, and show that functional gender and number (e.g., "broken plurals") and the related rationality feature improve over form-based features. It is the first time functional morphological features are used for Arabic NLP.

1 Introduction

Parsers need to learn the syntax of the modeled language in order to project structure on newly seen sentences. Parsing model design aims to come up with features that best help parsers to learn the syntax and choose among different parses. One aspect of syntax, which is often not explicitly modeled in parsing, involves morphological constraints on syntactic structure, such as agreement, which often plays an important role in morphologically rich languages. In this paper, we explore the role of morphological features in parsing Modern Standard Arabic (MSA). For MSA, the space of possible morphological features is fairly large. We determine which morphological features help and why. We also explore going beyond the easily detectable, regular form-based ("surface") features, by representing *functional* values for some morphological features. We expect that representing lexical abstractions and inflectional features participating in agreement relations would help parsing quality, but other inflectional features would not help. We further expect functional features to be superior to surface-only features.

The paper is structured as follows. We first present the corpus we use (Section 2), then relevant Arabic linguistic facts (Section 3); we survey related work (Section 4), describe our experiments (Section 5), and conclude with an analysis of parsing error types (Section 6).

2 Corpus

We use the Columbia Arabic Treebank (CATiB) (Habash and Roth, 2009). Specifically, we use the portion converted automatically from part 3 of the Penn Arabic Treebank (PATB) (Maamouri et al., 2004) to the CATiB format, which enriches the CATiB dependency trees with full PATB morphological information. CATiB's dependency representation is based on traditional Arabic grammar and emphasizes syntactic case relations. It has a reduced POS tagset (with six tags only – henceforth CATIB6), but a standard set of eight dependency relations: **SBJ** and **OBJ** for subject and (direct or indirect) object, respectively, (whether they appear pre- or post-verbally); **IDF** for the idafa (possessive) relation; **MOD** for most other modifications; and other less common relations that we will not discuss here. For more information, see Habash et al. (2009). The CATiB treebank uses the word segmentation of the PATB: it splits off several categories of orthographic clitics, but not the definite article ال *Al*. In all of the experiments reported in this paper, we use the gold

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1586–1596,
Portland, Oregon, June 19-24, 2011.

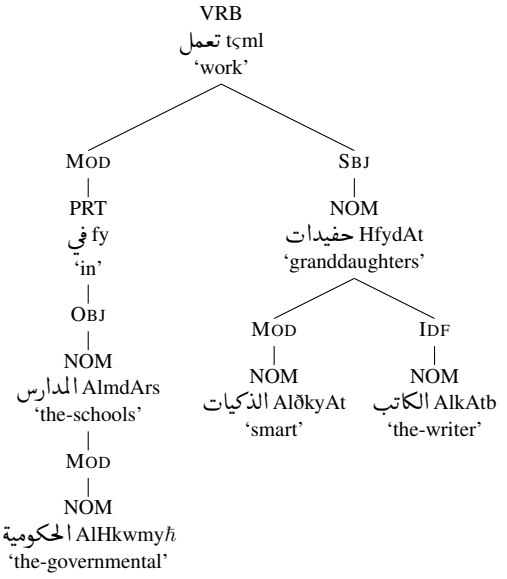


Figure 1: CATiB Annotation example (tree display from right to left). تعمل حفيدات الكاتب الذكيات في المدارس الحكومية *tçml HfydAt AlkAtb AlðkyAt fy AlmdArs AlHkwmyħ* 'The writer's smart granddaughters work for public schools.'

segmentation. An example CATiB dependency tree is shown in Figure 1.

3 Relevant Linguistic Concepts

In this section, we present the linguistic concepts relevant to our discussion of Arabic parsing.

Orthography The Arabic script uses optional diacritics to represent short vowels, consonantal doubling and the indefininteness morpheme (*nunation*). For example, the word كَتَبَ *kataba* 'he wrote' is often written as كتب *ktb*, which can be ambiguous with other words such as كُتُبٌ *kutubũ* 'books'. In news text, only around 1.6% of all words have any diacritic (Habash, 2010). As expected, the lack of diacritics contributes heavily to Arabic's morphological ambiguity. In this work, we only use undiacritized text; however, some of our parsing features which are derived through morphological disambiguation include diacritics (specifically, *lemmas*, see below).

Morphemes Words can be described in terms of their morphemes; in Arabic, in addition to concatenative prefixes and suffixes, there are templatic morphemes called *root* and *pattern*. For example, the word يكاتبون *yu+kAtib+uwn* 'they correspond' has one prefix and one suffix, in addition to a stem composed of the root كتب *k-t-b* 'writing related' and the pattern *1A2i3*.[1]

Lexeme and features Alternatively, Arabic words can be described in terms of lexemes and inflectional features. The set of word forms that only vary inflectionally among each other is called the *lexeme*. A *lemma* is a specific word form chosen to represent the lexeme word set; for example, Arabic verb lemmas are third person masculine singular perfective. We explore using both the diacritized lemma and the undiacritized lemma (hereafter LMM). Just as the lemma abstracts over inflectional morphology, the *root* abstracts over both inflectional and derivational morphology and thus provides a deeper level of lexical abstraction, indicating the "core" meaning of the word. The *pattern* is a generally complementary abstraction sometimes indicating sematic notions such causation and reflexiveness. We use the pattern of the lemma, not of the word form. We group the ROOT, PATTERN, LEMMA and LMM in our discussion as *lexical features*. Nominal lexemes can also be classified into two groups: rational (i.e., human) or irrational (i.e.,, non-human).[2] The rationality feature interacts with syntactic agreement and other inflectional features (discussed next); as such, we group it with those features in this paper's experiments.

The *inflectional features* define the the space of variations of the word forms associated with a lexeme. PATB-tokenized words vary along nine dimensions: GENDER and NUMBER (for nominals and verbs); PERSON, ASPECT, VOICE and MOOD (for verbs); and CASE, STATE, and the attached definite article proclitic DET (for nominals). Inflectional features abstract away from the specifics of morpheme forms. Some inflectional features affect more than one morpheme in the same word. For example, changing the value of the ASPECT feature in the example above from imperfective to perfective yields the word form كاتبوا *kAtab+uwA* 'they corresponded', which differs in terms of prefix, suffix and pattern.

[1]The digits in the pattern correspond to the positions root radicals are inserted.

[2]Note that rationality ('human-ness' 'عاقل/غير عاقل') is narrower than animacy; its expression is wide-spead in Arabic, but less so English, where it mainly shows in pronouns (*he/she* vs. *it*) and relativizers (*the student who...* vs. *the desk/bird which...*).

Surface vs. functional features Additionally, some inflectional features, specifically gender and number, are expressed using different morphemes in different words (even within the same part-of-speech). There are four *sound* gender-number suffixes in Arabic:[3] $+\phi$ *(null morpheme)* for masculine singular, ة+ $+\hbar$ for feminine singular, ون+ *+wn* for masculine plural and ات+ *+At* for feminine plural. Plurality can be expressed using *sound plural* suffixes or using a pattern change together with *singular* suffixes. A sound plural example is the word pair حفيدة/حفيدات *Hafiyd+aħ/Hafiyd+At* 'granddaughter/granddaughters'. On the other hand, the plural of the inflectionally and morphemically feminine singular word مدرسة *madras+aħ* 'school' is the word مدارس *madAris+*ϕ 'schools', which is feminine and plural inflectionally, but has a masculine singular suffix. This irregular inflection, known as *broken plural*, is similar to the English *mouse/mice*, but is much more common in Arabic (over 50% of plurals in our training data). A similar inconsistency appears in feminine nouns that are not inflected using *sound* gender suffixes, e.g., the feminine form of the masculine singular adjective أزرق *Âzraq+*ϕ 'blue' is زرقاء *zarqA'+*ϕ not أزرقة* **Âzraq+aħ*. To address this inconsistency in the correspondence between inflectional features and morphemes, and inspired by (Smrž, 2007), we distinguish between two types of inflectional features: surface (or form-based)[4] features and functional features.

Most available Arabic NLP tools and resources model morphology using surface inflectional features and do not mark rationality; this includes the PATB (Maamouri et al., 2004), the Buckwalter morphological analyzer (BAMA) (Buckwalter, 2004) and tools using them such as the Morphological Analysis and Disambiguation for Arabic (MADA) system (Habash and Rambow, 2005). The Elixir-FM analyzer (Smrž, 2007) readily provides the functional inflectional number feature, but not full functional gender (only for adjectives and verbs but not for nouns), nor rationality. Most recently, Alkuhlani and Habash (2011) present a version of the PATB (part 3) that is annotated for functional gender, number and rationality features for Arabic. We use this resource in modeling these features in Section 5.5.

Morpho-syntactic interactions Inflectional features and rationality interact with syntax in two ways. In **agreement relations**, two words in a specific syntactic configuration have coordinated values for specific sets of features. MSA has standard (i.e., matching value) agreement for subject-verb pairs on PERSON, GENDER, and NUMBER, and for noun-adjective pairs on NUMBER, GENDER, CASE, and DET. There are three very common cases of exceptional agreement: verbs preceding subjects are always singular, adjectives of irrational plural nouns are always feminine singular, and verbs whose subjects are irrational plural are also always feminine singular. See the example in Figure 1: the adjective, الذكيات *Alðkyat* 'smart', of the feminine plural (and rational) حفيدات *HafiydAt* 'granddaughters' is feminine plural; but the adjective, الحكومية *AlHkwmyħ* 'the-governmental', of the feminine plural (and irrational) مدارس *madAris* 'schools' is feminine singular. These agreement rules always refer to functional morphology categories; they are orthogonal to the morpheme-feature inconsistency discussed above.

MSA exhibits **marking relations** in CASE and STATE marking. Different types of dependents have different CASE, e.g., verbal subjects are always marked NOMINATIVE. CASE and STATE are rarely explicitly manifested in undiacritized MSA. The DET feature plays an important role in distinguishing between the N-N *idafa* (possessive) construction, in which only the last noun may bear the definite article, and the N-A modifier construction, in which both elements generally exhibit agreement in definiteness.

Lexical features do not constrain syntactic structure as inflectional features do. Instead, bilexical dependencies are used to model semantic relations which often are the only way to disambiguate among different possible syntactic structures. Lexical abstraction also reduces data sparseness.

The core POS tagsets Words also have associated part-of-speech (POS) tags, e.g., "verb", which further abstract over morphologically and syntactically similar lexemes. Traditional Arabic grammars often describe a very general three-way distinction into verbs, nominals and particles. In com-

[3] We ignore duals, which are regular in Arabic, and case/state variations in this discussion for simplicity.

[4] Smrž (2007) uses the term *illusory* for surface features.

parison, the tagset of the Buckwalter Morphological Analyzer (Buckwalter, 2004) used in the PATB has a core POS set of 44 tags (before morphological extension). Cross-linguistically, a core set containing around 12 tags is often assumed, including: noun, proper noun, verb, adjective, adverb, preposition, particles, connectives, and punctuation. Henceforth, we reduce CORE44 to such a tagset, and dub it CORE12. The CATIB6 tagset can be viewed as a further reduction, with the exception that CATIB6 contains a passive voice tag; however, this tag constitutes only 0.5% of the tags in the training.

Extended POS tagsets The notion of "POS tagset" in natural language processing usually does *not* refer to a core set. Instead, the Penn English Treebank (PTB) uses a set of 46 tags, including not only the core POS, but also the complete set of morphological features (this tagset is still fairly small since English is morphologically impoverished). In PATB-tokenized MSA, the corresponding type of tagset (core POS extended with a complete description of morphology) would contain upwards of 2,000 tags, many of which are extremely rare (in our training corpus of about 300,000 words, we encounter only 430 of such POS tags with complete morphology). Therefore, researchers have proposed tagsets for MSA whose size is similar to that of the English PTB tagset, as this has proven to be a useful size computationally. These tagsets are hybrids in the sense that they are neither simply the core POS, nor the complete morphologically enriched tagset, but instead they selectively enrich the core POS tagset with only certain morphological features. A full dicussion of how these tagsets affect parsing is presented in Marton et al. (2010); we summarize the main points here.

The following are the various tagsets we use in this paper: **(a)** the core POS tagset CORE12; **(b)** the CATiB treebank tagset CATIBEX, a newly introduced extension of CATIB6 (Habash and Roth, 2009) by simple regular expressions of the word form, indicating particular morphemes such as the prefix ال *Al+* or the suffix ون *+wn*; this tagset is the best-performing tagset for Arabic on predicted values. **(c)** the PATB full tagset (BW), size ≈2000+ (Buckwalter, 2004); We only discuss here the best performing tagsets (on predicted values), and BW for comparison.

4 Related Work

Much work has been done on the use of morphological features for parsing of morphologically rich languages. Collins et al. (1999) report that an optimal tagset for parsing Czech consists of a basic POS tag plus a CASE feature (when applicable). This tagset (size 58) outperforms the basic Czech POS tagset (size 13) and the complete tagset (size ≈3000+). They also report that the use of gender, number and person features did not yield any improvements. We got similar results for CASE in the gold experimental setting (Marton et al., 2010) but not when using predicted POS tags (POS tagger output). This may be a result of CASE tagging having a lower error rate in Czech (5.0%) (Hajič and Vidová-Hladká, 1998) compared to Arabic (≈14.0%, see Table 2). Similarly, Cowan and Collins (2005) report that the use of a subset of Spanish morphological features (number for adjectives, determiners, nouns, pronouns, and verbs; and mode for verbs) outperforms other combinations. Our approach is comparable to their work in terms of its systematic exploration of the space of morphological features. We also find that the number feature helps for Arabic. Looking at Hebrew, a Semitic language related to Arabic, Tsarfaty and Sima'an (2007) report that extending POS and phrase structure tags with definiteness information helps unlexicalized PCFG parsing.

As for work on Arabic, results have been reported on PATB (Kulick et al., 2006; Diab, 2007; Green and Manning, 2010), the Prague Dependency Treebank (PADT) (Buchholz and Marsi, 2006; Nivre, 2008) and the Columbia Arabic Treebank (CATiB) (Habash and Roth, 2009). Recently, Green and Manning (2010) analyzed the PATB for annotation consistency, and introduced an enhanced split-state constituency grammar, including labels for short *Idafa* constructions and verbal or equational clauses.

Nivre (2008) reports experiments on Arabic parsing using his MaltParser (Nivre et al., 2007), trained on the PADT. His results are not directly comparable to ours because of the different treebanks' representations, even though all the experiments reported here were performed using MaltParser. Our results agree with previous work on Arabic and Hebrew in that marking the definite article is helpful for parsing. However, we go beyond previous work in that

we also extend this morphologically enhanced feature set to include additional lexical and inflectional features. Previous work with MaltParser in Russian, Turkish and Hindi showed gains with case but not with agreement features (Nivre et al., 2008; Eryigit et al., 2008; Nivre, 2009). Our work is the first using MaltParser to show gains using agreement-oriented features (Marton et al., 2010), and the first to use functional features for this task (this paper).

5 Experiments

Throughout this section, we only report results using *predicted* input feature values (e.g., generated automatically by a POS tagger). After presenting the parser we use (Section 5.1), we examine a large space of settings in the following order: the contribution of numerous inflectional features in a controlled fashion (Section 5.2);[5] the contribution of the lexical features in a similar fashion, as well as the combination of lexical and inflectional features (Section 5.3); an extension of the DET feature (Section 5.4); using functional NUMBER and GENDER feature values, as well as the RATIONALITY feature (Section 5.5); finally, putting best feature combinations to test with the best-performing POS tagset, and on an unseen test set (Section 5.6). All results are reported mainly in terms of labeled attachment accuracy score (parent word and the dependency relation to it, a.k.a. LAS). Unlabeled attachment accuracy score (UAS) is also given. We use McNemar's statistical significance test as implemented by Nilsson and Nivre (2008), and denote $p < 0.05$ and $p < 0.01$ with $^{+}$ and $^{++}$, respectively.

5.1 Parser

For all experiments reported here we used the syntactic dependency parser MaltParser v1.3 (Nivre, 2003; Nivre, 2008; Kübler et al., 2009) – a transition-based parser with an input buffer and a stack, using SVM classifiers to predict the next state in the parse derivation. All experiments were done using the Nivre "eager" algorithm.[6] For training, development and testing, we follow the splits used by Roth et al. (2008) for PATB part 3 (Maamouri et al., 2004). We kept the test unseen during training.

There are five default *attributes*, in the MaltParser terminology, for each token in the text: word ID (ordinal position in the sentence), word form, POS tag, head (parent word ID), and deprel (the dependency relation between the current word and its parent). There are default *MaltParser features* (in the machine learning sense),[7] which are the values of functions over these attributes, serving as input to the MaltParser internal classifiers. The most commonly used feature functions are the top of the input buffer (next word to process, denoted buf[0]), or top of the stack (denoted stk[0]); following items on buffer or stack are also accessible (buf[1], buf[2], stk[1], etc.). Hence MaltParser features are defined as POS tag at stk[0], word form at buf[0], etc. Kübler et al. (2009) describe a "typical" MaltParser model configuration of attributes and features.[8] Starting with it, in a series of initial controlled experiments, we settled on using buf[0-1] + stk[0-1] for wordforms, and buf[0-3] + stk[0-2] for POS tags. For features of new MaltParser-attributes (discussed later), we used buf[0] + stk[0]. We did not change the features for deprel. This new MaltParser configuration resulted in gains of 0.3-1.1% in labeled attachment accuracy (depending on the POS tagset) over the default MaltParser configuration.[9] All experiments reported below were conducted using this new configuration.

5.2 Inflectional features

In order to explore the contribution of inflectional and lexical information in a controlled manner, we focused on the best performing core ("morphology-free") POS tagset, CORE12, as baseline; using three

[5] In this paper, we do not examine the contribution of different POS tagsets, see Marton et al. (2010) for details.

[6] Nivre (2008) reports that non-projective and pseudo-projective algorithms outperform the "eager" projective algorithm in MaltParser, but our training data did not contain any non-projective dependencies. The Nivre "standard" algorithm is also reported to do better on Arabic, but in a preliminary experimentation, it did similarly or slightly worse than the "eager" one, perhaps due to high percentage of right branching (left headed structures) in our Arabic training set – an observation already noted in Nivre (2008).

[7] The terms "feature" and "attribute" are overloaded in the literature. We use them in the linguistic sense, unless specifically noted otherwise, e.g., "MaltParser feature(s)".

[8] It is slightly different from the default configuration.

[9] We also experimented with normalizing word forms (*Alif Maqsura* conversion to *Ya*, and *hamza* removal from *Alif* forms) as is common in parsing and statistical machine translation literature – but it resulted in a similar or slightly decreased performance, so we settled on using non-normalized word forms.

setup		LAS	LAS_{diff}	UAS
All	**CORE12**	78.68	—	82.48
	+ all inflectional features	77.91	-0.77	82.14
Sep	+DET	**79.82**$^{++}$	**1.14**	83.18
	+STATE	79.34^{++}	0.66	82.85
	+GENDER	78.75	0.07	82.35
	+PERSON	78.74	0.06	82.45
	+NUMBER	78.66	-0.02	82.39
	+VOICE	78.64	-0.04	82.41
	+ASPECT	78.60	-0.08	82.39
	+MOOD	78.54	-0.14	82.35
	+CASE	75.81	-2.87	80.24
Greedy	+DET+STATE	79.42^{++}	0.74	82.84
	+DET+GENDER	79.90^{++}	1.22	83.20
	+DET+GENDER+PERSON	79.94^{++}	1.26	83.21
	+DET+PNG	**80.11**$^{++}$	**1.43**	83.29
	+DET+PNG+VOICE	79.96^{++}	1.28	83.18
	+DET+PNG+ASPECT	80.01^{++}	1.33	83.20
	+DET+PNG+MOOD	80.03^{++}	1.35	83.21

Table 1: CORE12 with inflectional features, predicted input. Top: Adding all nine features to CORE12. Second part: Adding each feature separately, comparing difference from CORE12. Third part: Greedily adding best features from second part.

different setups, we added nine morphological features with values predicted by MADA: DET (presence of the definite determiner), PERSON, ASPECT, VOICE, MOOD, GENDER, NUMBER, STATE (morphological marking as head of an *idafa* construction), and CASE. In setup ***All***, we augmented the baseline model with all nine MADA features (as nine additional MaltParser attributes); in setup ***Sep***, we augmented the baseline model with the MADA features, one at a time; and in setup ***Greedy***, we combined them in a greedy heuristic (since the entire feature space is too vast to exhaust): starting with the most gainful feature from *Sep*, adding the next most gainful feature, keeping it if it helped, or discarding it otherwise, and continuing through the least gainful feature. See Table 1.

Somewhat surprisingly, setup *All* hurts performance. This can be explained if one examines the prediction accuracy of each feature (top of Table 2). Features which are not predicted with very high accuracy, such as CASE (86.3%), can dominate the negative contribution, even though they are top contributors when provided as gold input (Marton et al., 2010); when all features are provided as gold input, *All* actually does better than individual features, which puts to rest a concern that its decrease here

feature	acc	set size
DET	99.6	3*
PERSON	99.1	4*
ASPECT	99.1	5*
VOICE	98.9	4*
MOOD	98.6	5*
GENDER	99.3	3*
NUMBER	99.5	4*
STATE	95.6	4*
CASE	86.3	5*
ROOT	98.4	9646
PATTERN	97.0	338
LEMMA (diacritized)	96.7	16837
LMM (undiacritized lemma)	98.3	15305
normalized word form (A,Y)	99.3	29737
non-normalized word form	98.9	29980

Table 2: Feature prediction accuracy and set sizes. * = The set includes a "N/A" value.

setup		LAS	LAS_{diff}	UAS
All	**CORE12 (repeated)**	78.68	—	82.48
	+ all lexical features	78.85	0.17	82.46
Sep	+LMM	**78.96**$^{+}$	**0.28**	82.54
	+ROOT	78.94^{+}	0.26	82.64
	+LEMMA	78.80	0.12	82.42
	+PATTERN	78.59	-0.09	82.39
Greedy	+LMM+ROOT	79.04^{++}	0.36	82.63
	+LMM+ROOT+LEMMA	**79.05**$^{++}$	**0.37**	82.63
	+LMM+ROOT+PATTERN	78.93	0.25	82.58

Table 3: Lexical features. Top part: Adding each feature separately; difference from CORE12 (predicted). Bottom part: Greedily adding best features from previous part.

is due to data sparseness. Here, when features are predicted, the DET feature (determiner), followed by the STATE (construct state, *idafa*) feature, are top individual contributors in setup *Sep*. Adding DET and the so-called ϕ-features (PERSON, NUMBER, GENDER, also shorthanded PNG) in the *Greedy* setup, yields 1.43% gain over the CORE12 baseline.

5.3 Lexical features

Next, we experimented with adding the lexical features, which involve semantic abstraction to some degree: LEMMA, LMM (the undiacritized lemma), and ROOT. We experimented with the same setups as above: *All*, *Sep*, and *Greedy*. Adding all four features yielded a minor gain in setup *All*. LMM was the best single contributor, closely followed by ROOT in *Sep*. CORE12+LMM+ROOT (with or with-

CORE12 + ...	LAS	LAS_{diff}	UAS
+DET+PNG (repeated)	**80.11**$^{++}$	1.43	83.29
+DET+PNG+LMM	**80.23**$^{++}$	**1.55**	83.34
+DET+PNG+LMM +ROOT	80.10^{++}	1.42	83.25
+DET+PNG+LMM +PATTERN	80.03^{++}	1.35	83.15

Table 4: Inflectional+lexical features together.

CORE12 + ...	LAS	LAS_{diff}	UAS
+DET (repeated)	79.82^{++}	—	83.18
+DET2	**80.13**$^{++}$	0.31	83.49
+DET+PNG+LMM (repeated)	**80.23**$^{++}$	—	83.34
+DET2+PNG+LMM	80.21^{++}	-0.02	83.39

Table 5: Extended inflectional features.

out LEMMA) was the best greedy combination in setup *Greedy*. See Table 3. All lexical features are predicted with high accuracy (bottom of Table 2).

Following the same greedy heuristic, we augmented the best inflection-based model CORE12+DET+PNG with lexical features, and found that only the undiacritized lemma (LMM) alone improved performance (80.23%). See Table 4.

5.4 Inflectional feature engineering

So far we experimented with morphological feature values as predicted by MADA. However, it is likely that from a machine-learning perspective, representing similar categories with the same tag may be useful for learning. Therefore, we next experimented with modifying inflectional features that proved most useful.

As DET may help distinguish the N-N *idafa* construction from the N-A modifier construction, we attempted modeling also the DET values of previous and next elements (as MaltParser's stk[1] + buf[1], in addition to stk[0] + buf[0]). This variant, denoted DET2, indeed helps: when added to the CORE12, DET2 improves non-gold parsing quality by more than 0.3%, compared to DET (Table 5). This improvement unfortunately does not carry over to our best feature combination to date, CORE12+DET+PNG+LMM. However, in subsequent feature combinations, we see that DET2 helps again, or at least, doesn't hurt: LAS goes up by 0.06% in conjunction with features LMM+PERSON +FN*NGR in Table 6.

CORE12 + ...	LAS	LAS_{diff}	UAS
CORE12 (repeated)	78.68	–	82.48
+PERSON (repeated)	78.74	0.06	82.45
+GENDER (repeated)	78.75	0.07	82.35
+NUMBER (repeated)	78.66	-0.02	82.39
+FN*GENDER	78.96^{++}	0.28	82.53
+FN*NUMBER	78.88^{+}	0.20	82.53
+FN*NUMDGTBIN	78.87	0.19	82.53
+FN*RATIONALITY	78.91^{+}	0.23	82.60
+FN*GNR	79.32^{++}	0.64	82.78
+PERSON+FN*GNR	79.34^{++}	0.66	82.82
+DET+LMM+PERSON+FN*NGR	80.47^{++}	1.79	83.57
+DET2+LMM+PERSON+FN*NGR	**80.53**$^{++}$	1.85	83.66
+DET2+LMM+PERSON+FN*NG	80.43^{++}	1.75	83.56
+DET2+LMM+PNG+FN*NGR	80.51^{++}	1.83	83.66
CATIBEX	79.74	–	83.30
+DET2+LMM +PERSON+FN*NGR	**80.83**$^{++}$	1.09	84.02
BW	72.64	–	77.91
+DET2+LMM +PERSON+FN*NGR	**74.40**$^{++}$	1.76	79.40

Table 6: Functional features: gender, number, rationality.

We also experimented with PERSON. We changed the values of proper names from "N/A" to "3" (third person), but it resulted in a similar or slightly decreased performance, so it was abandoned.

5.5 Functional feature values

The NUMBER and GENDER features we have used so far only reflect *surface* (as opposed to *functional*) values, e.g., broken plurals are marked as singular. This might have a negative effect on learning generalizations over the complex agreement patterns in MSA (see Section 3), beyond memorization of word pairs seen together in training.

Predicting functional features To predict functional GENDER, functional NUMBER and RATIONALITY, we build a simple maximum likelihood estimate (MLE) model using these annotations in the corpus created by Alkuhlani and Habash (2011). We train using the same training data we use throughout this paper. For all three features, we select the most seen value in training associated with the triple *word*-CATIBEX-*lemma*; we back off to CATIBEX-*lemma* and then to *lemma*. For gender and number, we further back off to the surface values; for rationality, we back off to the most common value (*irrational*). On our predicted dev set, the overall accuracy baseline of predicting correct functional *gender-number-rationality* using surface features is

85.1% (for all POS tags). Our MLE model reduces the error by two thirds reaching an overall accuracy of 95.5%. The high accuracy may be a result of the low percentage of words in the dev set that do not appear in training (around 4.6%).

Digit tokens (e.g., "4") are also marked singular by default. They don't show surface agreement, even though the corresponding number-word token (*Arbς ħ* 'four.fem.sing') would. We further observe that MSA displays complex agreement patterns with numbers (Dada, 2007). Therefore, we alternatively experimented with binning the digit tokens' NUMBER value accordingly:

- the number 0 and numbers ending with 00
- the number 1 and numbers ending with 01
- the number 2 and numbers ending with 02
- the numbers 3-10 and those ending with 03-10
- the numbers, and numbers ending with, 11-99
- all other number tokens (e.g., 0.35 or 7/16)

and denoted these experiments with NUMDGTBIN. Almost 1.5% of the tokens are digit tokens in the training set, and 1.2% in the dev set.[10]

Results using these new features are shown in Table 6. The first part repeats the CORE12 baseline. The second part repeats previous experiments with surface morphological features. The third part uses the new functional morphological features instead. The performance using NUMBER and GENDER increases by 0.21% and 0.22%, respectively, as we replace surface features with functional features. (Recall that there is no functional PERSON.) We then see that the change in the representation of digits does not help; in the large space of experiments we have performed, we saw some improvement through the use of this alternative representation, but not in any of the feature combinations that performed best and that we report on in this paper. We then use just the RATIONALITY feature, which results in an increase over the baseline. The combination of all three functional features (NUMBER, GENDER, RATIONALITY) provides for a nice cumulative effect. Adding PERSON hardly improves further.

In the fourth part of the table, we include the other features which we found previously to be helpful, namely DET and LMM. Here, using DET2 instead of DET (see Section 5.4) gives us a slight improvement, providing our best result using the CORE12 POS tagset: 80.53%. This is a 1.85% improvement over using only the CORE12 POS tags (an 8.7% error reduction); of this improvement, 0.3% absolute (35% relative) is due to the use of functional features. We then use the best configuration, but without the RATIONALITY feature; we see that this feature on its own contributes 0.1% absolute, confirming its place in Arabic syntax. In gold experiments which we do not report here, the contribution was even higher (0.6-0.7%). The last row in the fourth part of Table 6 shows that using both surface and functional variants of NUMBER and GENDER does not help (hurts, in fact); the functional morphology features carry sufficient information for syntactic disambiguation.

The last part of the table revalidates the gains achieved of the best feature combination using the two other POS tagsets mentioned in Section 3: CATIBEX (the best performing tagset with predicted values), and BW (the best POS tagset with gold values in Marton et al. (2010), but results shown here are with predicted values). The CATIBEX result of 80.83% is our overall best result. The result using BW reconfirms that BW is not the best tagset to use for parsing Arabic with current prediction ability.

5.6 Validating results on unseen test set

Once experiments on the development set were done, we ran the best performing models on the previously unseen test set (Section 5.1). Table 7 shows that the same trends hold on this set as well.

Model	LAS	LAS_{diff}	UAS
CATIBEX	78.46	—	81.81
+DET2+LMM+PER+FN*NGR	**79.45**$^{++}$	**0.99**	**82.56**

Table 7: Results on unseen test set for models which performed best on dev set – predicted input.

6 Error Analysis

We analyze the attachment accuracy by attachment type. We show the accuracy for selected attachment types in Table 8. Using just CORE12, we see that some attachments (subject, modifications) are harder than others (objects, idafa). We see that by

[10] We didn't mark the number-words since in our training data there were less than 30 lemmas of less than 2000 such tokens, so presumably their agreement patterns can be more easily learned.

Features	SBJ	OBJ	MN	MP	IDF	Tot.
CORE12	67.9	90.4	72.0	70.3	94.5	78.7
CORE12 + LMM	68.8	90.4	72.6	70.9	94.6	79.0
CORE12 + DET2 +LMM+PNG	71.7	91.0	74.9	72.4	95.5	80.2
CORE12 + DET2 +LMM+PERS +FN*NGR	72.3	91.0	76.0	73.3	95.4	80.5

Table 8: Error analysis: Accuracy by attachment type (selected): subject, object, modification by a noun, modification (of a verb or a noun) by a preposition, idafa, and overall results (which match previously shown results)

adding LMM, all attachment types improve a little bit; this is as expected, since this feature provides a slight lexical abstraction. We then add features designed to improve idafa and those relations subject to agreement, subject and nominal modification (DET2, PERSON, NUMBER, GENDER). We see that as expected, subject, nominal modification (MN), and idafa reduce error by substantial margins (error reduction over CORE12+LMM greater than 8%, in the case of idafa the error reduction is 16.7%), while object and prepositional attachment (MP) improve to a lesser degree (error reduction of 6.2% or less). We assume that the relations not subject to agreement (object and prepositional attachment) improve because of the overall improvement in the parse due to the improvements in the other relations.

When we move to the functional features, we again see a reduction in the attachments which are subject to agreement, namely subject and nominal modification (error reductions over surface features of 2.1% and 4.4%, respectively). Idafa decreases slightly (since this relation is not affected by the functional features), while object stays the same. Surprisingly, prepositional attachment also improves, with an error reduction of 3.3%. Again, we can only explain this by proposing that the improvement in nominal modification attachment has the indirect effect of ruling out some bad prepositional attachments as well.

In summary, we see that not only do morphological features – and functional morphology features in particular – improve parsing, but they improve parsing in the way that we expect: those relations subject to agreement improve more than those that are not.

Last, we point out that MaltParser does not model generalized feature checking or matching directly, i.e., it has not learned that certain syntactic relations require identical (functional) morphological feature values. The gains in parsing quality reflect that the MaltParser SVM classifier has learned that the pairing of specific morphological feature values – e.g., *fem.sing.* for both the verb and its subject – is useful, with no generalization from each specific value to other values, or to general pair-wise value matching.

7 Conclusions and Future Work

We explored the contribution of different morphological (inflectional and lexical) features to dependency parsing of Arabic. We find that definiteness (DET), ϕ-features (PERSON, NUMBER, GENDER), and undiacritized lemma (LMM) are most helpful for Arabic dependency parsing on predicted input. We further find that functional morphology features and rationality improve over surface morphological features, as predicted by the complex agreement rules of Arabic. To our knowledge, this is the first result in Arabic NLP that uses functional morphology features, and that shows an improvement over surface features.

In future work, we intend to improve the prediction of functional morphological features in order to improve parsing accuracy. We also intend to investigate how these features can be integrated into other parsing frameworks; we expect them to help independently of the framework. We plan to make our parser available to other researchers. Please contact the authors if interested.

Acknowledgments

This work was supported by the DARPA GALE program, contract HR0011-08-C-0110. We thank Joakim Nivre for his useful remarks, Otakar Smrž for his help with Elixir-FM, Ryan Roth and Sarah Alkuhlani for their help with data, and three anonymous reviewers for useful comments. Part of the work was done while the first author was at Columbia University.

References

Sarah Alkuhlani and Nizar Habash. 2011. A corpus for modeling morpho-syntactic agreement in Arabic: gender, number and rationality. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, Portland, Oregon, USA.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of Computational Natural Language Learning (CoNLL)*, pages 149–164.

Timothy A. Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer Version 2.0. Linguistic Data Consortium, University of Pennsylvania, 2002. LDC Cat alog No.: LDC2004L02, ISBN 1-58563-324-0.

Michael Collins, Jan Hajic, Lance Ramshaw, and Christoph Tillmann. 1999. A statistical parser for czech. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, College Park, Maryland, USA, June.

Brooke Cowan and Michael Collins. 2005. Morphology and reranking for the statistical parsing of spanish. In *Proceedings of Human Language Technology (HLT) and the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 795–802.

Ali Dada. 2007. Implementation of Arabic numerals and their syntax in GF. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 9–16, Prague, Czech Republic.

Mona Diab. 2007. Towards an optimal pos tag set for modern standard arabic processing. In *Proceedings of Recent Advances in Natural Language Processing (RANLP)*, Borovets, Bulgaria.

Gülsen Eryigit, Joakim Nivre, and Kemal Oflazer. 2008. Dependency parsing of turkish. *Computational Linguistics*, 34(3):357–389.

Spence Green and Christopher D. Manning. 2010. Better Arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, pages 394–402, Beijing, China.

Nizar Habash and Owen Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 573–580, Ann Arbor, Michigan.

Nizar Habash and Ryan Roth. 2009. Catib: The columbia arabic treebank. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 221–224, Suntec, Singapore, August.

Nizar Habash, Reem Faraj, and Ryan Roth. 2009. Syntactic Annotation in the Columbia Arabic Treebank. In *Proceedings of MEDAR International Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

Nizar Habash. 2010. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Jan Hajič and Barbora Vidová-Hladká. 1998. Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structured Tagset. In *Proceedings of the International Conference on Computational Linguistics (COLING)- the Association for Computational Linguistics (ACL)*, pages 483–490.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool Publishers.

Seth Kulick, Ryan Gabbard, and Mitch Marcus. 2006. Parsing the Arabic Treebank: Analysis and improvements. In *Proceedings of the Treebanks and Linguistic Theories Conference*, pages 31–42, Prague, Czech Republic.

Mohamed Maamouri, Ann Bies, Timothy A. Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *Proceedings of the NEMLAR Conference on Arabic Language Resources and Tools*, pages 102–109, Cairo, Egypt.

Yuval Marton, Nizar Habash, and Owen Rambow. 2010. Improving Arabic dependency parsing with inflectional and lexical morphological features. In *Proceedings of Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL) at the 11th Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL) - Human Language Technology (HLT)*, Los Angeles, USA.

Jens Nilsson and Joakim Nivre. 2008. MaltEval: An evaluation and visualization tool for dependency parsing. In *Proceedings of the sixth International Conference on Language Resources and Evaluation (LREC)*, Marrakech, Morocco.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gulsen Eryigit, Sandra Kubler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Joakim Nivre, Igor M. Boguslavsky, and Leonid K. Iomdin. 2008. Parsing the SynTagRus Treebank of Russian. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 641–648.

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

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4).

Joakim Nivre. 2009. Parsing Indian languages with MaltParser. In *Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing*, pages 12–18.

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin. 2008. Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking. In *Proceedings of ACL-08: HLT, Short Papers*, pages 117–120, Columbus, Ohio, June. Association for Computational Linguistics.

Otakar Smrž. 2007. *Functional Arabic Morphology. Formal System and Implementation*. Ph.D. thesis, Charles University, Prague.

Reut Tsarfaty and Khalil Sima'an. 2007. Three-dimensional parametrization for parsing morphologically rich languages. In *Proceedings of the 10th International Conference on Parsing Technologies (IWPT)*, pages 156–167, Morristown, NJ, USA.

Partial Parsing from Bitext Projections

Prashanth Mannem and Aswarth Dara
Language Technologies Research Center
International Institute of Information Technology
Hyderabad, AP, India - 500032
{prashanth,abhilash.d}@research.iiit.ac.in

Abstract

Recent work has shown how a parallel corpus can be leveraged to build syntactic parser for a target language by projecting automatic source parse onto the target sentence using word alignments. The projected target dependency parses are not always fully connected to be useful for training traditional dependency parsers. In this paper, we present a greedy non-directional parsing algorithm which doesn't need a fully connected parse and can learn from partial parses by utilizing available structural and syntactic information in them. Our parser achieved statistically significant improvements over a baseline system that trains on only fully connected parses for Bulgarian, Spanish and Hindi. It also gave a significant improvement over previously reported results for Bulgarian and set a benchmark for Hindi.

1 Introduction

Parallel corpora have been used to transfer information from source to target languages for Part-Of-Speech (POS) tagging, word sense disambiguation (Yarowsky et al., 2001), syntactic parsing (Hwa et al., 2005; Ganchev et al., 2009; Jiang and Liu, 2010) and machine translation (Koehn, 2005; Tiedemann, 2002). Analysis on the source sentences was induced onto the target sentence via projections across word aligned parallel corpora.

Equipped with a source language parser and a word alignment tool, parallel data can be used to build an automatic treebank for a target language. The parse trees given by the parser on the source sentences in the parallel data are projected onto the target sentence using the word alignments from the alignment tool. Due to the usage of automatic source parses, automatic word alignments and differences in the annotation schemes of source and target languages, the projected parses are not always fully connected and can have edges missing (Hwa et al., 2005; Ganchev et al., 2009). Non-literal translations and divergences in the syntax of the two languages also lead to incomplete projected parse trees.

Figure 1 shows an English-Hindi parallel sentence with correct source parse, alignments and target dependency parse. For the same sentence, Figure 2 is a sample partial dependency parse projected using an automatic source parser on aligned text. This parse is not fully connected with the words *banaa*, *kottaige* and *dikhataa* left without any parents.

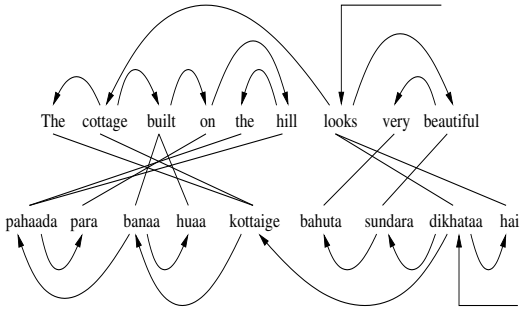


Figure 1: Word alignment with dependency parses for an English-Hindi parallel sentence

To train the traditional dependency parsers (Yamada and Matsumoto, 2003; Eisner, 1996; Nivre, 2003), the dependency parse has to satisfy four constraints: *connectedness*, *single-headedness*, *acyclicity* and *projectivity* (Kuhlmann and Nivre, 2006). Projectivity can be relaxed in some parsers (McDonald et al., 2005; Nivre, 2009). But these parsers can not directly be used to learn from partially connected parses (Hwa et al., 2005; Ganchev et al., 2009).

In the projected Hindi treebank (section 4) that was extracted from English-Hindi parallel text, only 5.9% of the sentences had full trees. In

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1597–1606,
Portland, Oregon, June 19-24, 2011.

Spanish and Bulgarian projected data extracted by Ganchev et al. (2009), the figures are 3.2% and 12.9% respectively. Learning from data with such high proportions of partially connected dependency parses requires special parsing algorithms which are not bound by *connectedness*. Its only during learning that the constraint doesn't satisfy. For a new sentence (i.e. during inference), the parser should output fully connected dependency tree.

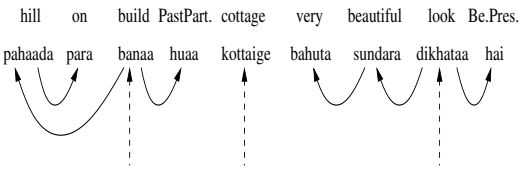


Figure 2: A sample dependency parse with partial parses

In this paper, we present a dependency parsing algorithm which can train on partial projected parses and can take rich syntactic information as features for learning. The parsing algorithm constructs the partial parses in a bottom-up manner by performing a greedy search over all possible relations and choosing the best one at each step without following either left-to-right or right-to-left traversal. The algorithm is inspired by earlier non-directional parsing works of Shen and Joshi (2008) and Goldberg and Elhadad (2010). We also propose an extended partial parsing algorithm that can learn from partial parses whose yields are partially contiguous.

Apart from bitext projections, this work can be extended to other cases where learning from partial structures is required. For example, while bootstrapping parsers high confidence parses are extracted and trained upon (Steedman et al., 2003; Reichart and Rappoport, 2007). In cases where these parses are few, learning from partial parses might be beneficial.

We train our parser on projected Hindi, Bulgarian and Spanish treebanks and show statistically significant improvements in accuracies between training on fully connected trees and learning from partial parses.

2 Related Work

Learning from partial parses has been dealt in different ways in the literature. Hwa et al. (2005) used post-projection completion/transformation rules to get full parse trees from the projections and train Collin's parser (Collins, 1999) on them. Ganchev et al. (2009) handle partial projected parses by avoiding committing to entire projected tree during training. The posterior regularization based framework constrains the projected syntactic relations to hold approximately and only in expectation. Jiang and Liu (2010) refer to alignment matrix and a dynamic programming search algorithm to obtain better projected dependency trees. They deal with partial projections by breaking down the projected parse into a set of edges and training on the set of projected relations rather than on trees.

While Hwa et al. (2005) requires full projected parses to train their parser, Ganchev et al. (2009) and Jiang and Liu (2010) can learn from partially projected trees. However, the discriminative training in (Ganchev et al., 2009) doesn't allow for richer syntactic context and it doesn't learn from all the relations in the partial dependency parse. By treating each relation in the projected dependency data independently as a classification instance for parsing, Jiang and Liu (2010) sacrifice the context of the relations such as global structural context, neighboring relations that are crucial for dependency analysis. Due to this, they report that the parser suffers from local optimization during training.

The parser proposed in this work (section 3) learns from partial trees by using the available structural information in it and also in neighboring partial parses. We evaluated our system (section 5) on Bulgarian and Spanish projected dependency data used in (Ganchev et al., 2009) for comparison. The same could not be carried out for Chinese (which was the language (Jiang and Liu, 2010) worked on) due to the unavailability of projected data used in their work. Comparison with the traditional dependency parsers (McDonald et al., 2005; Yamada and Matsumoto, 2003; Nivre, 2003; Goldberg and Elhadad, 2010) which train on complete dependency parsers is out of the scope of this work.

3 Partial Parsing

A standard dependency graph satisfies four graph constraints: *connectedness*, *single-headedness*, *acyclicity* and *projectivity* (Kuhlmann and Nivre, 2006). In our work, we assume the dependency graph for a sentence only satisfies the single-

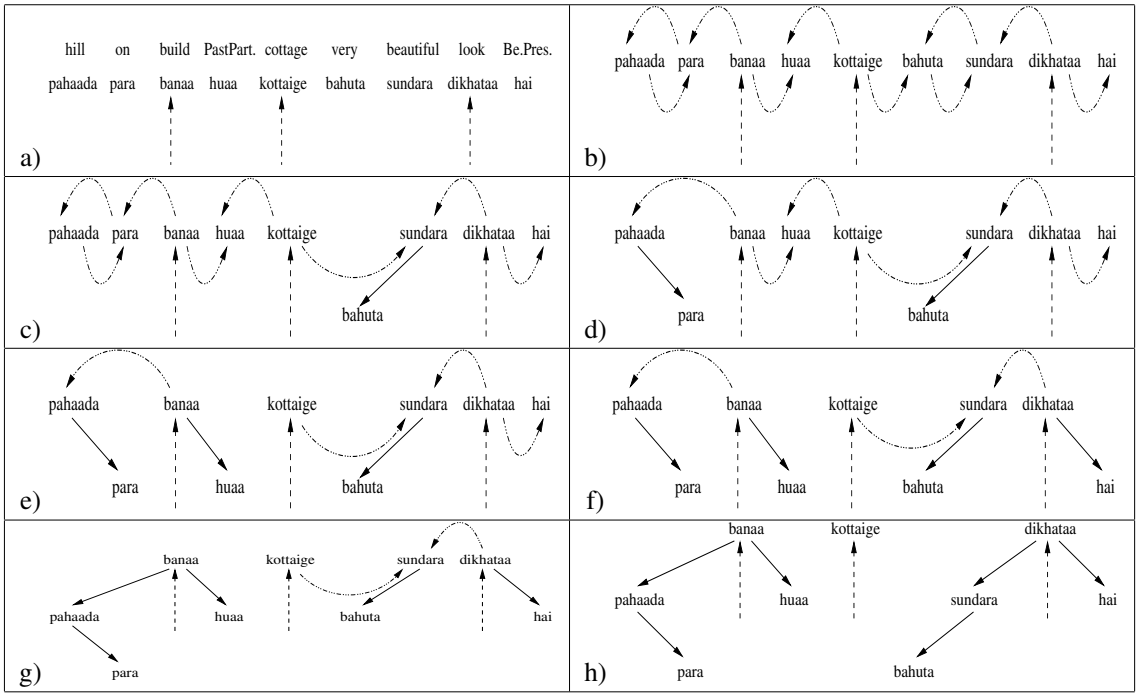


Figure 3: Steps taken by GNPPA. The dashed arcs indicate the unconnected words in *unConn*. The dotted arcs indicate the candidate arcs in *candidateArcs* and the solid arcs are the high scoring arcs that are stored in *builtPPs*

headedness, acyclicity and projectivity constraints while not necessarily being connected i.e. all the words need not have parents.

Given a sentence W=$w_0 \cdots w_n$ with a set of directed arcs A on the words in W, $w_i \rightarrow w_j$ denotes a dependency arc from w_i to w_j, (w_i,w_j) ϵ A. w_i is the parent in the arc and w_j is the child in the arc. $\stackrel{*}{\rightarrow}$ denotes the reflexive and transitive closure of the arc. $w_i \stackrel{*}{\rightarrow} w_j$ says that w_i *dominates* w_j, i.e. there is (possibly empty) path from w_i to w_j.

A node w_i is *unconnected* if it does not have an incoming arc. R is the set of all such unconnected nodes in the dependency graph. For the example in Figure 2, R={`banaa`, `kottaige`, `dikhataa`}. A *partial parse* rooted at node w_i denoted by $\rho(w_i)$ is the set of arcs that can be traversed from node w_i. The *yield* of a partial parse $\rho(w_i)$ is the set of nodes dominated by it. We use $\pi(w_i)$ to refer to the yield of $\rho(w_i)$ arranged in the linear order of their occurrence in the sentence. The *span* of the partial tree is the first and last words in its yield.

The dependency graph D can now be represented in terms of partial parses by $D = (W, R, \varrho(R))$ where W={$w_0 \cdots w_n$} is the sentence, R={$r_1 \cdots r_m$} is the set of unconnected nodes and $\varrho(R)$= {$\rho(r_1) \cdots \rho(r_m)$} is the set of partial parses rooted at these unconnected nodes. w_0 is a dummy word added at the beginning of W to behave as a root of a fully connected parse. A fully connected dependency graph would have only one element w_0 in R and the dependency graph rooted at w_0 as the only (fully connected) parse in $\varrho(R)$.

We assume the combined yield of $\varrho(R)$ spans the entire sentence and each of the partial parses in $\varrho(R)$ to be contiguous and non-overlapping with one another. A partial parse is *contiguous* if its yield is contiguous i.e. if a node $w_j \epsilon \pi(w_i)$, then all the words between w_i and w_j also belong to $\pi(w_i)$. A partial parse $\rho(w_i)$ is *non-overlapping* if the intersection of its yield $\pi(w_i)$ with yields of all other partial parses is empty.

3.1 Greedy Non-directional Partial Parsing Algorithm (GNPPA)

Given the sentence W and the set of unconnected nodes R, the parser follows a non-directional greedy approach to establish relations in a bottom up manner. The parser does a greedy search over all the possible relations and picks the one with

the highest score at each stage. This process is repeated until parents for all the nodes that do not belong to R are chosen.

Algorithm 1 lists the outline of the greedy non-directional partial parsing algorithm (GNPPA). *builtPPs* maintains a list of all the partial parses that have been built. It is initialized in line 1 by considering each word as a separate partial parse with just one node. *candidateArcs* stores all the arcs that are possible at each stage of the parsing process in a bottom up strategy. It is initialized in line 2 using the method *initCandidateArcs(*$w_0 \cdots w_n$*)*. *initCandidateArcs(*$w_0 \cdots w_n$*)* adds two candidate arcs for each pair of consecutive words with each other as parent (see Figure 3b). If an arc has one of the nodes in R as the child, it isn't included in *candidateArcs*.

Algorithm 1 Partial Parsing Algorithm

Input: sentence $w_0 \cdots w_n$ and set of partial tree roots *unConn*=$\{r_1 \cdots r_m\}$
Output: set of partial parses whose roots are in *unConn* (*builtPPs* = $\{\rho(r_1) \cdots \rho(r_m)\}$)

1: *builtPPs* = $\{\rho(r_1) \cdots \rho(r_n)\} \leftarrow \{w_0 \cdots w_n\}$
2: *candidateArcs* = initCandidateArcs($w_0 \cdots w_n$)
3: **while** *candidateArcs*.isNotEmpty() **do**
4: *bestArc* = $\underset{c_i \in candidateArcs}{\text{argmax}}\ score(c_i, \overrightarrow{w})$
5: *builtPPs*.remove(*bestArc*.child)
6: *builtPPs*.remove(*bestArc*.parent)
7: *builtPPs*.add(*bestArc*)
8: updateCandidateArcs(*bestArc*, *candidateArcs*, *builtPPs*, *unConn*)
9: **end while**
10: **return** builtPPs

Once initialized, the candidate arc with the highest score (line 4) is chosen and accepted into *builtPPs*. This involves replacing the best arc's child partial parse ρ(*arc*.child) and parent partial parse ρ(*arc*.parent) over which the arc has been formed with the arc ρ(*arc*.parent) $\rightarrow$ ρ(*arc*.child) itself in *builtPPs* (lines 5-7). In Figure 3f, to accept the best candidate arc ρ(banaa) $\rightarrow$ ρ(pahaada), the parser would remove the nodes ρ(banaa) and ρ(pahaada) in *builtPPs* and add ρ(banaa) $\rightarrow$ ρ(pahaada) to *builtPPs* (see Figure 3g).

After the best arc is accepted, the *candidateArcs* has to be updated (line 8) to remove the arcs that are no longer valid and add new arcs in the context of the updated *builtPPs*. Algorithm 2 shows the update procedure. First, all the arcs that end on the child are removed (lines 3-7) along with the arc from child to parent. Then, the immediately previous and next partial parses of the best arc in *builtPPs* are retrieved (lines 8-9) to add possible candidate arcs between them and the partial parse representing the best arc (lines 10-23). In the example, between Figures 3b and 3c, the arcs ρ(kottaige) $\rightarrow$ ρ(bahuta) and ρ(bahuta) $\rightarrow$ ρ(sundara) are first removed and the arc ρ(kottaige) $\rightarrow$ ρ(sundara) is added to *candidateArcs*. Care is taken to avoid adding arcs that end on unconnected nodes listed in R.

The entire GNPPA parsing process for the example sentence in Figure 2 is shown in Figure 3.

Algorithm 2 updateCandidateArcs(*bestArc*, *candidateArcs*, *builtPPs*, *unConn*)

1: *baChild* = bestArc.child
2: *baParent* = bestArc.parent
3: **for all** *arc* ϵ *candidateArcs* **do**
4: **if** *arc*.child = *baChild* or (*arc*.parent = *baChild* and *arc*.child = *baParent*) **then**
5: remove *arc*
6: **end if**
7: **end for**
8: *prevPP* = *builtPPs*.previousPP(*bestArc*)
9: *nextPP* = *builtPPs*.nextPP(*bestArc*)
10: **if** *bestArc*.direction == *LEFT* **then**
11: *newArc1* = new Arc(*prevPP*,*baParent*)
12: *newArc2* = new Arc(*baParent*,*prevPP*)
13: **end if**
14: **if** *bestArc*.direction == *RIGHT* **then**
15: *newArc1* = new Arc(*nextPP*,*baParent*)
16: *newArc2* = new Arc(*baParent*,*nextPP*)
17: **end if**
18: **if** *newArc1*.parent $\notin$ *unConn* **then**
19: *candidateArcs*.add(*newArc1*)
20: **end if**
21: **if** newArc2.parent $\notin$ unConn **then**
22: *candidateArcs*.add(*newArc2*)
23: **end if**
24: **return** *candidateArcs*

3.2 Learning

The algorithm described in the previous section uses a weight vector $\overrightarrow{w}$ to compute the best arc from the list of candidate arcs. This weight vector is learned using a simple Perceptron like algorithm similar to the one used in (Shen and Joshi, 2008). Algorithm 3 lists the learning framework for GNPPA.

For a training sample with sentence $w_0 \cdots w_n$, projected partial parses *projectedPPs*=$\{\rho(r_i) \cdots \rho(r_m)\}$, unconnected words *unConn* and weight vector $\overrightarrow{w}$, the *builtPPs* and *candidateArcs* are initiated as in algorithm 1. Then the arc with the highest score is selected. If this arc belongs to the parses in *projectedPPs*, *builtPPs* and *candidateArcs* are updated similar to the operations in

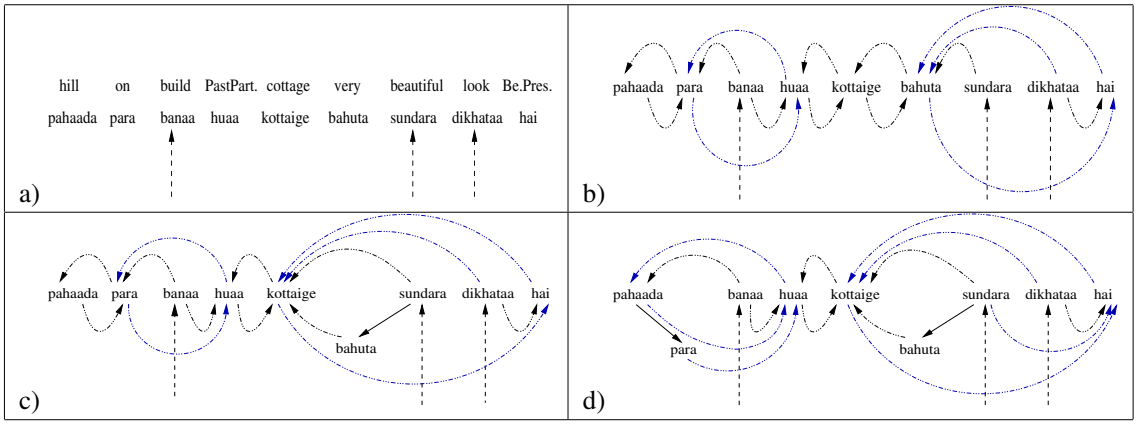


Figure 4: First four steps taken by E-GNPPA. The blue colored dotted arcs are the additional candidate arcs that are added to *candidateArcs*

algorithm 1. If it doesn't, it is treated as a negative sample and a corresponding positive candidate arc which is present both *projectedPPs* and *candidateArcs* is selected (lines 11-12).

The weights of the positive candidate arc are increased while that of the negative sample (best arc) are decreased. To reduce over fitting, we use averaged weights (Collins, 2002) in algorithm 1.

Algorithm 3 Learning for Non-directional Greedy Partial Parsing Algorithm

Input: sentence $w_0 \cdots w_n$, projected partial parses *projectedPPs*, unconnected words *unConn*, current $\overrightarrow{w}$
Output: updated $\overrightarrow{w}$
1: *builtPPs* = $\{\rho(r_1) \cdots \rho(r_n)\} \leftarrow \{w_0 \cdots w_n\}$
2: *candidateArcs* = initCandidateArcs($w_0 \cdots w_n$)
3: **while** *candidateArcs*.isNotEmpty() **do**
4: *bestArc* = $\underset{c_i \in candidateArcs}{\text{argmax}}\ score(c_i, \overrightarrow{w})$
5: **if** *bestArc* $\in$ *projectedPPs* **then**
6: *builtPPs*.remove(*bestArc*.child)
7: *builtPPs*.remove(*bestArc*.parent)
8: *builtPPs*.add(*bestArc*)
9: updateCandidateArcs(*bestArc*, *candidateArcs*, *builtPPs*, *unConn*)
10: **else**
11: allowedArcs = $\{c_i \mid c_i \in$ *candidateArcs* && $c_i \in$ *projectedArcs*$\}$
12: *compatArc* = $\underset{c_i \in allowedArcs}{\text{argmax}}\ score(c_i, \overrightarrow{w})$
13: promote(*compatArc*, $\overrightarrow{w}$)
14: demote(*bestArc*, $\overrightarrow{w}$)
15: **end if**
16: **end while**
17: **return** builtPPs

3.3 Extended GNPPA (E-GNPPA)

The GNPPA described in section 3.1 assumes that the partial parses are contiguous. The example in Figure 5 has a partial tree ρ(dikhataa) which isn't contiguous. Its yield doesn't contain bahuta and sundara. We call such *non-contiguous* partial parses whose yields encompass the yield of an other partial parse as *partially contiguous*. Partially contiguous parses are common in the projected data and would not be parsable by the algorithm 1 (ρ(dikhataa) $\rightarrow$ ρ(kottaige) would not be identified).

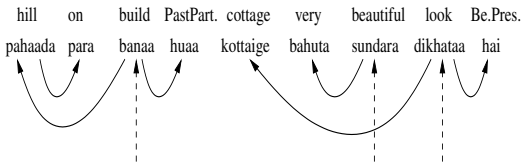


Figure 5: Dependency parse with a partially contiguous partial parse

In order to identify and learn from relations which are part of partially contiguous partial parses, we propose an extension to GNPPA. The extended GNPAA (E-GNPPA) broadens its scope while searching for possible candidate arcs given R and *builtPPs*. If the immediate previous or the next partial parses over which arcs are to be formed are designated unconnected nodes, the parser looks further for a partial parse over which it can form arcs. For example, in Figure 4b, the arc ρ(para) $\rightarrow$ ρ(banaa) can not be added to the *candidateArcs* since banaa is a designated unconnected node in $unConn$. The E-GNPPA looks over the unconnected node and adds the arc ρ(para) $\rightarrow$ ρ(huaa) to the candidate arcs list *candidateArcs*.

E-GNPPA differs from algorithm 1 in lines 2 and 8. The E-GNPPA uses an extended initialization method *initCandidateArcsExtended(w_0)* for

Parent and Child	par.pos, chd.pos, par.lex, chd.lex
Sentence Context	*par-1*.pos, *par-2*.pos, *par+1*.pos, *par+2*.pos, *par-1*.lex, *par+1*.lex *chd-1*.pos, *chd-2*.pos, *chd+1*.pos, *chd+2*.pos, *chd-1*.lex, *chd+1*.lex
Structural Info	leftMostChild(par).pos, rightMostChild(par).pos, leftSibling(chd).pos, rightSibling(chd).pos
Partial Parse Context	previousPP().pos, previousPP().lex, nextPP().pos, nextPP().lex

Table 1: Information on which features are defined. par denotes the parent in the relation and chd the child. .pos and .lex is the POS and word-form of the corresponding node. *+/-i* is the previous/next i^{th} word in the sentence. leftMostChild() and rightMostChild() denote the left most and right most children of a node. leftSibling() and rightSibling() get the immediate left and right siblings of a node. previousPP() and nextPP() return the immediate previous and next partial parses of the arc in *builtPPs* at the state.

candidateArcs in line 2 and an extended procedure *updateCandidateArcsExtended* to update the *candidateArcs* after each step in line 8. Algorithm 4 shows the changes w.r.t algorithm 2. Figure 4 presents the steps taken by the E-GNPPA parser for the example parse in Figure 5.

Algorithm 4 updateCandidateArcsExtended (*bestArc*, *candidateArcs*, *builtPPs*,*unConn*)

```
··· lines 1 to 7 of Algorithm 2 ···
prevPP = builtPPs.previousPP(bestArc)
while prevPP ∈ unConn do
    prevPP = builtPPs.previousPP(prevPP)
end while
nextPP = builtPPs.nextPP(bestArc)
while nextPP ∈ unConn do
    nextPP = builtPPs.nextPP(nextPP)
end while
··· lines 10 to 24 of Algorithm 2 ···
```

3.4 Features

Features for a relation (candidate arc) are defined on the POS tags and lexical items of the nodes in the relation and those in its context. Two kinds of context are used a) context from the input sentence (*sentence context*) b) context in *builtPPs* i.e. nearby partial parses (*partial parse context*). Information from the partial parses (*structural info*) such as left and right most children of the parent node in the relation, left and right siblings of the child node in the relation are also used. Table 1 lists the information on which features are defined in the various configurations of the three language parsers. The actual features are combinations of the information present in the table. The set varies depending on the language and whether its GNPPA or E-GNPPA approach.

While training, no features are defined on whether a node is unconnected (present in *unConn*) or not as this information isn't available during testing.

4 Hindi Projected Dependency Treebank

We conducted experiments on English-Hindi parallel data by transferring syntactic information from English to Hindi to build a projected dependency treebank for Hindi.

The TIDES English-Hindi parallel data containing 45,000 sentences was used for this purpose [1] (Venkatapathy, 2008). Word alignments for these sentences were obtained using the widely used GIZA++ toolkit in grow-diag-final-and mode (Och and Ney, 2003). Since Hindi is a morphologically rich language, root words were used instead of the word forms. A bidirectional English POS tagger (Shen et al., 2007) was used to POS tag the source sentences and the parses were obtained using the first order MST parser (McDonald et al., 2005) trained on dependencies extracted from Penn treebank using the head rules of Yamada and Matsumoto (2003). A CRF based Hindi POS tagger (PVS. and Gali, 2007) was used to POS tag the target sentences.

English and Hindi being morphologically and syntactically divergent makes the word alignment and dependency projection a challenging task. The source dependencies are projected using an approach similar to (Hwa et al., 2005). While they use post-projection transformations on the projected parse to account for annotation differences, we use pre-projection transformations on the source parse. The projection algorithm pro-

[1] The original data had 50,000 parallel sentences. It was later refined by IIIT-Hyderabad to remove repetitions and other trivial errors. The corpus is still noisy with typographical errors, mismatched sentences and unfaithful translations.

duces acyclic parses which could be unconnected and non-projective.

4.1 Annotation Differences in Hindi and English

Before projecting the source parses onto the target sentence, the parses are transformed to reflect the annotation scheme differences in English and Hindi. While English dependency parses reflect the PTB annotation style (Marcus et al., 1994), we project them to Hindi to reflect the annotation scheme described in (Begum et al., 2008). The differences in the annotation schemes are with respect to three phenomena: a) head of a verb group containing auxiliary and main verbs, b) prepositions in a prepositional phrase (PP) and c) coordination structures.

In the English parses, the auxiliary verb is the head of the main verb while in Hindi, the main verb is the head of the auxiliary in the verb group. For example, in the Hindi parse in Figure 1, `dikhataa` is the head of the auxiliary verb `hai`. The prepositions in English are realized as post-positions in Hindi. While prepositions are the heads in a preposition phrase, post-positions are the modifiers of the preceding nouns in Hindi. In `pahaada para` (*on the hill*), `hill` is the head of `para`. In coordination structures, while English differentiates between how NP coordination and VP coordination structures behave, Hindi annotation scheme is consistent in its handling. Left-most verb is the head of a VP coordination structure in English whereas the rightmost noun is the head in case of NP coordination. In Hindi, the conjunct is the head of the two verbs/nouns in the coordination structure.

These three cases are identified in the source tree and appropriate transformations are made to the source parse itself before projecting the relations using word alignments.

5 Experiments

We carried out all our experiments on parallel corpora belonging to English-Hindi, English-Bulgarian and English-Spanish language pairs. While the Hindi projected treebank was obtained using the method described in section 4, Bulgarian and Spanish projected datasets were obtained using the approach in (Ganchev et al., 2009). The datasets of Bulgarian and Spanish that contributed to the best accuracies for Ganchev et al. (2009) were used in our work (7 rules dataset for Bulgarian and 3 rules dataset for Spanish). The Hindi, Bulgarian and Spanish projected dependency treebanks have 44760, 39516 and 76958 sentences respectively. Since we don't have confidence scores for the projections on the sentences, we picked 10,000 sentences randomly in each of the three datasets for training the parsers[2]. Other methods of choosing the 10K sentences such as those with the max. no. of relations, those with least no. of unconnected words, those with max. no. of contiguous partial trees that can be learned by GNPPA parser etc. were tried out. Among all these, random selection was consistent and yielded the best results. The errors introduced in the projected parses by errors in word alignment, source parser and projection are not consistent enough to be exploited to select the better parses from the entire projected data.

Statistic	Hindi	Bulgarian	Spanish
N(Words)	226852	71986	133124
N(Parent==-1)	44607	30268	54815
P(Parent==-1)	19.7	42.0	41.1
N(Full trees)	593	1299	327
N(GNPPA)	30063	10850	19622
P(GNPPA)	16.4	26.0	25.0
N(E-GNPPA)	35389	12281	24577
P(E-GNPPA)	19.3	29.4	30.0

Table 2: Statistics of the Hindi, Bulgarian and Spanish projected treebanks used for experiments. Each of them has 10,000 randomly picked parses. N(X) denotes *number of X* and P(X) denotes *percentage of X*. N(Words) is the number of words. N(Parents==-1) is the number of words without a parent. N(Full trees) is the number of parses which are fully connected. N(GNPPA) is the number of relations learnt by GNPPA parser and N(E-GNPPA) is the number of relations learnt by E-GNPPA parser. Note that P(GNPPA) is calculated as N(GNPPA)/(N(Words) - N(Parents==-1)).

Table 2 gives an account of the randomly chosen 10k sentences in terms of the number of words, words without parents etc. Around 40% of the words spread over 88% of sentences in Bulgarian and 97% of sentences in Spanish have no parents. Traditional dependency parsers which only train from fully connected trees would not be able to learn from these sentences. P(GNPPA) is the percentage of relations in the data that are learned by the GNPPA parser satisfying the contiguous partial tree constraint and P(E-GNPPA) is the per-

[2]Exactly 10K sentences were selected in order to compare our results with those of (Ganchev et al., 2009).

Parser	Hindi		Bulgarian		Spanish	
	Punct	*NoPunct*	*Punct*	*NoPunct*	*Punct*	*NoPunct*
Baseline	78.70	77.39	51.85	55.15	41.60	45.61
GNPPA	80.03*	78.81*	77.03*	79.06*	65.49*	68.70*
E-GNPPA	**81.10***†	**79.94***†	**78.93***†	**80.11***†	**67.69***†	**70.90***†

Table 3: UAS for Hindi, Bulgarian and Spanish with the baseline, GNPPA and E-GNPPA parsers trained on 10k parses selected randomly. *Punct* indicates evaluation with punctuation whereas *NoPunct* indicates without punctuation. * next to an accuracy denotes statistically significant (McNemar's and $p < 0.05$) improvement over the baseline. † denotes significance over GNPPA

centage that satisfies the partially contiguous constraint. E-GNPPA parser learns around 2-5% more no. of relations than GNPPA due to the relaxation in the constraints.

The Hindi test data that was released as part of the ICON-2010 Shared Task (Husain et al., 2010) was used for evaluation. For Bulgarian and Spanish, we used the same test data that was used in the work of Ganchev et al. (2009). These test datasets had sentences from the training section of the CoNLL Shared Task (Nivre et al., 2007) that had lengths less than or equal to 10. All the test datasets have gold POS tags.

A baseline parser was built to compare learning from partial parses with learning from fully connected parses. Full parses are constructed from partial parses in the projected data by randomly assigning parents to unconnected parents, similar to the work in (Hwa et al., 2005). The unconnected words in the parse are selected randomly one by one and are assigned parents randomly to complete the parse. This process is repeated for all the sentences in the three language datasets. The parser is then trained with the GNPPA algorithm on these fully connected parses to be used as the baseline.

Table 3 lists the accuracies of the baseline, GNPPA and E-GNPPA parsers. The accuracies are unlabeled attachment scores (UAS): the percentage of words with the correct head. Table 4 compares our accuracies with those reported in (Ganchev et al., 2009) for Bulgarian and Spanish.

5.1 Discussion

The baseline reported in (Ganchev et al., 2009) significantly outperforms our baseline (see Table 4) due to the different baselines used in both the works. In our work, while creating the data for the baseline by assigning random parents to unconnected words, acyclicity and projectivity constraints are not enforced. Ganchev et al. (2009)'s baseline is similar to the first iteration of their discriminative model and hence performs better than ours. Our Bulgarian E-GNPPA parser achieved a 1.8% gain over theirs while the Spanish results are lower. Though their training data size is also 10K, the training data is different in both our works due to the difference in the method of choosing 10K sentences from the large projected treebanks.

Parser	Bulgarian	Spanish
Ganchev-Baseline	72.6	69.0
Baseline	55.15	45.61
Ganchev-Discriminative	78.3	**72.3**
GNPPA	79.06	68.70
E-GNPPA	**80.11**	70.90

Table 4: Comparison of baseline, GNPPA and E-GNPPA with baseline and discriminative model from (Ganchev et al., 2009) for Bulgarian and Spanish. Evaluation didn't include punctuation.

The GNPPA accuracies (see table 3) for all the three languages are significant improvements over the baseline accuracies. This shows that learning from partial parses is effective when compared to imposing the connected constraint on the partially projected dependency parse. Even while projecting source dependencies during data creation, it is better to project high confidence relations than look to project more relations and thereby introduce noise.

The E-GNPPA which also learns from partially contiguous partial parses achieved statistically significant gains for all the three languages. The gains across languages is due to the fact that in the 10K data that was used for training, E-GNPPA parser could learn $2 - 5\%$ more relations over GNPPA (see Table 2).

Figure 6 shows the accuracies of baseline and E-

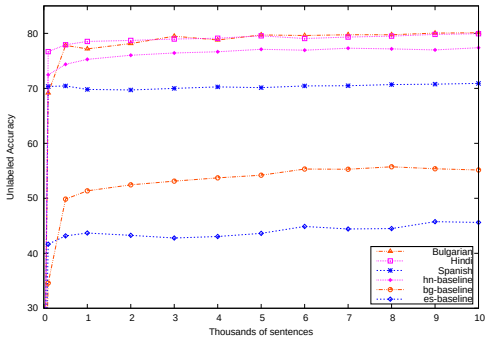


Figure 6: Accuracies (without punctuation) w.r.t varying training data sizes for baseline and E-GNPPA parsers.

GNPPA parser for the three languages when training data size is varied. The parsers peak early with less than 1000 sentences and make small gains with the addition of more data.

6 Conclusion

We presented a non-directional parsing algorithm that can learn from partial parses using syntactic and contextual information as features. A Hindi projected dependency treebank was developed from English-Hindi bilingual data and experiments were conducted for three languages Hindi, Bulgarian and Spanish. Statistically significant improvements were achieved by our partial parsers over the baseline system. The partial parsing algorithms presented in this paper are not specific to bitext projections and can be used for learning from partial parses in any setting.

References

R. Begum, S. Husain, A. Dhwaj, D. Sharma, L. Bai, and R. Sangal. 2008. Dependency annotation scheme for indian languages. In *In Proceedings of The Third International Joint Conference on Natural Language Processing (IJCNLP)*, Hyderabad, India.

Michael John Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA. AAI9926110.

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of the 16th conference on Computational linguistics - Volume 1*, pages 340–345, Morristown, NJ, USA. Association for Computational Linguistics.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL-IJCNLP '09, pages 369–377, Morristown, NJ, USA. Association for Computational Linguistics.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 742–750, Morristown, NJ, USA. Association for Computational Linguistics.

Samar Husain, Prashanth Mannem, Bharath Ambati, and Phani Gadde. 2010. Icon 2010 tools contest on indian language dependency parsing. In *Proceedings of ICON 2010 NLP Tools Contest*.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Nat. Lang. Eng.*, 11:311–325, September.

Wenbin Jiang and Qun Liu. 2010. Dependency parsing and projection based on word-pair classification. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 12–20, Morristown, NJ, USA. Association for Computational Linguistics.

P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5. Citeseer.

Marco Kuhlmann and Joakim Nivre. 2006. Mildly non-projective dependency structures. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 507–514, Morristown, NJ, USA. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewicz. 1994. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jens Nilsson and Joakim Nivre. 2008. Malteval: an evaluation and visualization tool for dependency parsing. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan Mcdonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic. Association for Computational Linguistics.

Joakim Nivre. 2003. An Efficient Algorithm for Projective Dependency Parsing. In *Eighth International Workshop on Parsing Technologies*, Nancy, France.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359, Suntec, Singapore, August. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Avinesh PVS. and Karthik Gali. 2007. Part-Of-Speech Tagging and Chunking using Conditional Random Fields and Transformation-Based Learning. In *Proceedings of the IJCAI and the Workshop On Shallow Parsing for South Asian Languages (SPSAL)*, pages 21–24.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 616–623, Prague, Czech Republic, June. Association for Computational Linguistics.

Libin Shen and Aravind Joshi. 2008. LTAG dependency parsing with bidirectional incremental construction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 495–504, Honolulu, Hawaii, October. Association for Computational Linguistics.

L. Shen, G. Satta, and A. Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, pages 331–338, Morristown, NJ, USA. Association for Computational Linguistics.

Jrg Tiedemann. 2002. MatsLex - a multilingual lexical database for machine translation. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC'2002)*, volume VI, pages 1909–1912, Las Palmas de Gran Canaria, Spain, 29-31 May.

Sriram Venkatapathy. 2008. Nlp tools contest - 2008: Summary. In *Proceedings of ICON 2008 NLP Tools Contest*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *In Proceedings of IWPT*, pages 195–206.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*, HLT '01, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

Ranking Class Labels Using Query Sessions

Marius Paşca
Google Inc.
1600 Amphitheatre Parkway
Mountain View, California 94043
mars@google.com

Abstract

The role of search queries, as available within query sessions or in isolation from one another, in examined in the context of ranking the class labels (e.g., *brazilian cities*, *business centers*, *hilly sites*) extracted from Web documents for various instances (e.g., *rio de janeiro*). The co-occurrence of a class label and an instance, in the same query or within the same query session, is used to reinforce the estimated relevance of the class label for the instance. Experiments over evaluation sets of instances associated with Web search queries illustrate the higher quality of the query-based, re-ranked class labels, relative to ranking baselines using document-based counts.

1 Introduction

Motivation: The offline acquisition of instances (*rio de janeiro*, *porsche cayman*) and their corresponding class labels (*brazilian cities*, *locations*, *vehicles*, *sports cars*) from text has been an active area of research. In order to extract fine-grained classes of instances, existing methods often apply manually-created (Banko et al., 2007; Talukdar et al., 2008) or automatically-learned (Snow et al., 2006) extraction patterns to text within large document collections.

In Web search, the relative ranking of documents returned in response to a query directly affects the outcome of the search. Similarly, the quality of the relative ranking among class labels extracted for a given instance influences any applications (e.g., query refinements or structured extraction) using the extracted data. But due to noise in Web data and limitations of extraction techniques, class labels acquired for a given instance (e.g., *oil shale*) may fail to properly capture the semantic classes to which the instance may belong (Kozareva et al., 2008). Inevitably, some of the extracted class labels will be less useful (e.g., *sources*, *mutual concerns*) or incorrect (e.g., *plants* for the instance *oil shale*). In previous work, the relative ranking of class labels for an instance is determined mostly based on features derived from the source Web documents from which the data has been extracted, such as variations of the frequency of co-occurrence or diversity of extraction patterns producing a given pair (Etzioni et al., 2005).

Contributions: This paper explores the role of Web search queries, rather than Web documents, in inducing superior ranking among class labels extracted automatically from documents for various instances. It compares two sources of indirect ranking evidence available within anonymized query logs: a) co-occurrence of an instance and its class label in the same query; and b) co-occurrence of an instance and its class label, as separate queries within the same query session. The former source is a noisy attempt to capture queries that narrow the search results to a particular class of the instance (e.g., *jaguar car maker*). In comparison, the latter source noisily identifies searches that specialize from a class (e.g., *car maker*) to an instance (e.g., *jaguar*) or, conversely, generalize from an instance to a class. To our knowledge, this is the first study comparing inherently-noisy queries and query sessions for the purpose of ranking of open-domain, labeled class instances.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1607–1615,
Portland, Oregon, June 19-24, 2011.

The remainder of the paper is organized as follows. Section 2 introduces intuitions behind an approach using queries for ranking class labels of various instances, and describes associated ranking functions. Sections 3 and 4 describe the experimental setting and evaluation results over evaluation sets of instances associated with Web search queries. The results illustrate the higher quality of the query-based, re-ranked lists of class labels, relative to alternative ranking methods using only document-based counts.

2 Instance Class Ranking via Query Logs

Ranking Hypotheses: We take advantage of anonymized query logs, to induce superior ranking among the class labels associated with various class instances within an IsA repository acquired from Web documents. Given a class instance $\mathcal{I}$, the functions used for the ranking of its class labels are chosen following several observations.

• *Hypothesis H_1*: If $\mathcal{C}$ is a prominent class of an instance $\mathcal{I}$, then $\mathcal{C}$ and $\mathcal{I}$ are likely to occur in text in contexts that are indicative of an IsA relation.

• *Hypothesis H_2*: If $\mathcal{C}$ is a prominent class of an instance $\mathcal{I}$, and $\mathcal{I}$ is ambiguous, then a fraction of the queries about $\mathcal{I}$ may also refer to and contain $\mathcal{C}$.

• *Hypothesis H_3*: If $\mathcal{C}$ is a prominent class of an instance $\mathcal{I}$, then a fraction of the queries about $\mathcal{I}$ may be followed by queries about $\mathcal{C}$, and vice-versa.

Ranking Functions: The ranking functions follow directly from the above hypotheses.

• *Ranking based on H_1* (using documents): The first hypothesis H_1 is a reformulation of findings from previous work (Etzioni et al., 2005). In practice, a class label is deemed more relevant for an instance if the pair is extracted more frequently and by multiple patterns, with the scoring formula:

$$Score_{H1}(\mathcal{C},\mathcal{I}) = Freq(\mathcal{C},\mathcal{I}) \times Size(\{Pattern(\mathcal{C})\})^2 \quad (1)$$

where $Freq(\mathcal{C},\mathcal{I})$ is the frequency of extraction of $\mathcal{C}$ for the instance $\mathcal{I}$, and $Size(\{Pattern(\mathcal{C})\})$ is the number of unique patterns extracting the class label $\mathcal{C}$ for the instance $\mathcal{I}$. The patterns are hand-written, following (Hearst, 1992):

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

where $\mathcal{I}$ is a potential instance (e.g., *diderot*) and $\mathcal{C}$ is a potential class label (e.g., *writers*). The boundaries are approximated from the part-of-speech tags of the sentence words, for potential class labels $\mathcal{C}$; and identified by checking that $\mathcal{I}$ occurs as an entire query in query logs, for instances $\mathcal{I}$ (Van Durme and Paşca, 2008).

The application of the scoring formula (1) to candidates extracted from the Web produces a ranked list of class labels $L_{H1}(\mathcal{I})$.

• *Ranking based on H_2* (using queries): Intuitively, Web users searching for information about $\mathcal{I}$ sometimes add some or all terms of $\mathcal{C}$ to a search query already containing $\mathcal{I}$, either to further specify their query, or in response to being presented with sets of search results spanning several meanings of an ambiguous instance. Examples of such queries are *happiness emotion* and *diderot philosopher*. Moreover, queries like *happiness positive psychology* and *diderot enlightenment* may be considered to weakly and partially reinforce the relevance of the class labels *positive emotions* and *enlightenment writers* of the instances *happiness* and *diderot* respectively. In practice, a class label is deemed more relevant if its individual terms occur in popular queries containing the instance. More precisely, for each term within any class label from $L_{H1}(\mathcal{I})$, we compute a score $TermQueryScore$. The score is the frequency sum of the term within anonymized queries containing the instance $\mathcal{I}$ as a prefix, and the term anywhere else in the queries. Terms are stemmed before the computation.

Each class label $\mathcal{C}$ is assigned the geometric mean of the scores of its N terms $\mathcal{T}_i$, after ignoring stop words:

$$Score_{H2}(\mathcal{C},\mathcal{I}) = (\prod_{i=1}^{N} TermQueryScore(\mathcal{T}_i))^{1/N} \quad (2)$$

The geometric mean is preferred to the arithmetic mean, because the latter is more strongly affected by outlier values. The class labels are ranked according to the means, resulting in a ranked list $L_{H2}(\mathcal{I})$. In case of ties, $L_{H2}(\mathcal{I})$ keeps the relative ranking from $L_{H1}(\mathcal{I})$.

• *Ranking based on H_3* (using query sessions): Given the third hypothesis H_3, Web users searching for information about $\mathcal{I}$ may subsequently search for more general information about one of its classes $\mathcal{C}$. Conversely, users may specialize their search from a class $\mathcal{C}$ to one of its instances $\mathcal{I}$. Examples of such queries are *happiness* followed later by *emotions*, or *diderot* followed by *philosophers*; or *emo-*

tions followed later by *happiness*, or *philosophers* followed by *diderot*. In practice, a class label is deemed more relevant if its individual terms occur as part of queries that are in the same query session as a query containing only the instance. More precisely, for each term within any class label from $L_{H1}(\mathcal{I})$, we compute a score $TermSessionScore$, equal to the frequency sum of the anonymized queries from the query sessions that contain the term and are: a) either the initial query of the session, with the instance $\mathcal{I}$ being one of the subsequent queries from the same session; or b) one of the subsequent queries of the session, with the instance $\mathcal{I}$ being the initial query of the same session. Before computing the frequencies, the class label terms are stemmed.

Each class label $\mathcal{C}$ is assigned the geometric mean of the scores of its terms, after ignoring stop words:

$$Score_{H3}(\mathcal{C},\mathcal{I}) = (\prod_{i=1}^{N} TermSessionScore(\mathcal{T}_i))^{1/N} \quad (3)$$

The class labels are ranked according to the geometric means, resulting in a ranked list $L_{H3}(\mathcal{I})$. In case of ties, $L_{H3}(\mathcal{I})$ preserves the relative ranking from $L_{H1}(\mathcal{I})$.

Unsupervised Ranking: Given an instance $\mathcal{I}$, the ranking hypotheses and corresponding functions $L_{H1}(\mathcal{I})$, $L_{H2}(\mathcal{I})$ and $L_{H3}(\mathcal{I})$ (or any combination of them) can be used together to generate a merged, ranked list of class labels per instance $\mathcal{I}$. The score of a class label in the merged list is determined by the inverse of the average rank in the lists $L_{H1}(\mathcal{I})$ and $L_{H2}(\mathcal{I})$ and $L_{H3}(\mathcal{I})$, computed with the following formula:

$$Score_{H1+H2+H3}(\mathcal{C},\mathcal{I}) = \frac{N}{\sum_i^N Rank(\mathcal{C}, L_{Hi})} \quad (4)$$

where N is the number of input lists of class labels (in this case, 3), and Rank($\mathcal{C}$, L_{Hi}) is the rank of $\mathcal{C}$ in the input list of class labels L_{Hi} (L_{H1}, L_{H2} or L_{H3}). The rank is set to 1000, if $\mathcal{C}$ is not present in the list L_{Hi}. By using only the relative ranks and not the absolute scores of the class labels within the input lists, the outcome of the merging is less sensitive to how class labels of a given instance are numerically scored within the input lists. In case of ties, the scores of the class labels from $L_{H1}(\mathcal{I})$ serve as a secondary ranking criterion. Thus, every instance $\mathcal{I}$ from the IsA repository is associated with a ranked list of class labels computed according to this ranking formula. Conversely, each class label $\mathcal{C}$ from the IsA repository is associated with a ranked list of class instances computed with the earlier scoring formula (1) used to generate lists $L_{H1}(\mathcal{I})$.

Note that the ranking formula can also consider only a subset of the available input lists. For instance, $Score_{H1+H2}$ would use only $L_{H1}(\mathcal{I})$ and $L_{H2}(\mathcal{I})$ as input lists; $Score_{H1+H3}$ would use only $L_{H1}(\mathcal{I})$ and $L_{H3}(\mathcal{I})$ as input lists; etc.

3 Experimental Setting

Textual Data Sources: The acquisition of the IsA repository relies on unstructured text available within Web documents and search queries. The queries are fully-anonymized queries in English submitted to Google by Web users in 2009, and are available in two collections. The first collection is a random sample of 50 million unique queries that are independent from one another. The second collection is a random sample of 5 million query sessions. Each session has an initial query and a series of subsequent queries. A subsequent query is a query that has been submitted by the same Web user within no longer than a few minutes after the initial query. Each subsequent query is accompanied by its frequency of occurrence in the session, with the corresponding initial query. The document collection consists of a sample of 100 million documents in English.

Experimental Runs: The experimental runs correspond to different methods for extracting and ranking pairs of an instance and a class:

- from the repository extracted here, with class labels of an instance ranked based on the frequency and the number of extraction patterns ($Score_{H1}$ from Equation (1) in Section 2), in run $\underline{\mathbf{R}_d}$;
- from the repository extracted here, with class labels of an instance ranked via the rank-based merging of: $Score_{H1+H2}$ from Section 2, in run $\underline{\mathbf{R}_p}$, which corresponds to re-ranking using co-occurrence of an instance and its class label in the same query; $Score_{H1+H3}$ from Section 2, in run $\underline{\mathbf{R}_s}$, which corresponds to re-ranking using co-occurrence of an instance and its class label, as separate queries within the same query session; and $Score_{H1+H2+H3}$ from Section 2, in run $\underline{\mathbf{R}_u}$, which corresponds to re-ranking using both types of co-occurrences in queries.

Evaluation Procedure: The manual evaluation of open-domain information extraction output is time consuming (Banko et al., 2007). A more practical alternative is an automatic evaluation procedure for ranked lists of class labels, based on existing resources and systems.

Assume that there is a gold standard, containing gold class labels that are each associated with a gold set of their instances. The creation of such gold standards is discussed later. Based on the gold standard, the ranked lists of class labels available within an IsA repository can be automatically evaluated as follows. First, for each gold label, the ranked lists of class labels of individual gold instances are retrieved from the IsA repository. Second, the individual retrieved lists are merged into a ranked list of class labels, associated with the gold label. The merged list can be computed, e.g., using an extension of the $Score_{H1+H2+H3}$ formula (Equation (4)) described earlier in Section 2. Third, the merged list is compared against the gold label, to estimate the accuracy of the merged list. Intuitively, a ranked list of class labels is a better approximation of a gold label, if class labels situated at better ranks in the list are closer in meaning to the gold label.

Evaluation Metric: Given a gold label and a list of class labels, if any, derived from the IsA repository, the rank of the highest class label that matches the gold label determines the score assigned to the gold label, in the form of the reciprocal rank of the match. Thus, if the gold label matches a class label at rank 1, 2 or 3 in the computed list, the gold label receives a score of 1, 0.5 or 0.33 respectively. The score is 0 if the gold label does not match any of the top 20 class labels. The overall score over the entire set of gold labels is the mean reciprocal rank (MRR) score over all gold labels from the set. Two types of MRR scores are automatically computed:

• **MRR**$_f$ considers a gold label and a class label to match, if they are identical;

• **MRR**$_p$ considers a gold label and a class label to match, if one or more of their tokens that are not stop words are identical.

During matching, all string comparisons are case-insensitive, and all tokens are first converted to their singular form (e.g., *european countries* to *european country*) using WordNet (Fellbaum, 1998). Thus, *insurance carriers* and *insurance companies* are considered to not match in MRR$_f$ scores, but match in MRR$_p$ scores. On the other hand, MRR$_p$ scores may give credit to less relevant class labels, such as *insurance policies* for the gold label *insurance carriers*. Therefore, MRR$_p$ is an optimistic, and MRR$_f$ is a pessimistic estimate of the actual usefulness of the computed ranked lists of class labels as approximations of the gold labels.

Query Set: Sample of Queries
Q_e (807 queries): 2009 movies, amino acids, asian countries, bank, board games, buildings, capitals, chemical functional groups, clothes, computer language, dairy farms near modesto ca, disease, egyptian pharaohs, eu countries, fetishes, french presidents, german islands, hawaiian islands, illegal drugs, irc clients, lakes, macintosh models, mobile operator india, nba players, nobel prize winners, orchids, photo editors, programming languages, renaissance artists, roller costers, science fiction tv series, slr cameras, soul singers, states of india, taliban members, thomas edison inventions, u.s. presidents, us president, water slides
Q_m (40 queries): actors, actresses, airlines, american presidents, antibiotics, birds, cars, celebrities, colors, computer languages, digital camera, dog breeds, dogs, drugs, elements, endangered animals, european countries, flowers, fruits, greek gods, horror movies, idioms, ipods, movies, names, netbooks, operating systems, park slope restaurants, planets, presidents, ps3 games, religions, renaissance artists, rock bands, romantic movies, states, universities, university, us cities, vitamins

Table 1: Size and composition of evaluation sets of queries associated with non-filtered (Q_e) or manually-filtered (Q_m) instances

4 Evaluation

IsA Repository: The IsA repository, extracted from the document collection, covers a total of 4.04 million instances associated with 7.65 million class labels. The number of class labels available per instance and vice-versa follows a long-tail distribution, indicating that 2.12 million of the instances each have two or more class labels (with an average of 19.72 class labels per instance).

Evaluation Sets of Queries: Table 1 shows samples of two query sets, introduced in (Paşca, 2010) and used in the evaluation. The first set, denoted $\mathbf{Q}_e$,

Query Set	Min	Max	Avg	Median
Number of Gold Instances:				
Q_e	10	100	70.4	81
Q_m	8	33	16.9	17
Number of Query Tokens:				
Q_e	1	8	2.0	2
Q_m	1	3	1.4	1

Table 2: Number of gold instances (upper part) and number of query tokens (lower part) available per query, over the evaluation sets of queries associated with non-filtered gold instances (Q_e) or manually-filtered gold instances (Q_m)

is obtained from a random sample of anonymized, class-seeking queries submitted by Web users to Google Squared. The set contains 807 queries, each associated with a ranked list of between 10 and 100 gold instances automatically extracted by Google Squared.

Since the gold instances available as input for each query as part of Q_e are automatically extracted, they may or may not be true instances of the respective queries. As described in (Paşca, 2010), the second evaluation set $\mathbf{Q}_m$ is a subset of 40 queries from Q_e, such that the gold instances available for each query in Q_m are found to be correct after manual inspection. The 40 queries from Q_m are associated with between 8 and 33 human-validated instances.

As shown in the upper part of Table 2, the queries from Q_e are up to 8 tokens in length, with an average of 2 tokens per query. Queries from Q_m are comparatively shorter, both in maximum (3 tokens) and average (1.4 tokens) length. The lower part of Table 2 shows the number of gold instances available as input, which average around 70 and 17 per query, for queries from Q_e and Q_m respectively. To provide another view on the distribution of the queries from evaluation sets, Table 3 lists tokens that are not stop words, which occur in most queries from Q_e. Comparatively, few query tokens occur in more than one query in Q_m.

Evaluation Procedure: Following the general evaluation procedure, each query from the sets $\mathbf{Q}_e$ and $\mathbf{Q}_m$ acts as a gold class label associated with the corresponding set of instances. Given a query and its instances $\mathcal{I}$ from the evaluation sets Q_e or Q_m, a merged, ranked lists of class labels is computed out of the ranked lists of class labels available in the underlying IsA repository for each instance $\mathcal{I}$. The evaluation compares the merged lists of class labels, with the corresponding queries from Q_e or Q_m.

Query Token	Cnt.	Examples of Queries Containing the Token
countries	22	african countries, eu countries, poor countries
cities	21	australian cities, cities in california, greek cities
presidents	18	american presidents, korean presidents, presidents of the south korea
restaurants	15	atlanta restaurants, nova scotia restaurants, restaurants 10024
companies	14	agriculture companies, gas utility companies, retail companies
states	14	american states, states of india, united states national parks
prime	11	australian prime ministers, indian prime ministers, prime ministers
cameras	10	cameras, digital cameras olympus, nikon cameras
movies	10	2009 movies, movies, romantic movies
american	9	american authors, american president, american revolution battles
ministers	9	australian prime ministers, indian prime ministers, prime ministers

Table 3: Query tokens occurring most frequently in queries from the Q_e evaluation set, along with the number (Cnt) and examples of queries containing the tokens

Accuracy of Lists of Class Labels: Table 4 summarizes results from comparative experiments, quantifying a) horizontally, the impact of alternative parameter settings on the computed lists of class labels; and b) vertically, the comparative accuracy of the experimental runs over the query sets. The experimental parameters are the number of input instances from the evaluation sets that are used for retrieving class labels, I-per-Q, set to 3, 5, 10; and the number of class labels retrieved per input instance, C-per-I, set to 5, 10, 20.

Four conclusions can be derived from the results. First, the scores over Q_m are higher than those over Q_e, confirming the intuition that the higher-quality

	Accuracy								
I-per-Q	3			5			10		
C-per-I	5	10	20	5	10	20	5	10	20
MRR_f computed over Q_e:									
R_d	0.186	0.195	0.198	0.198	0.207	0.210	0.204	0.214	0.218
R_p	0.202	0.211	0.216	0.232	0.238	0.244	0.245	0.255	0.257
R_s	**0.258**	**0.260**	**0.261**	**0.278**	**0.277**	**0.276**	**0.279**	**0.280**	**0.282**
R_u	0.234	0.241	0.244	0.260	0.263	0.270	0.274	0.275	0.278
MRR_p computed over Q_e:									
R_d	0.489	0.495	0.495	0.517	0.528	0.529	0.541	0.553	0.557
R_p	0.520	0.531	0.533	0.564	0.573	0.578	0.590	0.601	0.602
R_s	**0.576**	**0.584**	**0.583**	**0.612**	**0.616**	0.614	**0.641**	0.636	0.628
R_u	0.561	0.570	0.571	0.606	0.614	**0.617**	0.640	**0.641**	**0.636**
MRR_f computed over Q_m:									
R_d	0.406	0.436	0.442	0.431	0.447	0.466	0.467	0.470	0.501
R_p	0.423	0.426	0.429	0.436	0.483	0.508	0.500	0.526	0.530
R_s	**0.590**	**0.601**	**0.594**	**0.578**	**0.604**	**0.595**	**0.624**	**0.612**	**0.624**
R_u	0.481	0.502	0.508	0.531	0.539	0.545	0.572	0.588	0.575
MRR_p computed over Q_m:									
R_d	0.667	0.662	0.660	0.675	0.677	0.699	0.702	0.695	0.716
R_p	0.711	0.703	0.680	0.734	0.731	0.748	0.733	0.797	0.782
R_s	**0.841**	**0.822**	**0.820**	**0.835**	**0.828**	**0.823**	**0.850**	**0.856**	**0.844**
R_u	0.800	0.810	0.781	0.795	0.794	0.779	0.806	0.827	0.816

Table 4: Accuracy of instance set labeling, as full-match (MRR_f) or partial-match (MRR_p) scores over the evaluation sets of queries associated with non-filtered instances (Q_e) or manually-filtered instances (Q_m), for various experimental runs (I-per-Q=number of gold instances available in the input evaluation sets that are used for retrieving class labels; C-per-I=number of class labels retrieved from IsA repository per input instance)

input set of instances available in Q_m relative to Q_e should lead to higher-quality class labels for the corresponding queries. Second, when I-per-Q is fixed, increasing C-per-I leads to small, if any, score improvements. Third, when C-per-I is fixed, even small values of I-per-Q, such as 3 (that is, very small sets of instances provided as input) produce scores that are competitive with those obtained with a higher value like 10. This suggests that useful class labels can be generated even in extreme scenarios, where the number of instances available as input is as small as 3 or 5. Fourth and most importantly, for most combinations of parameter settings and on both query sets, the runs that take advantage of query logs (R_p, R_s, R_u) produce the highest scores. In particular, when I-per-Q is set to 10 and C-per-I to 20, run R_u identifies the original query as an exact match among the top three to four class labels returned (score 0.278); and as a partial match among the top one to two class labels returned (score 0.636), as an average over the Q_e set. The corresponding MRR_f score of 0.278 over the Q_e set obtained with run R_u is 27% higher than with run R_d.

In all experiments, the higher scores of R_p, R_s and R_u can be attributed to higher-quality lists of class labels, relative to R_d. Among combinations of parameter settings described in Table 4, values around 10 for I-per-Q and 20 for C-per-I give the highest scores over both Q_e and Q_m.

Among the query-based runs R_p, R_s and R_u, the highest scores in Table 4 are obtained mostly for run R_s. Thus, between the presence of a class label and an instance either in the same query, or as separate queries within the same query session, it is the latter that provides a more useful signal during the re-ranking of class labels of each instance.

Table 5 illustrates the top class labels from the ranked lists generated in run R_s for various queries from both Q_e and Q_m. The table suggests that the computed class labels are relatively resistant to noise and variation within the input set of gold instances. For example, the top elements of the lists of class la-

Query	Query Set	Gold Instances		Top Labels Generated Using Top 10 Gold Instances
		Cnt.	Sample from Top Gold Instances	
actors	Q_e	100	abe vigoda, ben kingsley, bill hickman	actors, stars, favorite actors, celebrities, movie stars
	Q_m	28	al pacino, christopher walken, danny devito	actors, celebrities, favorite actors, movie stars, stars
computer languages	Q_e	59	acm transactions on mathematical software, applescript, c	languages, programming languages, programs, standard programming languages, computer programming languages
	Q_m	17	applescript, eiffel, haskell	languages, programming languages, computer languages, modern programming languages, high-level languages
european countries	Q_e	60	abkhazia, armenia, bosnia & herzegovina	countries, european countries, eu countries, foreign countries, western countries
	Q_m	19	belgium, finland, greece	countries, european countries, eu countries, foreign countries, western countries
endangered animals	Q_e	98	arkive, arabian oryx, bagheera	species, animals, endangered species, animal species, endangered animals
	Q_m	21	arabian oryx, blue whale, giant hispaniolan galliwasp	animals, endangered species, species, endangered animals, rare animals
park slope restaurants	Q_e	100	12th street bar & grill, aji bar lounge, anthony's	businesses, departments
	Q_m	18	200 fifth restaurant bar, applewood restaurant, beet thai restaurant	(none)
renaissance artists	Q_e	95	michele da verona, andrea sansovino, andrea del sarto	artists, famous artists, great artists, renaissance artists, italian artists
	Q_m	11	botticelli, filippo lippi, giorgione	artists, famous artists, renaissance artists, great artists, italian artists
rock bands	Q_e	65	blood doll, nightmare, rockaway beach	songs, hits, films, novels, famous songs
	Q_m	15	arcade fire, faith no more, indigo girls	bands, rock bands, favorite bands, great bands, groups

Table 5: Examples of gold instances available in the input, and actual ranked lists of class labels produced by run R_s for various queries from the evaluation sets of queries associated with non-filtered gold instances (Q_e) or manually-filtered gold instances (Q_m)

bels generated for *computer languages* are relevant and also quite similar for Q_e vs. Q_m, although the list of gold instances in Q_e may contain incorrect items (e.g., *acm transactions on mathematical software*). Similarly, the class labels computed for *european countries* are almost the same for Q_e vs. Q_m, although the overlap of the respective lists of 10 gold instances used as input is not large. The table shows at least one query (*park slope restaurants*) for which the output is less than optimal, either because the class labels (e.g., *businesses*) are quite distant semantically from the query (for Q_e), or because no output is produced at all, due to no class labels being found in the IsA repository for any of the 10 input gold instances (for Q_m). For many queries, however, the computed class labels arguably capture the meaning of the original query, although not necessarily in the exact same lexical form, and sometimes only partially. For example, for the query *endangered animals*, only the fourth class label from Q_m identifies the query exactly. However, class labels preceding *endangered animals* already capture the notion of *animals* or *species* (first and third labels), or that they are *endangered* (second label).

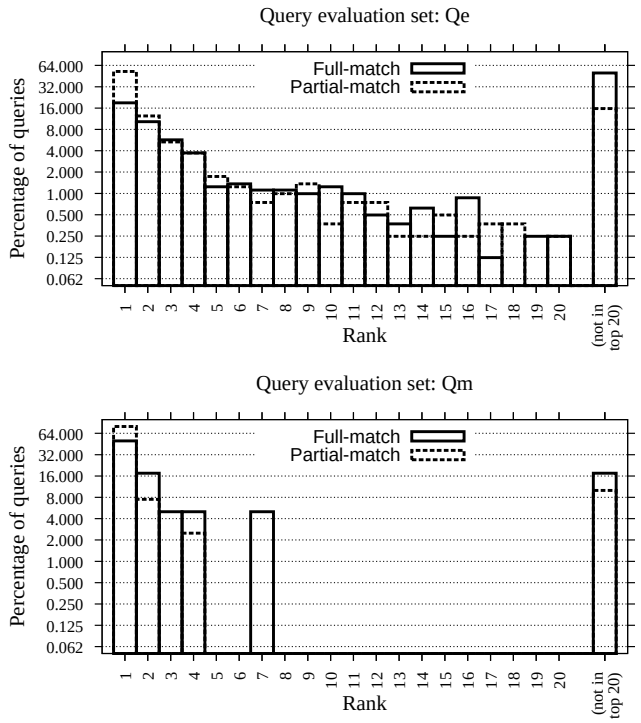


Figure 1: Percentage of queries from the evaluation sets, for which the earliest class labels from the computed ranked lists of class labels, which match the queries, occur at various ranks in the ranked lists returned by run R_s

Figure 1 provides a detailed view on the distribution of queries from the Q_e and Q_m evaluation sets, for which the class label that matches the query occurs at a particular rank in the computed list of class labels. In the first graph of Figure 1, for Q_e, the query matches the automatically-generated class label at ranks 1, 2, 3, 4 and 5 for 18.9%, 10.3%, 5.7%, 3.7% and 1.2% of the queries respectively, with full string matching, i.e., corresponding to MRR_f; and for 52.6%, 12.4%, 5.3%, 3.7% and 1.7% respectively, with partial string matching, corresponding to MRR_p. The second graph confirms that higher MRR scores are obtained for Q_m than for Q_e. In particular, the query matches the class label at rank 1 and 2 for 50.0% and 17.5% (or a combined 67.5%) of the queries from Q_m, with full string matching; and for 52.6% and 12.4% (or a combined 67%), with partial string matching.

Discussion: The quality of lists of items extracted from documents can benefit from query-driven ranking, particularly for the task of ranking class labels of instances within IsA repositories. The use of queries for ranking is generally applicable: it can be seen as a post-processing stage that enhances the ranking of the class labels extracted for various instances by any method into any IsA repository.

Open-domain class labels extracted from text and re-ranked as described in this paper are useful in a variety of applications. Search tools such as Google Squared return a set of instances, in response to class-seeking queries (e.g., *insurance companies*). The labeling of the returned set of instances, using the re-ranked class labels available per instances, allows for the generation of query refinements (e.g., *insurers*). In search over semi-structured data (Cafarella et al., 2008), the labeling of column cells is useful to infer the semantics of a table column, when the subject row of the table in which the column appears is either absent or difficult to detect.

5 Related Work

The role of anonymized query logs in Web-based information extraction has been explored in tasks such as class attribute extraction (Paşca and Van Durme, 2007), instance set expansion (Pennacchiotti and Pantel, 2009) and extraction of sets of similar entities (Jain and Pennacchiotti, 2010). Our work compares the usefulness of queries and query sessions for ranking class labels in extracted IsA repositories. It shows that query sessions produce better-ranked class labels than isolated queries do. A task complementary to class label ranking is entity ranking (Billerbeck et al., 2010), also referred to as ranking for typed search (Demartini et al., 2009).

The choice of search queries and query substitutions is often influenced by, and indicative of, various semantic relations holding among full queries or query terms (Jones et al., 2006). Semantic relations may be loosely defined, e.g., by exploring the acquisition of untyped, similarity-based relations from query logs (Baeza-Yates and Tiberi, 2007). In comparison, queries are used here to re-rank class labels capturing a well-defined type of open-domain relations, namely IsA relations.

6 Conclusion

In an attempt to bridge the gap between information stated in documents and information requested

in search queries, this study shows that inherently-noisy queries are useful in re-ranking class labels extracted from Web documents for various instances, with query sessions leading to higher quality than isolated queries. Current work investigates the impact of ambiguous input instances (Vyas and Pantel, 2009) on the quality of the generated class labels.

References

R. Baeza-Yates and A. Tiberi. 2007. Extracting semantic relations from query logs. In *Proceedings of the 13th ACM Conference on Knowledge Discovery and Data Mining (KDD-07)*, pages 76–85, San Jose, California.

M. Banko, Michael J Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2670–2676, Hyderabad, India.

B. Billerbeck, G. Demartini, C. Firan, T. Iofciu, and R. Krestel. 2010. Ranking entities using Web search query logs. In *Proceedings of the 14th European Conference on Research and Advanced Technology for Digital Libraries (ECDL-10)*, pages 273–281, Glasgow, Scotland.

M. Cafarella, A. Halevy, D. Wang, E. Wu, and Y. Zhang. 2008. WebTables: Exploring the power of tables on the Web. In *Proceedings of the 34th Conference on Very Large Data Bases (VLDB-08)*, pages 538–549, Auckland, New Zealand.

G. Demartini, T. Iofciu, and A. de Vries. 2009. Overview of the INEX 2009 Entity Ranking track. In *INitiative for the Evaluation of XML Retrieval Workshop*, pages 254–264, Brisbane, Australia.

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the Web: an experimental study. *Artificial Intelligence*, 165(1):91–134.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database and Some of its Applications*. MIT Press.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 539–545, Nantes, France.

A. Jain and M. Pennacchiotti. 2010. Open entity extraction from Web search query logs. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING-10)*, pages 510–518, Beijing, China.

R. Jones, B. Rey, O. Madani, and W. Greiner. 2006. Generating query substitutions. In *Proceedings of the 15h World Wide Web Conference (WWW-06)*, pages 387–396, Edinburgh, Scotland.

Z. Kozareva, E. Riloff, and E. Hovy. 2008. Semantic class learning from the Web with hyponym pattern linkage graphs. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pages 1048–1056, Columbus, Ohio.

M. Paşca and B. Van Durme. 2007. What you seek is what you get: Extraction of class attributes from query logs. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2832–2837, Hyderabad, India.

M. Paşca. 2010. The role of queries in ranking labeled instances extracted from text. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING-10)*, pages 955–962, Beijing, China.

M. Pennacchiotti and P. Pantel. 2009. Entity extraction via ensemble semantics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pages 238–247, Singapore.

R. Snow, D. Jurafsky, and A. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*, pages 801–808, Sydney, Australia.

P. Talukdar, J. Reisinger, M. Paşca, D. Ravichandran, R. Bhagat, and F. Pereira. 2008. Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, pages 582–590, Honolulu, Hawaii.

B. Van Durme and M. Paşca. 2008. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI-08)*, pages 1243–1248, Chicago, Illinois.

V. Vyas and P. Pantel. 2009. Semi-automatic entity set refinement. In *Proceedings of the 2009 Conference of the North American Association for Computational Linguistics (NAACL-HLT-09)*, pages 290–298, Boulder, Colorado.

Insights from Network Structure for Text Mining

Zornitsa Kozareva and Eduard Hovy
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
{kozareva,hovy}@isi.edu

Abstract

Text mining and data harvesting algorithms have become popular in the computational linguistics community. They employ patterns that specify the kind of information to be harvested, and usually bootstrap either the pattern learning or the term harvesting process (or both) in a recursive cycle, using data learned in one step to generate more seeds for the next. They therefore treat the source text corpus as a network, in which words are the nodes and relations linking them are the edges. The results of computational network analysis, especially from the world wide web, are thus applicable. Surprisingly, these results have not yet been broadly introduced into the computational linguistics community. In this paper we show how various results apply to text mining, how they explain some previously observed phenomena, and how they can be helpful for computational linguistics applications.

1 Introduction

Text mining / harvesting algorithms have been applied in recent years for various uses, including learning of semantic constraints for verb participants (Lin and Pantel, 2002) related pairs in various relations, such as part-whole (Girju et al., 2003), cause (Pantel and Pennacchiotti, 2006), and other typical information extraction relations, large collections of entities (Soderland et al., 1999; Etzioni et al., 2005), features of objects (Pasca, 2004) and ontologies (Carlson et al., 2010). They generally start with one or more seed terms and employ patterns that specify the desired information as it relates to the seed(s). Several approaches have been developed specifically for learning patterns, including guided pattern collection with manual filtering (Riloff and Shepherd, 1997) automated surface-level pattern induction (Agichtein and Gravano, 2000; Ravichandran and Hovy, 2002) probabilistic methods for taxonomy relation learning (Snow et al., 2005) and kernel methods for relation learning (Zelenko et al., 2003). Generally, the harvesting procedure is recursive, in which data (terms or patterns) gathered in one step of a cycle are used as seeds in the following step, to gather more terms or patterns.

This method treats the source text as a graph or network, consisting of terms (words) as nodes and inter-term relations as edges. Each relation type induces a different network[1]. Text mining is a process of network traversal, and faces the standard problems of handling cycles, ranking search alternatives, estimating yield maxima, etc.

The computational properties of large networks and large network traversal have been studied intensively (Sabidussi, 1966; Freeman, 1979; Watts and Strogatz, 1998) and especially, over the past years, in the context of the world wide web (Page et al., 1999; Broder et al., 2000; Kleinberg and Lawrence, 2001; Li et al., 2005; Clauset et al., 2009). Surprisingly, except in (Talukdar and Pereira, 2010), this work has not yet been related to text mining research in the computational linguistics community.

The work is, however, relevant in at least two ways. It sometimes explains why text mining algo-

[1]These networks are generally far larger and more densely interconnected than the world wide web's network of pages and hyperlinks.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1616–1625,
Portland, Oregon, June 19-24, 2011.

rithms have the limitations and thresholds that are empirically found (or suspected), and it may suggest ways to improve text mining algorithms for some applications.

In Section 2, we review some related work. In Section 3 we describe the general harvesting procedure, and follow with an examination of the various statistical properties of implicit semantic networks in Section 4, using our implemented harvester to provide illustrative statistics. In Section 5 we discuss implications for computational linguistics research.

2 Related Work

The Natural Language Processing knowledge harvesting community has developed a good understanding of how to harvests various kinds of semantic information and use this information to improve the performance of tasks such as information extraction (Riloff, 1993), textual entailment (Zanzotto et al., 2006), question answering (Katz et al., 2003), and ontology creation (Suchanek et al., 2007), among others. Researchers have focused on the automated extraction of semantic lexicons (Hearst, 1992; Riloff and Shepherd, 1997; Girju et al., 2003; Pasca, 2004; Etzioni et al., 2005; Kozareva et al., 2008). While clustering approaches tend to extract general facts, pattern based approaches have shown to produce more constrained but accurate lists of semantic terms. To extract this information, (Lin and Pantel, 2002) showed the effect of using different sizes and genres of corpora such as news and Web documents. The latter has been shown to provide broader and more complete information.

Researchers outside computational linguistics have studied complex networks such as the World Wide Web, the Social Web, the network of scientific papers, among others. They have investigated the properties of these text-based networks with the objective of understanding their structure and applying this knowledge to determine node importance/centrality, connectivity, growth and decay of interest, etc. In particular, the ability to analyze networks, identify influential nodes, and discover hidden structures has led to important scientific and technological breakthroughs such as the discovery of communities of like-minded individuals (Newman and Girvan, 2004), the identification of influential people (Kempe et al., 2003), the ranking of scientists by their citation indexes (Radicchi et al., 2009), and the discovery of important scientific papers (Walker et al., 2006; Chen et al., 2007; Sayyadi and Getoor, 2009). Broder et al. (2000) demonstrated that the Web link structure has a "bow-tie" shape, while (2001) classified Web pages into *authorities* (pages with relevant information) and *hubs* (pages with useful references). These findings resulted in the development of the PageRank (Page et al., 1999) algorithm which analyzes the structure of the hyperlinks of Web documents to find pages with authoritative information. PageRank has revolutionized the whole Internet search society.

However, no-one has studied the properties of the text-based semantic networks induced by semantic relations between terms with the objective of understanding their structure and applying this knowledge to improve concept discovery. Most relevant to this theme is the work of Steyvers and Tenenbaum (Steyvers and Tenenbaum, 2004), who studied three manually built lexical networks (association norms, WordNet, and Roget's Thesaurus (Roget, 1911)) and proposed a model of the growth of the semantic structure over time. These networks are limited to the semantic relations among nouns.

In this paper we take a step further to explore the statistical properties of semantic networks relating *proper names*, *nouns*, *verbs*, and *adjectives*. Understanding the semantics of nouns, verbs, and adjectives has been of great interest to linguists and cognitive scientists such as (Gentner, 1981; Levin and Somers, 1993; Gasser and Smith, 1998). We implement a general harvesting procedure and show its results for these word types. A fundamental difference with the work of (Steyvers and Tenenbaum, 2004) is that we study very large semantic networks built 'naturally' by (millions of) users rather than 'artificially' by a small set of experts. The large networks capture the semantic intuitions and knowledge of the collective mass. It is conceivable that an analysis of this knowledge can begin to form the basis of a large-scale theory of semantic meaning and its interconnections, support observation of the process of lexical development and usage in humans, and even suggest explanations of how knowledge is organized in our brains, especially when performed for differ-

ent languages on the WWW.

3 Inducing Semantic Networks in the Web

Text mining algorithms such as those mentioned above raise certain questions, such as: *Why are some seed terms more powerful (provide a greater yield) than others?*, *How can one find high-yield terms?*, *How many steps does one need, typically, to learn all terms for a given relation?*, *Can one estimate the total eventual yield of a given relation?*, and so on. On the face of it, one would need to know the structure of the network a priori to be able to provide answers. But research has shown that some surprising regularities hold. For example, in the text mining community, (Kozareva and Hovy, 2010b) have shown that one can obtain a quite accurate estimate of the eventual yield of a pattern and seed after only five steps of harvesting. Why is this? They do not provide an answer, but research from the network community does.

To illustrate the properties of networks of the kind induced by semantic relations, and to show the applicability of network research to text harvesting, we implemented a harvesting algorithm and applied it to a representative set of relations and seeds in two languages.

Since the goal of this paper is not the development of a new text harvesting algorithm, we implemented a version of an existing one: the so-called DAP (doubly-anchored pattern) algorithm (Kozareva et al., 2008), because it (1) is easy to implement, (2) requires minimum input (one pattern and one seed example), (3) achieves very high precision compared to existing methods (Pasca, 2004; Etzioni et al., 2005; Pasca, 2007), (4) enriches existing semantic lexical repositories such as WordNet and Yago (Suchanek et al., 2007), (5) can be formulated to learn semantic lexicons and relations for *noun*, *verb* and *verb+preposition* syntactic constructions; (6) functions equally well in different languages. Next we describe the knowledge harvesting procedure and the construction of the text-mined semantic networks.

3.1 Harvesting to Induce Semantic Networks

For a given semantic class of interest say *singers*, the algorithm starts with a *seed* example of the *class*, say *Madonna*. The *seed* term is inserted in the lexico-syntactic pattern "*class* such as *seed* and *", which learns on the position of the $*$ new terms of type *class*. The newly learned terms are then individually placed into the position of the *seed* in the pattern, and the bootstrapping process is repeated until no new terms are found. The output of the algorithm is a set of terms for the semantic class. The algorithm is implemented as a breadth-first search and its mechanism is described as follows:

1. Given:
   a language L={English, Spanish}
   a pattern P_i={such as, including, verb prep, noun}
   a seed term $seed$ for P_i
2. Build a query for P_i using template T_i 'class such as $seed$ and *', 'class including $seed$ and *', '* and $seed$ verb prep', '* and $seed$ noun', '$seed$ and * noun'
3. Submit T_i to Yahoo! or other search engine
4. Extract terms occupying the * position
5. Feed terms from 4. into 2.
6. Repeat steps 2–5. until no new terms are found

The output of the knowledge harvesting algorithm is a network of semantic terms interconnected by the semantic relation captured in the pattern. We can represent the traversed (implicit) network as a directed graph $G(V, E)$ with nodes $V(|V| = n)$ and edges $E(|E| = m)$. A node u in the network corresponds to a term discovered during bootstrapping. An edge $(u, v) \in E$ represents an existing link between two terms. The direction of the edge indicates that the term v was generated by the term u. For example, given the sentence (where the pattern is in italics and the extracted term is underlined) "He loves *singers such as Madonna and* Michael Jackson", two nodes *Madonna* and *Michael Jackson* with an edge e=(*Madonna*, *Michael Jackson*) would be created in the graph G. Figure 1 shows a small example of the singer network. The starting seed term *Madonna* is shown in red color and the harvested terms are in blue.

3.2 Data

We harvested data from the Web for a representative selection of semantic classes and relations, of

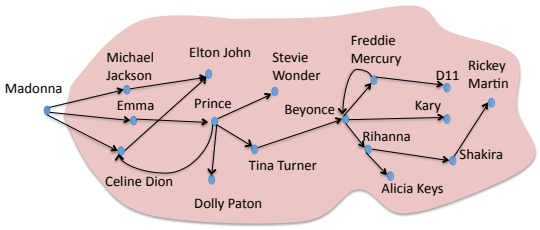


Figure 1: Harvesting Procedure.

the type used in (Etzioni et al., 2005; Pasca, 2007; Kozareva and Hovy, 2010a):

- semantic classes that can be learned using different seeds (e.g., "singers such as **Madonna** and *" and "singers such as **Placido Domingo** and *");
- semantic classes that are expressed through different lexico-syntactic patterns (e.g., "weapons **such as** bombs and *" and "weapons **including** bombs and *");
- verbs and adjectives characterizing the semantic class (e.g., "**expensive** and * car", "dogs **run** and *");
- semantic relations with more complex lexico-syntactic structure (e.g., "* and Easyjet **fly to**", "* and Sam **live in**");
- semantic classes that are obtained in different languages, such as English and Spanish (e.g., "**singers** such as Madonna and *" and "**cantantes** como Madonna y *");

While most of these variations have been explored in individual papers, we have found no paper that covers them all, and none whatsoever that uses verbs and adjectives as seeds.

Using the above procedure to generate the data, each pattern was submitted as a query to Yahoo!Boss. For each query the top 1000 text snippets were retrieved. The algorithm ran until exhaustion. In total, we collected 10GB of data which was part-of-speech tagged with Treetagger (Schmid, 1994) and used for the semantic term extraction. Table 1 summarizes the number of nodes and edges learned for each semantic network using pattern P_i and the initial seed shown in italics.

Lexico-Syntactic Pattern	Nodes	Edges
P_1="**singers** such as *Madonna* and *"	1115	1942
P_2="**singers** such as *Placido Domingo* and *"	815	1114
P_3="**emotions** including *anger* and *"	113	250
P_4="**emotions** such as *anger* and *"	748	2547
P_5="**diseases** such as *malaria* and *"	3168	6752
P_6="**drugs** such as *ibuprofen* and *"	2513	9428
P_7="*expensive* and * **cars**"	4734	22089
P_8="* and *tasty* **fruits**"	1980	7874
P_9="**whales** *swim* and *"	869	2163
P_{10}="**dogs** *chase* and *"	4252	20212
P_{11}="**Britney Spears** *dances* and *"	354	540
P_{12}="**John** *reads* and *"	3894	18545
P_{13}="* and *Easyjet* **fly to**"	3290	6480
P_{14}="* and *Charlie* **work for**"	2125	3494
P_{15}="* and *Sam* **live in**"	6745	24348
P_{16}="**cantantes** como *Madonna* y *"	240	318
P_{17}="**gente** como *Jorge* y *"	572	701

Table 1: *Size of the Semantic Networks.*

4 Statistical Properties of Text-Mined Semantic Networks

In this section we apply a range of relevant measures from the network analysis community to the networks described above.

4.1 Centrality

The first statistical property we explore is centrality. It measures the degree to which the network structure determines the importance of a node in the network (Sabidussi, 1966; Freeman, 1979).

We explore the effect of two centrality measures: $indegree$ and $outdegree$. The $indegree$ of a node u denoted as $indegree(u)$=$\sum(v,u)$ considers the sum of all incoming edges to u and captures the ability of a semantic term to be discovered by other semantic terms. The $outdegree$ of a node u denoted as $outdegree(u)$=$\sum(u,v)$ considers the number of outgoing edges of the node u and measures the ability of a semantic term to discover new terms. Intuitively, the more central the node u is, the more confident we are that it is a correct term.

Since harvesting algorithms are notorious for extracting erroneous information, we use the two centrality measures to rerank the harvested elements. Table 2 shows the accuracy[2] of the singer semantic terms at different ranks using the in and out degree measures. Consistently, $outdegree$ outperforms $indegree$ and reaches higher accuracy. This

[2]Accuracy is calculated as the number of correct terms at rank R divided by the total number of terms at rank R.

shows that for the text-mined semantic networks, the ability of a term to discover new terms is more important than the ability to be discovered.

@rank	in-degree	out-degree
10	.92	1.0
25	.91	1.0
50	.90	.97
75	.90	.96
100	.89	.96
150	.88	.95

Table 2: *Accuracy of the Singer Terms.*

This poses the question "*What are the terms with high and low outdegree?*". Table 3 shows the top and bottom 10 terms of the semantic class.

Semantic Class	top 10 outDegree	bottom 10 outDegree
Singers	Frank Sinatra	Alanis Morisette
	Ella Fitzgerald	Christine Agulera
	Billie Holiday	Buffy Sainte-Marie
	Britney Spears	Cece Winans
	Aretha Franklin	Wolfman Jack
	Michael Jackson	Billie Celebration
	Celine Dion	Alejandro Sanz
	Beyonce	France Gall
	Bessie Smith	Peter
	Joni Mitchell	Sarah

Table 3: *Singer Term Ranking with Centrality Measures.*

The nodes with high outdegree correspond to famous or contemporary singers. The lower-ranked nodes are mostly spelling errors such as *Alanis Morisette* and *Christine Agulera*, less known singers such as *Buffy Sainte-Marie* and *Cece Winans*, non-American singers such as *Alejandro Sanz* and *France Gall*, extractions due to part-of-speech tagging errors such as *Billie Celebration*, and general terms such as *Peter* and *Sarah*. Potentially, knowing which terms have a high *outdegree* allows one to rerank candidate seeds for more effective harvesting.

4.2 Power-law Degree Distribution

We next study the degree distributions of the networks. Similarly to the Web (Broder et al., 2000) and social networks like Orkut and Flickr, the text-mined semantic networks also exhibit a power-law distribution. This means that while a few terms have a significantly high degree, the majority of the semantic terms have small degree. Figure 2 shows the $indegree$ and $outdegree$ distributions for different semantic classes, lexico-syntactic patterns, and languages (English and Spanish). For each semantic network, we plot the best-fitting power-law function (Clauset et al., 2009) which fits well all degree distributions. Table 4 shows the power-law exponent values for all text-mined semantic networks.

Patt.	γ_{in}	γ_{out}	Patt.	γ_{in}	γ_{out}
P_1	2.37	1.27	P_{10}	1.65	1.12
P_2	2.25	1.21	P_{11}	2.42	1.41
P_3	2.20	1.76	P_{12}	1.60	1.13
P_4	2.28	1.18	P_{13}	2.26	1.20
P_5	2.49	1.18	P_{14}	2.43	1.25
P_6	2.42	1.30	P_{15}	2.51	1.43
P_7	1.95	1.20	P_{16}	2.74	1.31
P_8	1.94	1.07	P_{17}	2.90	1.20
P_9	1.96	1.30			

Table 4: *Power-Law Exponents of Semantic Networks.*

It is interesting to note that the $indegree$ power-law exponents for all semantic networks fall within the same range ($\gamma_{in} \approx 2.4$), and similarly for the $outdegree$ exponents ($\gamma_{out} \approx 1.3$). However, the values of the $indegree$ and $outdegree$ exponents differ from each other. This observation is consistent with Web degree distributions (Broder et al., 2000). The difference in the distributions can be explained by the link asymmetry of semantic terms: A discovering B does not necessarily mean that B will discover A. In the text-mined semantic networks, this asymmetry is caused by patterns of language use, such as the fact that people use first adjectives of the size and then of the color (e.g., big red car), or prefer to place male before female proper names. Harvesting patterns should take into account this tendency.

4.3 Sparsity

Another relevant property of the semantic networks concerns sparsity. Following Preiss (Preiss, 1999), a graph is sparse if $|E| = O(|V|^k)$ and $1 < k < 2$, where $|E|$ is the number of edges and $|V|$ is the number of nodes, otherwise the graph is dense. For the studied text-semantic networks, k is ≈ 1.08. Sparsity can be also captured through the density of the semantic network which is computed as $\frac{|E|}{V(V-1)}$. All networks have low density which suggests that the networks exhibit a sparse connectivity pattern. On average a node (semantic term) is connected to a very small percentage of other nodes. Similar behavior was reported for the WordNet and Roget's semantic networks (Steyvers and Tenenbaum, 2004).

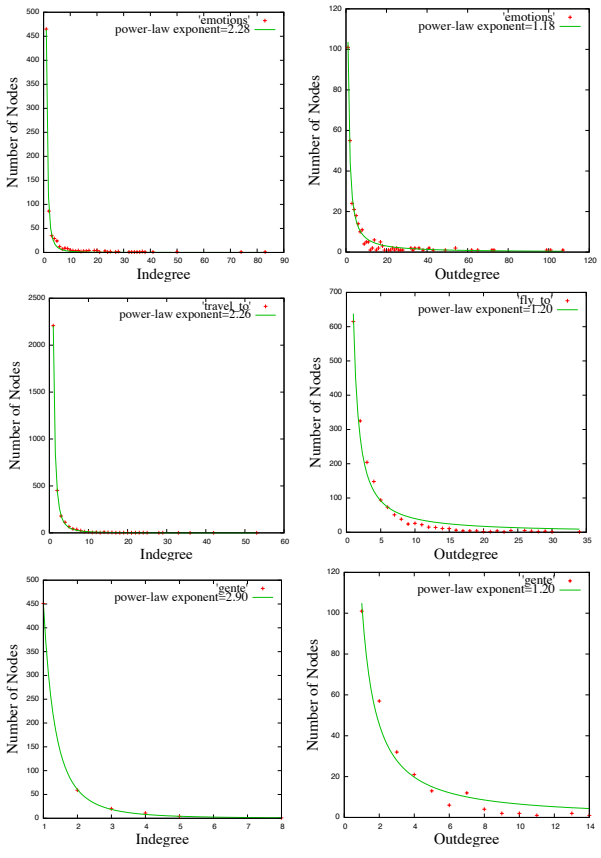


Figure 2: Degree Distributions of Semantic Networks.

4.4 Connectedness

For every network, we computed the strongly connected component (SCC) such that for all nodes (semantic terms) in the SCC, there is a path from any node to another node in the SCC considering the direction of the edges between the nodes. For each network, we found that there is only one SCC. The size of the component is shown in Table 5. Unlike WordNet and Roget's semantic networks where the SCC consists 96% of all semantic terms, in the text-mined semantic networks only 12 to 55% of the terms are in the SCC. This shows that not all nodes can reach (discover) every other node in the network. This also explains the findings of (Kozareva et al., 2008; Vyas et al., 2009) why starting with a good seed is important.

4.5 Path Lengths and Diameter

Next, we describe the properties of the shortest paths between the semantic terms in the SCC. The distance between two nodes in the SCC is measured as the length of the shortest path connecting the terms. The direction of the edges between the terms is taken into consideration. The average distance is the average value of the shortest path lengths over all pairs of nodes in the SCC. The diameter of the SCC is calculated as the maximum distance over all pairs of nodes (u, v), such that a node v is reachable from node u. Table 5 shows the average distance and the diameter of the semantic networks.

Patt.	#nodes in SCC	SCC Average Distance	SCC Diameter
P_1	364 (.33)	5.27	16
P_2	285 (.35)	4.65	13
P_3	48 (.43)	2.85	6
P_4	274 (.37)	2.94	7
P_5	1249 (.38)	5.99	17
P_6	1471 (.29)	4.82	15
P_7	2255 (.46)	3.51	11
P_8	1012 (.50)	3.87	11
P_9	289 (.33)	4.93	13
P_{10}	2342 (.55)	4.50	12
P_{11}	87 (.24)	5.00	11
P_{12}	1967 (.51)	3.20	13
P_{13}	1249 (.38)	4.75	13
P_{14}	608 (.29)	7.07	23
P_{15}	1752 (.26)	5.32	15
P_{16}	56 (.23)	4.79	12
P_{17}	69 (.12)	5.01	13

Table 5: *SCC, SCC Average Distance and SCC Diameter of the Semantic Networks.*

The diameter shows the maximum number of steps necessary to reach from any node to any other, while the average distance shows the number of steps necessary on average. Overall, all networks have very short average path lengths and small diameters that are consistent with Watt's finding for small-world networks. Therefore, the yield of harvesting seeds can be predicted within five steps explaining (Kozareva and Hovy, 2010b; Vyas et al., 2009).

We also compute for any randomly selected node in the semantic network on average how many hops (steps) are necessary to reach from one node to another. Figure 3 shows the obtained results for some of the studied semantic networks.

4.6 Clustering

The clustering coefficient (C) is another measure to study the connectivity structure of the networks (Watts and Strogatz, 1998). This measure captures the probability that the two neighbors of a randomly selected node will be neighbors. The clustering coefficient of a node u is calculated as $C_u = \frac{|e_{ij}|}{k_u(k_u-1)}$

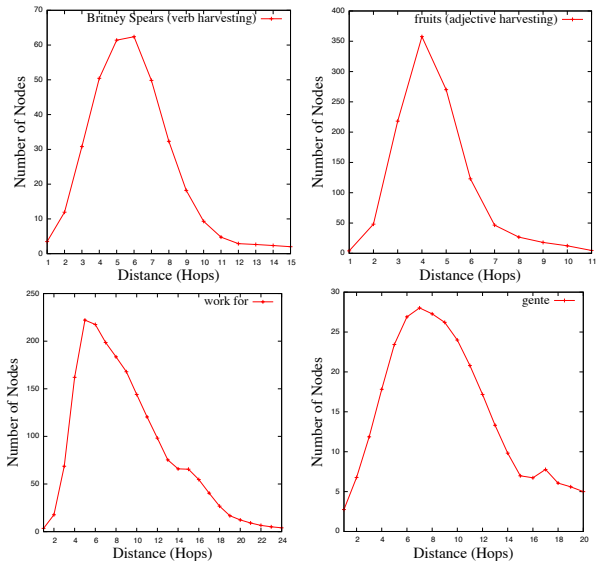


Figure 3: Hop Plot of the Semantic Networks.

Patt.	C	ClosedTriads	OpenTriads
P_1	.01	14096 (.97)	388 (.03)
P_2	.01	6487 (.97)	213 (.03)
P_3	.30	1898 (.94)	129 (.06)
P_4	.33	60734 (.94)	3944 (.06)
P_5	.10	79986 (.97)	2321 (.03)
P_6	.11	78716 (.97)	2336 (.03)
P_7	.17	910568 (.95)	43412 (.05)
P_8	.19	21138 (.95)	10728 (.05)
P_9	.20	27830 (.95)	1354 (.05)
P_{10}	.15	712227 (.96)	62101(.04)
P_{11}	.09	3407 (.98)	63 (.02)
P_{12}	.15	734724 (.96)	32517 (.04)
P_{13}	.06	66162 (.99)	858 (.01)
P_{14}	.05	28216 (.99)	408 (.01)
P_{15}	.09	1336679 (.97)	47110 (.03)
P_{16}	.09	1525 (.98)	37 (.02)
P_{17}	.05	2222 (.99)	21 (.01)

Table 6: *Clustering Coefficient of the Semantic Networks.*

: $v_i, v_j \in N_u, e_{ij} \in E$, where k_u is the total degree of the node u and N_u is the neighborhood of u. The clustering coefficient C for the whole semantic network is the average clustering coefficient of all its nodes, C=$\frac{1}{n}\sum C_i$. The value of the clustering coefficient ranges between $[0, 1]$, where 0 indicates that the nodes do not have neighbors which are themselves connected, while 1 indicates that all nodes are connected. Table 6 shows the clustering coefficient for all text-mined semantic networks together with the number of closed and open triads[3]. The analysis suggests the presence of a strong local cluster, however there are few possibilities to form overlapping neighborhoods of nodes. The clustering coefficient of WordNet (Steyvers and Tenenbaum, 2004) is similar to those of the text-mined networks.

4.7 Joint Degree Distribution

In social networks, understanding the preferential attachment of nodes is important to identify the speed with which epidemics or gossips spread. Similarly, we are interested in understanding how the nodes of the semantic networks connect to each other. For this purpose, we examine the Joint Degree Distribution (JDD) (Li et al., 2005; Newman, 2003). JDD is approximated by the degree correlation function k_{nn} which maps the $outdegree$ and the average $indegree$ of all nodes connected to a node with that $outdegree$. High values of k_{nn} indicate that high-degree nodes tend to connect to other high-degree nodes (forming a "core" in the network), while lower values of k_{nn} suggest that the high-degree nodes tend to connect to low-degree ones. Figure 4 shows the k_{nn} for the *singer*, *whale*, *live in*, *cars*, *cantantes*, and *gente* networks. The figure plots the outdegree and the average indegree of the semantic terms in the networks on a log-log scale. We can see that for all networks the high-degree nodes tend to connect to other high-degree ones. This explains why text mining algorithms should focus their effort on high-degree nodes.

4.8 Assortivity

The property of the nodes to connect to other nodes with similar degrees can be captured through the assortivity coefficient r (Newman, 2003). The range of r is $[-1, 1]$. A positive assortivity coefficient means that the nodes tend to connect to nodes of similar degree, while negative coefficient means that nodes are likely to connect to nodes with degree very different from their own. We find that the assortivitiy coefficient of our semantic networks is positive, ranging from 0.07 to 0.20. In this respect, the semantic networks differ from the Web, which has a negative assortivity (Newman, 2003). This implies a difference in text mining and web search traversal strategies: since starting from a highly-connected seed term will tend to lead to other highly-connected terms, text mining algorithms should prefer depth-first traversal, while web search algorithms starting

[3] A triad is three nodes that are connected by either two (open triad) or three (closed triad) directed ties.

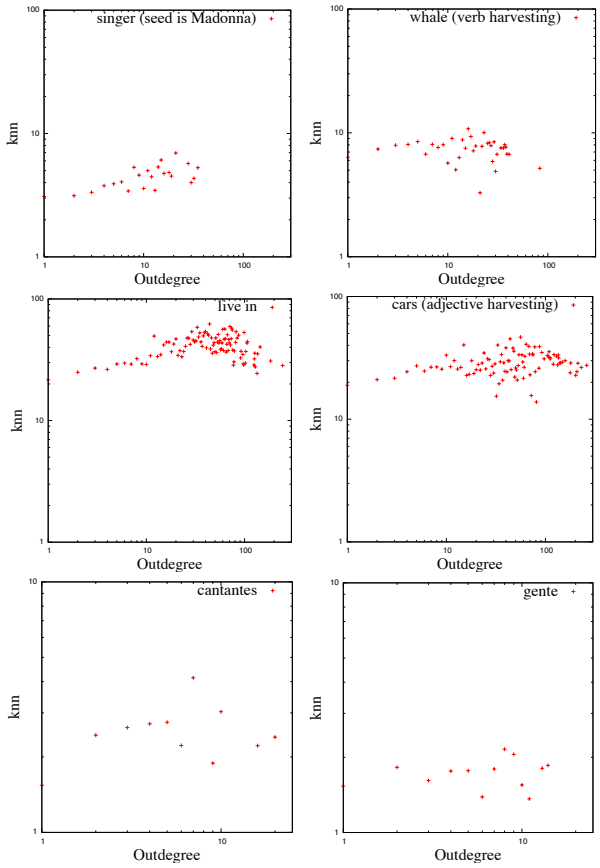


Figure 4: Joint Degree Distribution of the Semantic Networks.

from a highly-connected seed page should prefer a breadth-first strategy.

5 Discussion

The above studies show that many of the properties discovered of the network formed by the web hold also for the networks induced by semantic relations in text mining applications, for various semantic classes, semantic relations, and languages. We can therefore apply some of the research from network analysis to text mining.

The small-world phenomenon, for example, holds that any node is connected to any other node in at most six steps. Since as shown in Section 4.5 the semantic networks also exhibit this phenomenon, we can explain the observation of (Kozareva and Hovy, 2010b) that one can quite accurately predict the relative 'goodness' of a seed term (its eventual total yield and the number of steps required to obtain that) within five harvesting steps. We have shown that due to the strongly connected components in text mining networks, not all elements within the harvested graph can discover each other. This implies that harvesting algorithms have to be started with several seeds to obtain adequate Recall (Vyas et al., 2009). We have shown that centrality measures can be used successfully to rank harvested terms to guide the network traversal, and to validate the correctness of the harvested terms.

In the future, the knowledge and observations made in this study can be used to model the lexical usage of people over time and to develop new semantic search technology.

6 Conclusion

In this paper we describe the implicit 'hidden' semantic network graph structure induced over the text of the web and other sources by the semantic relations people use in sentences. We describe how term harvesting patterns whose seed terms are harvested and then applied recursively can be used to discover these semantic term networks. Although these networks differ considerably from the web in relation density, type, and network size, we show, somewhat surprisingly, that the same power-law, small-world effect, transitivity, and most other characteristics that apply to the web's hyperlinked network structure hold also for the implicit semantic term graphs—certainly for the semantic relations and languages we have studied, and most probably for almost all semantic relations and human languages.

This rather interesting observation leads us to surmise that the hyperlinks people create in the web are of essentially the same type as the semantic relations people use in normal sentences, and that they form an extension of normal language that was not needed before because people did not have the ability within the span of a single sentence to 'embed' structures larger than a clause—certainly not a whole other page's worth of information. The principal exception is the academic citation reference (lexicalized as "see"), which is not used in modern webpages. Rather, the 'lexicalization' now used is a formatting convention: the hyperlink is colored and often underlined, facilities offered by computer screens but not available to speech or easy in traditional typesetting.

Acknowledgments

We acknowledge the support of DARPA contract number FA8750-09-C-3705 and NSF grant IIS-0429360. We would like to thank Sujith Ravi for his useful comments and suggestions.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. pages 85–94.

Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. 2000. Graph structure in the web. *Comput. Netw.*, 33(1-6):309–320.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. pages 101–110.

Peng Chen, Huafeng Xie, Sergei Maslov, and Sid Redner. 2007. Finding scientific gems with google's pagerank algorithm. *Journal of Informetrics*, 1(1):8–15, January.

Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. 2009. Power-law distributions in empirical data. *SIAM Rev.*, 51(4):661–703.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: an experimental study. *Artificial Intelligence*, 165(1):91–134, June.

Linton Freeman. 1979. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239.

Michael Gasser and Linda B. Smith. 1998. Learning nouns and adjectives: A connectionist account. In *Language and Cognitive Processes*, pages 269–306.

Demdre Gentner. 1981. Some interesting differences between nouns and verbs. *Cognition and Brain Theory*, pages 161–178.

Roxana Girju, Adriana Badulescu, and Dan Moldovan. 2003. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 1–8.

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

Boris Katz, Jimmy Lin, Daniel Loreto, Wesley Hildebrandt, Matthew Bilotti, Sue Felshin, Aaron Fernandes, Gregory Marton, and Federico Mora. 2003. Integrating web-based and corpus-based techniques for question answering. In *Proceedings of the twelfth text retrieval conference (TREC)*, pages 426–435.

David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146.

Jon Kleinberg and Steve Lawrence. 2001. The structure of the web. *Science*, 29:1849–1850.

Zornitsa Kozareva and Eduard Hovy. 2010a. Learning arguments and supertypes of semantic relations using recursive patterns. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL 2010*, pages 1482–1491, July.

Zornitsa Kozareva and Eduard Hovy. 2010b. Not all seeds are equal: Measuring the quality of text mining seeds. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 618–626.

Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics ACL-08: HLT*, pages 1048–1056.

Beth Levin and Harold Somers. 1993. English verb classes and alternations: A preliminary investigation.

Lun Li, David Alderson, Reiko Tanaka, John C. Doyle, and Walter Willinger. 2005. Towards a Theory of Scale-Free Graphs: Definition, Properties, and Implications (Extended Version). *Internet Mathematica*, 2(4):431–523.

Dekang Lin and Patrick Pantel. 2002. Concept discovery from text. In *Proc. of the 19th international conference on Computational linguistics*, pages 1–7.

Mark E. Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Physical Review*, 69(2).

Mark Newman. 2003. Mixing patterns in networks. *Physical Review E*, 67.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. pages 113–120.

Marius Pasca. 2004. Acquisition of categorized named entities for web search. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 137–145.

Marius Pasca. 2007. Weakly-supervised discovery of named entities using web search queries. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007*, pages 683–690.

Bruno R. Preiss. 1999. *Data structures and algorithms with object-oriented design patterns in C++*.

Filippo Radicchi, Santo Fortunato, Benjamin Markines, and Alessandro Vespignani. 2009. Diffusion of scientific credits and the ranking of scientists. In *Phys. Rev. E 80, 056103*.

Deepack Ravichandran and Eduard H. Hovy. 2002. Learning surface text patterns for a question answering system. pages 41–47.

Ellen Riloff and Jessica Shepherd. 1997. A corpus-based approach for building semantic lexicons. In *Proceedings of the Empirical Methods for Natural Language Processing*, pages 117–124.

Ellen Riloff. 1993. Automatically constructing a dictionary for information extraction tasks. pages 811–816.

Peter Mark Roget. 1911. *Roget's thesaurus of English Words and Phrases*. New York Thomas Y. Crowell company.

Gert Sabidussi. 1966. The centrality index of a graph. *Psychometrika*, 31(4):581–603.

Hassan Sayyadi and Lise Getoor. 2009. Future rank: Ranking scientific articles by predicting their future pagerank. In *2009 SIAM International Conference on Data Mining (SDM09)*.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. pages 1297–1304.

Stephen Soderland, Claire Cardie, and Raymond Mooney. 1999. Learning information extraction rules for semi-structured and free text. *Machine Learning, 34(1-3)*, pages 233–272.

Mark Steyvers and Joshua B. Tenenbaum. 2004. The large-scale structure of semantic networks: Statistical analyses and a model of semantic growth. *Cognitive Science*, 29:41–78.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 697–706.

Partha Pratim Talukdar and Fernando Pereira. 2010. Graph-based weakly-supervised methods for information extraction and integration. pages 1473–1481.

Vishnu Vyas, Patrick Pantel, and Eric Crestan. 2009. Helping editors choose better seed sets for entity set expansion. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM*, pages 225–234.

Dylan Walker, Huafeng Xie, Koon-Kiu Yan, and Sergei Maslov. 2006. Ranking scientific publications using a simple model of network traffic. December.

Duncan Watts and Steven Strogatz. 1998. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442.

Fabio Massimo Zanzotto, Marco Pennacchiotti, and Maria Teresa Pazienza. 2006. Discovering asymmetric entailment relations between verbs using selectional preferences. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 849–856.

Dmitry Zelenko, Chinatsu Aone, Anthony Richardella, Jaz K, Thomas Hofmann, Tomaso Poggio, and John Shawe-taylor. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research 3*.

Event Extraction as Dependency Parsing

David McClosky, Mihai Surdeanu, and Christopher D. Manning
Department of Computer Science
Stanford University
Stanford, CA 94305
{mcclosky,mihais,manning}@stanford.edu

Abstract

Nested event structures are a common occurrence in both open domain and domain specific extraction tasks, e.g., a "crime" event can cause a "investigation" event, which can lead to an "arrest" event. However, most current approaches address event extraction with highly local models that extract each event and argument independently. We propose a simple approach for the extraction of such structures by taking the tree of event-argument relations and using it directly as the representation in a reranking dependency parser. This provides a simple framework that captures global properties of both nested and flat event structures. We explore a rich feature space that models both the events to be parsed and context from the original supporting text. Our approach obtains competitive results in the extraction of biomedical events from the BioNLP'09 shared task with a F1 score of 53.5% in development and 48.6% in testing.

1 Introduction

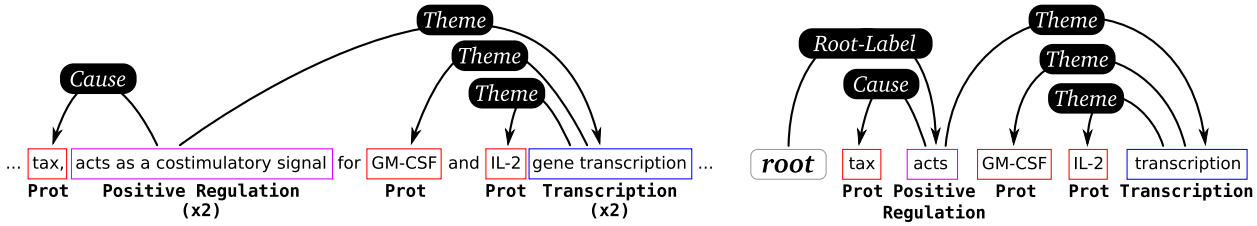
Event structures in open domain texts are frequently highly complex and nested: a "crime" event can cause an "investigation" event, which can lead to an "arrest" event (Chambers and Jurafsky, 2009). The same observation holds in specific domains. For example, the BioNLP'09 shared task (Kim et al., 2009) focuses on the extraction of nested biomolecular events, where, e.g., a REGULATION event causes a TRANSCRIPTION event (see Figure 1a for a detailed example). Despite this observation, many state-of-the-art supervised event extraction models still extract events and event arguments independently, ignoring their underlying structure (Björne et al., 2009; Miwa et al., 2010b).

In this paper, we propose a new approach for supervised event extraction where we take the tree of relations and their arguments and use it directly as the representation in a dependency parser (rather than conventional syntactic relations). Our approach is conceptually simple: we first convert the original representation of events and their arguments to dependency trees by creating dependency arcs between *event anchors* (phrases that anchor events in the supporting text) and their corresponding arguments.[1] Note that after conversion, only event anchors and entities remain. Figure 1 shows a sentence and its converted form from the biomedical domain with four events: two POSITIVE REGULATION events, anchored by the phrase "acts as a costimulatory signal," and two TRANSCRIPTION events, both anchored on "gene transcription." All events take either protein entity mentions (PROT) or other events as arguments. The latter is what allows for nested event structures. Existing dependency parsing models can be adapted to produce these semantic structures instead of syntactic dependencies. We built a global reranking parser model using multiple decoders from MSTParser (McDonald et al., 2005; McDonald et al., 2005b). The main contributions of this paper are the following:

1. We demonstrate that parsing is an attractive approach for extracting events, both nested and otherwise.

[1] While our approach only works on trees, we show how we can handle directed acyclic graphs in Section 5.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1626–1635,
Portland, Oregon, June 19-24, 2011.



(a) Original sentence with nested events

(b) After conversion to event dependencies

Figure 1: Nested events in the text fragment: "...the *HTLV-1 transactivator protein*, *tax*, **acts as a costimulatory signal** for *GM-CSF* and *IL-2* **gene transcription** ..." Throughout this paper, **bold text** indicates instances of event anchors and *italicized text* denotes entities (PROTEINs in the BioNLP'09 domain). Note that in (a) there are two copies of each type of event, which are merged to single nodes in the dependency tree (Section 3.1).

2. We propose a wide range of features for event extraction. Our analysis indicates that features which model the global event structure yield considerable performance improvements, which proves that modeling event structure jointly is beneficial.
3. We evaluate on the biomolecular event corpus from the the BioNLP'09 shared task and show that our approach obtains competitive results.

2 Related Work

The pioneering work of Miller et al. (1997) was the first, to our knowledge, to propose parsing as a framework for information extraction. They extended the syntactic annotations of the Penn Treebank corpus (Marcus et al., 1993) with entity and relation mentions specific to the MUC-7 evaluation (Chinchor et al., 1997) — e.g., EMPLOYEE OF relations that hold between person and organization named entities — and then trained a generative parsing model over this combined syntactic and semantic representation. In the same spirit, Finkel and Manning (2009) merged the syntactic annotations and the named entity annotations of the OntoNotes corpus (Hovy et al., 2006) and trained a discriminative parsing model for the joint problem of syntactic parsing and named entity recognition. However, both these works require a unified annotation of syntactic and semantic elements, which is not always feasible, and focused only on named entities and binary relations. On the other hand, our approach focuses on event structures that are nested and have an arbitrary number of arguments. We do not need a unified syntactic and semantic representation (but we can and do extract features from the underlying syntactic structure of the text).

Finkel and Manning (2009b) also proposed a parsing model for the extraction of nested named entity mentions, which, like this work, parses just the corresponding semantic annotations. In this work, we focus on more complex structures (events instead of named entities) and we explore more global features through our reranking layer.

In the biomedical domain, two recent papers proposed joint models for event extraction based on Markov logic networks (MLN) (Riedel et al., 2009; Poon and Vanderwende, 2010). Both works propose elegant frameworks where event anchors and arguments are jointly predicted for all events in the same sentence. One disadvantage of MLN models is the requirement that a human expert develop domain-specific predicates and formulas, which can be a cumbersome process because it requires thorough domain understanding. On the other hand, our approach maintains the joint modeling advantage, but our model is built over simple, domain-independent features. We also propose and analyze a richer feature space that captures more information on the global event structure in a sentence. Furthermore, since our approach is agnostic to the parsing model used, it could easily be tuned for various scenarios, e.g., models with lower inference overhead such as shift-reduce parsers.

Our work is conceptually close to the recent CoNLL shared tasks on semantic role labeling, where the predicate frames were converted to se-

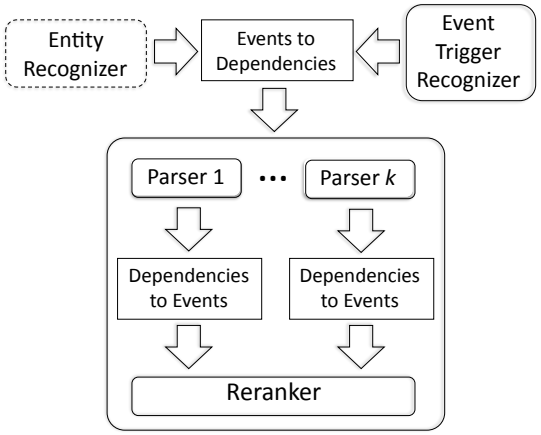


Figure 2: Overview of the approach. Rounded rectangles indicate domain-independent components; regular rectangles mark domain-specific modules; blocks in dashed lines surround components not necessary for the domain presented in this paper.

mantic dependencies between predicates and their arguments (Surdeanu et al., 2008; Hajic et al., 2009). In this representation the dependency structure is a directed acyclic graph (DAG), i.e., the same node can be an argument to multiple predicates, and there are no explicit dependencies between predicates. Due to this representation, all joint models proposed for semantic role labeling handle semantic frames independently.

3 Approach

Figure 2 summarizes our architecture. Our approach converts the original event representation to dependency trees containing both event anchors and entity mentions, and trains a battery of parsers to recognize these structures. The trees are built using event anchors predicted by a separate classifier. In this work, we do not discuss entity recognition because in the BioNLP'09 domain used for evaluation entities (PROTEINS) are given (but including entity recognition is an obvious extension of our model). Our parsers are several instances of MSTParser[2] (McDonald et al., 2005; McDonald et al., 2005b) configured with different decoders. However, our approach is agnostic to the actual parsing models used and could easily be adapted to other dependency parsers. The output from the reranking parser is converted back to the original event representation and passed to a reranker component (Collins, 2000; Charniak and Johnson, 2005), tailored to optimize the task-specific evaluation metric.

Note that although we use the biomedical event domain from the BioNLP'09 shared task to illustrate our work, the core of our approach is almost domain independent. Our only constraints are that each event mention be activated by a phrase that serves as an event anchor, and that the event-argument structures be mapped to a dependency tree. The conversion between event and dependency structures and the reranker metric are the only domain dependent components in our approach.

3.1 Converting between Event Structures and Dependencies

As in previous work, we extract event structures at sentence granularity, i.e., we ignore events which span sentences (Björne et al., 2009; Riedel et al., 2009; Poon and Vanderwende, 2010). These form approximately 5% of the events in the BioNLP'09 corpus. For each sentence, we convert the BioNLP'09 event representation to a graph (representing a labeled dependency tree) as follows. The nodes in the graph are protein entity mentions, event anchors, and a virtual ROOT node. Thus, the only words in this dependency tree are those which participate in events. We create edges in the graph in the following way. For each event anchor, we create one link to each of its arguments labeled with the slot name of the argument (for example, connecting **gene transcription** to *IL-2* with the label THEME in Figure 1b). We link the ROOT node to each entity that does not participate in an event using the ROOT-LABEL dependency label. Finally, we link the ROOT node to each top-level event anchor, (those which do not serve as arguments to other events) again using the ROOT-LABEL label. We follow the convention that the source of each dependency arc is the head while the target is the modifier.

The output of this process is a directed graph, since a phrase can easily play a role in two or more events. Furthermore, the graph may contain self-referential edges (self-loops) due to related events sharing the same anchor (example below). To guarantee that the output of this process is a tree, we must post-process the above graph with the follow-

[2] http://sourceforge.net/projects/mstparser/

ing three heuristics:

Step 1: We remove self-referential edges. An example of these can be seen in the text "the domain interacted preferentially with **underphosphorylated** *TRAF2*," there are two events anchored by the same **underphosphorylated** phrase, a NEGATIVE REGULATION and a PHOSPHORYLATION event, and the latter serves as a THEME argument for the former. Due to the shared anchor, our conversion component creates an self-referential THEME dependency. By removing these edges, 1.5% of the events in the training arguments are left without arguments, so we remove them as well.

Step 2: We break structures where one argument participates in multiple events, by keeping only the dependency to the event that appears first in text. For example, in the fragment "by enhancing its **inactivation** through **binding** to soluble *TNF-alpha receptor type II*," the protein *TNF-alpha receptor type II* is an argument in both a BINDING event (**binding**) and in a NEGATIVE REGULATION event (**inactivation**). As a consequence of this step, 4.7% of the events in training are removed.

Step 3: We unify events with the same types anchored on the same anchor phrase. For example, for the fragment "Surface **expression** of *intercellular adhesion molecule-1*, *P-selectin*, and *E-selectin*," the BioNLP'09 annotation contains three distinct GENE EXPRESSION events anchored on the same phrase (**expression**), each having one of the proteins as THEMEs. In such cases, we migrate all arguments to one of the events, and remove the empty events. 21.5% of the events in training are removed in this step (but no dependencies are lost).

Note that we do not guarantee that the resulting tree is projective. In fact, our trees are more likely to be non-projective than syntactic dependency trees of English sentences, because in our representation many nodes can be linked directly to the ROOT node. Our analysis indicates that 2.9% of the dependencies generated in the training corpus are non-projective and 7.9% of the sentences contain at least one non-projective dependency (for comparison, these numbers for the English Penn Treebank are 0.3% and 6.7%, respectively).

After parsing, we implement the inverse process, i.e., we convert the generated dependency trees to the BioNLP'09 representation. In addition to the obvious conversions, this process implements the heuristics proposed by Björne et al. (2009), which reverse step 3 above, e.g., we duplicate GENE EXPRESSION events with multiple THEME arguments. The heuristics are executed sequentially in the given order:

1. Since all non-BINDING events can have at most one THEME argument, we duplicate non-BINDING events with multiple THEME arguments by creating one separate event for each THEME.

2. Similarly, since REGULATION events accepts only one CAUSE argument, we duplicate REGULATION events with multiple CAUSE arguments, obtaining one event per CAUSE.

3. Lastly, we implement the heuristic of Björne et al. (2009) to handle the splitting of BINDING events with multiple THEME arguments. This is more complex because these events can accept one or more THEMEs. In such situations, we first group THEME arguments by the label of the first Stanford dependency (Marneffe and Manning, 2008) from the head word of the anchor to this argument. Then we create one event for each combination of THEME arguments in different groups.

3.2 Recognition of Event Anchors

For anchor detection, we used a multiclass classifier that labels each token independently.[3] Since over 92% of the anchor phrases in our evaluation domain contain a single word, we simplify the task by reducing all multi-word anchor phrases in the training corpus to their syntactic head word (e.g., "acts" for the anchor "**acts as a costimulatory signal**").

We implemented this model using a logistic regression classifier with L2 regularization over the following features:

[3] We experimented with using conditional random fields as a sequence labeler but did not see improvements in the biomedical domain. We hypothesize that the sequence tagger fails to capture potential dependencies between anchor labels – which are its main advantage over an i.i.d. classifier – because anchor words are typically far apart in text. This result is consistent with observations in previous work (Björne et al., 2009).

- **Token-level:** The form, lemma, and whether the token is present in a gazetteer of known anchor words.[4]
- **Surface context:** The above token features extracted from a context of two words around the current token. Additionally, we build token bigrams in this context window, and model them with similar features.
- **Syntactic context:** We model all syntactic dependency paths up to depth two starting from the token to be classified. These paths are built from Stanford syntactic dependencies (Marneffe and Manning, 2008). We extract token features from the first and last token in these paths. We also generate combination features by concatenating: (a) the last token in each path with the sequence of dependency labels along the corresponding path; and (b) the word to be classified, the last token in each path, and the sequence of dependency labels in that path.
- **Bag-of-word** and **entity count:** Extracted from (a) the entire sentence, and (b) a window of five words around the token to be classified.

3.3 Parsing Event Structures

Given the entities and event anchors from the previous stages in the pipeline, the parser generates labeled dependency links between them. Many dependency parsers are available and we chose MSTParser for its ability to produce non-projective and n-best parses directly. MSTParser frames parsing as a graph algorithm. To parse a sentence, MSTParser finds the tree covering all the words (nodes) in the sentence (graph) with the largest sum of edge weights, i.e., the maximum weighted spanning tree. Each labeled, directed edge in the graph represents a possible dependency between its two endpoints and has an associated score (weight). Scores for edges come from the dot product between the edge's corresponding feature vector and learned feature weights. As a result, all features for MSTParser must be *edge-factored*, i.e., functions of both endpoints and the label connecting them. McDonald et al. (2006) extends the basic model to include second-order dependencies (i.e., two adjacent sibling nodes and their parent). Both first and second-order modes include projective and non-projective decoders.

Our features for MSTParser use both the event structures themselves as well as the surrounding English sentences which include them. By mapping event anchors and entities back to the original text, we can incorporate information from the original English sentence as well its syntactic tree and corresponding Stanford dependencies. Both forms of context are valuable and complementary. MSTParser comes with a large number of features which, in our setup, operate on the event structure level (since this is the "sentence" from the parser's point of view). The majority of additional features that we introduced take advantage of the original text as context (primarily its associated Stanford dependencies). Our system includes the following first-order features:

- **Path:** Syntactic paths in the original sentence between nodes in an event dependency (as in previous work by Björne et al. (2009)). These have many variations including using Stanford dependencies ("collapsed" and "uncollapsed") or constituency trees as sources, optionally lexicalizing the path, and using words or relation names along the path. Additionally, we include the bucketed length of the paths.
- **Original sentence words:** Words from the full English sentence surrounding and between the nodes in event dependencies, and their bucketed distances. This additional context helps compensate for how our anchor detection provides only the head word of each anchor, which does not necessarily provide the full context for event disambiguation.
- **Graph:** Parents, children, and siblings of nodes in the Stanford dependencies graph along with label of the edge. This provides additional syntactic context.
- **Consistency:** Soft constraints on edges between anchors and their arguments (e.g., only regulation events can have edges labeled with CAUSE). These features fire if their constraints are violated.
- **Ontology:** Generalized types of the endpoints of edges using a given type hierarchy (e.g., POSITIVE REGULATION is a COM-

[4] These are automatically extracted from the training corpus.

PLEX EVENT[5] is an EVENT). Values of this feature are coded with the types of each of the endpoints on an edge, running over the cross-product of types for each endpoint. For instance, an edge between a BINDING event anchor and a POSITIVE REGULATION could cause this feature to fire with the values [head:EVENT, child:COMPLEX EVENT] or [head:SIMPLE EVENT, child:EVENT].[6] The latter feature can capture generalizations such as "simple event anchors cannot take other events as arguments."

Both **Consistency** and **Ontology** feature classes include domain-specific information but can be used on other domains under different constraints and type hierarchies. When using second-order dependencies, we use additional **Path** and **Ontology** features. We include the syntactic paths between sibling nodes (adjacent arguments of the same event anchor). These **Path** features are as above but differentiated as paths between sibling nodes. The second-order **Ontology** features use the type hierarchy information on both sibling nodes and their parent. For example, a POSITIVE REGULATION anchor attached to a PROTEIN and a BINDING event would produce an **Ontology** feature with the value [parent:COMPLEX EVENT, child_1:PROTEIN, child_2:SIMPLE EVENT] (among several other possible combinations).

To prune the number of features used, we employ a simple entropy-based measure. Our intuition is that good features should typically appear with only one edge label.[7] Given all edges enumerated during training and their gold labels, we obtain a distribution over edge labels (d_f) for each feature f. Given this distribution and the frequency of a feature, we can score the feature with the following:

$$score(f) = \alpha \times \log_2 \big(freq(f)\big) - H(d_f)$$

The α parameter adjusts the relative weight of the two components. The log frequency component favors more frequent features while the entropy component favors features with low entropy in their edge label distribution. Features are pruned by accepting all features with a score above a certain threshold.

3.4 Reranking Event Structures

When decoding, the parser finds the highest scoring tree which incorporates global properties of the sentence. However, its features are edge-factored and thus unable to take into account larger contexts. To incorporate arbitrary global features, we employ a two-step reranking parser. For the first step, we extend our parser to output its n-best parses instead of just its top scoring parse. In the second step, a discriminative reranker rescores each parse and reorders the n-best list. Rerankers have been successfully used in syntactic parsing (Collins, 2000; Charniak and Johnson, 2005; Huang, 2008) and semantic role labeling (Toutanova et al., 2008).

Rerankers provide additional advantages in our case due to the mismatch between the dependency structures that the parser operates on and their corresponding event structures. We convert the output from the parser to event structures (Section 3.1) before including them in the reranker. This allows the reranker to capture features over the actual event structures rather than their original dependency trees which may contain extraneous portions.[8] Furthermore, this lets the reranker optimize the actual BioNLP F1 score. The parser, on the other hand, attempts to optimize the Labeled Attachment Score (LAS) between the dependency trees and converted gold dependency trees. LAS is approximate for two reasons. First, it is much more local than the BioNLP metric.[9] Second, the converted gold dependency trees lose information that doesn't transfer to trees (specifically, that event structures are really multi-DAGs and not trees).

We adapt the maximum entropy reranker from Charniak and Johnson (2005) by creating a customized feature extractor for event structures — in all other ways, the reranker model is unchanged. We use the following types of features in the reranker:

- **Source:** Score and rank of the parse from the

[5] We define complex events are those which can accept other events are arguments. Simple events can only take PROTEINs.

[6] We omit listing the other two combinations.

[7] Labels include ROOT-LABEL, THEME, CAUSE, and NULL. We assign the NULL label to edges which aren't in the gold data.

[8] For instance, event anchors with no arguments could be proposed by the parser. These event anchors are automatically dropped by the conversion process.

[9] As an example, getting an edge label between an anchor and its argument correct is unimportant if the anchor is missing other arguments.

Decoder(s)	Unreranked			Reranked		
	R	P	F1	R	P	F1
1P	65.6	76.7	70.7	68.0	77.6	72.5
2P	67.4	77.1	71.9	67.9	77.3	72.3
1N	67.5	76.7	71.8	—	—	—
2N	68.9	77.1	72.7	—	—	—
1P, 2P, 2N	—	—	—	68.5	78.2	**73.1**

(a) Gold event anchors

Decoder(s)	Unreranked			Reranked		
	R	P	F1	R	P	F1
1P	44.7	62.2	52.0	47.8	59.6	53.1
2P	45.9	61.8	52.7	48.4	57.5	52.5
1N	46.0	61.2	52.5	—	—	—
2N	38.6	66.6	48.8	—	—	—
1P, 2P, 2N	—	—	—	48.7	59.3	**53.5**

(b) Predicted event anchors

Table 1: BioNLP recall, precision, and F1 scores of individual decoders and the best decoder combination on development data with the impact of event anchor detection and reranking. Decoder names include the features order (1 or 2) followed by the projectivity (P = projective, N = non-projective).

decoder; number of different decoders producing the parse (when using multiple decoders).

- **Event path:** Path from each node in the event tree up to the root. Unlike the **Path** features in the parser, these paths are over event structures, not the syntactic dependency graphs from the original English sentence. Variations of the **Event path** features include whether to include word forms (e.g., "binds"), types (BINDING), and/or argument slot names (THEME). We also include the path length as a feature.
- **Event frames:** Event anchors with all their arguments and argument slot names.
- **Consistency:** Similar to the parser **Consistency** features, but capable of capturing larger classes of errors (e.g., incorrect number or types of arguments). We include the number of violations from four different classes of errors.

To improve performance and robustness, features are pruned as in Charniak and Johnson (2005): selected features must distinguish a parse with the highest F1 score in a n-best list, from a parse with a suboptimal F1 score at least five times.

Rerankers can also be used to perform model combination (Toutanova et al., 2008; Zhang et al., 2009; Johnson and Ural, 2010). While we use a single parsing model, it has multiple decoders.[10] When combining multiple decoders, we concatenate their n-best lists and extract the unique parses.

4 Experimental Results

Our experiments use the BioNLP'09 shared task corpus (Kim et al., 2009) which includes 800 biomedical abstracts (7,449 sentences, 8,597 events) for training and 150 abstracts (1,450 sentences, 1,809 events) for development. The test set includes 260 abstracts, 2,447 sentences, and 3,182 events. Throughout our experiments, we report BioNLP F1 scores with approximate span and recursive event matching (as described in the shared task definition). For preprocessing, we parsed all documents using the self-trained biomedical McClosky-Charniak-Johnson reranking parser (McClosky, 2010). We bias the anchor detector to favor recall, allowing the parser and reranker to determine which event anchors will ultimately be used. When performing n-best parsing, $n = 50$. For parser feature pruning, $\alpha = 0.001$.

Table 1a shows the performance of each of the decoders when using gold event anchors. In both cases where n-best decoding is available, the reranker improves performance over the 1-best parsers. We also present the results from a reranker trained from multiple decoders which is our highest scoring model.[11] In Table 1b, we present the output for the predicted anchor scenario. In the case of the 2P decoder, the reranker does not improve performance, though the drop is minimal. This is because the reranker chose an unfortunate regularization constant during crossvalidation, most likely due to the small size of the training data. In later experiments where more

[10] We only have n-best versions of the projective decoders. For the non-projective decoders, we use their 1-best parse.

[11] Including the 1N decoder as well provided no gains, possibly because its outputs are mostly subsumed by the 2N decoder.

data is available, the reranker consistently improves accuracy (McClosky et al., 2011). As before, the reranker trained from multiple decoders outperforms unreranked models and reranked single decoders. All in all, our best model in Table 1a scores 1 F1 point higher than the best system at the BioNLP'09 shared task, and the best model in Table 1b performs similarly to the best shared task system (Björne et al., 2009), which also scores 53.5% on development.

We show the effects of each system component in Table 2. Note how our upper limit is 87.1% due to our conversion process, which enforces the tree constraint, drops events spanning sentences, and performs approximate reconstruction of BINDING events. Given that state-of-the-art systems on this task currently perform in the 50-60% range, we are not troubled by this number as it still allows for plenty of potential.[12] Björne et al. (2009) list 94.7% as the upper limit for their system. Considering this relatively large difference, we find the results in the previous table very encouraging. As in other BioNLP'09 systems, our performance drops when switching from gold to predicted anchor information. Our decrease is similar to the one seen in Björne et al. (2009).

To show the potential of reranking, we provide oracle reranker scores in Table 3. An oracle reranker picks the highest scoring parse from the available parses. We limit the n-best lists to the top k parses where $k \in \{1, 2, 10, \text{All}\}$. For single decoders, "All" uses the entire 50-best list. For multiple decoders, the n-best lists are concatenated together. The oracle score with multiple decoders and gold anchors is only 0.4% lower than our upper limit (see Table 2). This indicates that parses which could have achieved that limit were nearly always present. Improving the features in the reranker as well as the original parsers will help us move closer to the limit. With predicated anchors, the oracle score is about 13% lower but still shows significant potential.

Our final results on the test set, broken down by class, are shown in Table 4. As with other systems, complex events (e.g., REGULATION) prove harder than simple events. To get a complex event correct, one must correctly detect and parse all events in the event subtree allowing small errors to have large effects. Top systems on this task obtain F1 scores of 52.0% at the shared task evaluation (Björne et al., 2009) and 56.3% post evaluation (Miwa et al., 2010a). However, both systems are tailored to the biomedical domain (the latter uses multiple syntactic parsers), whereas our system has a design that is virtually domain independent.

[12] Additionally, improvements such as document-level parsing and DAG parsing would eliminate the need for much of the approximate and lossy portions of the conversion process.

AD	Parse	RR	Conv	R	P	F1
✓	✓		✓	45.9	61.8	52.7
✓	✓	✓	✓	48.7	59.3	53.5
G	✓		✓	68.9	77.1	72.7
G	✓	✓	✓	68.5	78.2	73.1
G	G	G	✓	81.6	93.4	87.1

Table 2: Effect of each major component to the overall performance in the development corpus. Components shown: AD — event anchor detection; Parse — best individual parsing model; RR — reranking multiple parsers; Conv — conversion between the event and dependency representations. 'G' indicates that gold data was used; '✓' indicates that the actual component was used.

		n-best parses considered			
Anchors	Decoder(s)	1	2	10	All
Gold	1P	70.7	76.6	84.0	85.7
	2P	71.8	77.5	84.8	86.2
	1P, 2P, 2N	—	—	—	86.7
Predicted	1P	52.0	60.3	69.9	72.5
	2P	52.7	60.7	70.1	72.5
	1P, 2P, 2N	—	—	—	73.4

Table 3: Oracle reranker BioNLP F1 scores for our n-best decoders and their combinations before reranking on the development corpus.

5 Discussion

We believe that the potential of our approach is higher than what the current experiments show. For example, the reranker can be used to combine not only several parsers but also multiple anchor recognizers. This passes the anchor selection decision to the reranker, which uses global information not available to the current anchor recognizer or parser. Furthermore, our approach can be adapted to parse event structures in entire documents (instead of in-

Event Class	Count	R	P	F1
Gene Expression	722	68.6	75.8	72.0
Transcription	137	42.3	51.3	46.4
Protein Catabolism	14	64.3	75.0	69.2
Phosphorylation	135	80.0	82.4	81.2
Localization	174	44.8	78.8	57.1
Binding	347	42.9	51.7	46.9
Regulation	291	23.0	36.6	28.3
Positive Regulation	983	28.4	42.5	34.0
Negative Regulation	379	29.3	43.5	35.0
Total	3,182	42.6	56.6	48.6

Table 4: Results in the test set broken by event class; scores generated with the main official metric of approximate span and recursive event matching.

dividual sentences) by using a representation with a unique ROOT node for all event structures in a document. This representation has the advantage that it maintains cross-sentence events (which account for 5% of BioNLP'09 events), and it allows for document-level features that model discourse structure. We plan to explore these ideas in future work.

One current limitation of the proposed model is that it constrains event structures to map to trees. In the BioNLP'09 corpus this leads to the removal of almost 5% of the events, which generate DAGs instead of trees. Local event extraction models (Björne et al., 2009) do not have this limitation, because their local decisions are blind to (and hence not limited by) the global event structure. However, our approach is agnostic to the actual parsing models used, so we can easily incorporate models that can parse DAGs (Sagae and Tsujii, 2008). Additionally, we are free to incorporate any new techniques from dependency parsing. Parsing using dual-decomposition (Rush et al., 2010) seems especially promising in this area.

6 Conclusion

In this paper we proposed a simple approach for the joint extraction of event structures: we converted the representation of events and their arguments to dependency trees with arcs between event anchors and event arguments, and used a reranking parser to parse these structures. Despite the fact that our approach has very little domain-specific engineering, we obtain competitive results. Most importantly, we showed that the joint modeling of event structures is beneficial: our reranker outperforms parsing models without reranking in five out of the six configurations investigated.

Acknowledgments

The authors would like to thank Mark Johnson for helpful discussions on the reranker component and the BioNLP shared task organizers, Sampo Pyysalo and Jin-Dong Kim, for answering questions. We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of DARPA, AFRL, or the US government.

References

Jari Björne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. 2009. *Extracting Complex Biological Events with Rich Graph-Based Feature Sets*. Proceedings of the Workshop on BioNLP: Shared Task.

Nate Chambers and Dan Jurafsky. 2009. *Unsupervised Learning of Narrative Schemas and their Participants*. Proceedings of ACL.

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

Nancy Chinchor. 1997. *Overview of MUC-7*. Proceedings of the Message Understanding Conference (MUC-7).

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML 2000)*, pages 175–182.

Jenny R. Finkel and Christopher D. Manning. 2009. *Joint Parsing and Named Entity Recognition*. Proceedings of NAACL.

Jenny R. Finkel and Christopher D. Manning. 2009b. *Nested Named Entity Recognition*. Proceedings of EMNLP.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria A. Marti, Lluis Marquez, Adam Meyers, Joakim Nivre, Sebastian Pado, Jan Stepanek, Pavel Stranak, Mihai Surdeanu, Nianwen

Xue, and Yi Zhang. 2009. *The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages*. Proceedings of CoNLL.

Eduard Hovy, Mitchell P. Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. *OntoNotes: The 90% Solution*. Proceedings of the NAACL-HLT.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Association for Computational Linguistics.

Mark Johnson and Ahmet Engin Ural. 2010. *Reranking the Berkeley and Brown Parsers*. Proceedings of NAACL.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. *Overview of the BioNLP'09 Shared Task on Event Extraction*. Proceedings of the NAACL-HLT 2009 Workshop on Natural Language Processing in Biomedicine (BioNLP'09).

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

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. *Stanford Typed Hierarchies Representation*. Proceedings of the COLING Workshop on Cross-Framework and Cross-Domain Parser Evaluation.

David McClosky. 2010. *Any Domain Parsing: Automatic Domain Adaptation for Natural Language Parsing*. PhD thesis, Department of Computer Science, Brown University.

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

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. *Online Large-Margin Training of Dependency Parsers*. Proceedings of ACL.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. *Non-projective Dependency Parsing using Spanning Tree Algorithms*. Proceedings of HLT/EMNLP.

Ryan McDonald and Fernando Pereira. 2006. *Online Learning of Approximate Dependency Parsing Algorithms*. Proceedings of EACL.

Scott Miller, Michael Crystal, Heidi Fox, Lance Ramshaw, Richard Schwartz, Rebecca Stone, and Ralph Weischedel. 1997. *BBN: Description of the SIFT System as Used for MUC-7*. Proceedings of the Message Understanding Conference (MUC-7).

Makoto Miwa, Sampo Pyysalo, Tadayoshi Hara, and Jun'ichi Tsujii. 2010a. *A Comparative Study of Syntactic Parsers for Event Extraction*. Proceedings of the 2010 Workshop on Biomedical Natural Language Processing.

Makoto Miwa, Rune Saetre, Jin-Dong Kim, and Jun'ichi Tsujii. 2010b. *Event Extraction with Complex Event Classification Using Rich Features*. Journal of Bioinformatics and Computational Biology, 8 (1).

Hoifung Poon and Lucy Vanderwende. 2010. *Joint Inference for Knowledge Extraction from Biomedical Literature*. Proceedings of NAACL.

Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. *A Markov Logic Approach to Bio-Molecular Event Extraction*. Proceedings of the Workshop on BioNLP: Shared Task.

Alexander M. Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. *Proceedings of EMNLP*.

Kenji Sagae and Jun'ichi Tsujii. 2008. *Shift-Reduce Dependency DAG Parsing*. Proceedings of the COLING.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluis Marquez, and Joakim Nivre. 2008. *The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies*. Proceedings of CoNLL.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. *A Global Joint Model for Semantic Role Labeling*. Computational Linguistics 34(2).

Zhang, H. and Zhang, M. and Tan, C.L. and Li, H. 2009. *K-best combination of syntactic parsers*. Proceedings of Empirical Methods in Natural Language Processing.

Extracting Comparative Entities and Predicates from Texts Using Comparative Type Classification

Seon Yang
Department of Computer Engineering,
Dong-A University,
Busan, Korea
seony.yang@gmail.com

Youngjoong Ko
Department of Computer Engineering,
Dong-A University,
Busan, Korea
yjko@dau.ac.kr

Abstract

The automatic extraction of comparative information is an important text mining problem and an area of increasing interest. In this paper, we study how to build a Korean comparison mining system. Our work is composed of two consecutive tasks: 1) classifying comparative sentences into different types and 2) mining comparative entities and predicates. We perform various experiments to find relevant features and learning techniques. As a result, we achieve outstanding performance enough for practical use.

1 Introduction

Almost every day, people are faced with a situation that they must decide upon one thing or the other. To make better decisions, they probably attempt to compare entities that they are interesting in. These days, many web search engines are helping people look for their interesting entities. It is clear that getting information from a large amount of web data retrieved by the search engines is a much better and easier way than the traditional survey methods. However, it is also clear that directly reading each document is not a perfect solution. If people only have access to a small amount of data, they may get a biased point of view. On the other hand, investigating large amounts of data is a time-consuming job. Therefore, a *comparison mining* system, which can automatically provide a summary of comparisons between two (or more) entities from a large quantity of web documents, would be very useful in many areas such as marketing.

We divide our work into two tasks to effectively build a comparison mining system. The first task is related to a sentence classification problem and the second is related to an information extraction problem.

Task 1. Classifying comparative sentences into one non-comparative class and seven comparative classes (or types); *1) Equality, 2) Similarity, 3) Difference, 4) Greater or lesser, 5) Superlative, 6) Pseudo,* and *7) Implicit* comparisons. The purpose of this task is to efficiently perform the following task.

Task 2. Mining comparative entities and predicates taking into account the characteristics of each type. For example, from the sentence "*Stock-X is worth more than stock-Y.*" belonging to "*4) Greater or lesser*" type, we extract "*stock-X*" as a subject entity (SE), "*stock-Y*" as an object entity (OE), and "*worth*" as a comparative predicate (PR).

These tasks are not easy or simple problems as described below.

Classifying comparative sentences (Task 1): For the first task, we extract comparative sentences from text documents and then classify the extracted comparative sentences into seven

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1636–1644,
Portland, Oregon, June 19-24, 2011.

comparative types. Our basic idea is a keyword search. Since Ha (1999a) categorized dozens of Korean comparative keywords, we easily build an initial keyword set as follows:

▪ K_{ling} = {"같 ([gat]: same)", "보다 ([bo-da]: than)", "가장 ([ga-jang]: most)", …}

In addition, we easily match each of these keywords to a particular type anchored to Ha's research, e.g., "같 ([gat]: same)" to "*1) Equality*", "보다 ([bo-da]: than)" to "*4) Greater or lesser*". However, any method that depends on just these linguistic-based keywords has obvious limitations as follows:

1) K_{ling} is insufficient to cover all of the actual comparison expressions.
2) There are many non-comparative sentences that contain some elements of K_{ling}.
3) There is no one-to-one relationship between keyword types and sentence types.

Mining comparative entities and predicates (Task 2): Our basic idea for the second task is selecting candidates first and finding answers from the candidates later. We regard each of noun words as a candidate for SE/OE, and each of adjective (or verb) words as a candidate for PR. However, this candidate detection has serious problems as follows:

4) There are many actual SEs, OEs, and PRs that consist of multiple words.
5) There are many sentences with no OE, especially among superlative sentences. It means that the ellipsis is frequently occurred in superlative sentences.

We focus on solving the above five problems. We perform various experiments to find relevant features and proper machine learning techniques. The final experimental results in 5-fold cross validation show the overall accuracy of 88.59% for the first task and the overall accuracy of 86.81% for the second task.

The remainder of the paper is organized as follows. Section 2 briefly introduces related work. Section 3 and Section 4 describe our first task and second task in detail, respectively. Section 5 reports our experimental results and finally Section 6 concludes.

2 Related Work

Linguistic researchers focus on defining the syntax and semantics of comparative constructs. Ha (1999a; 1999b) classified the structures of Korean comparative sentences into several classes and arranged comparison-bearing words from a linguistic perspective. Since he summarized the modern Korean comparative studies, his research helps us have a linguistic point of view. We also refer to Jeong (2000) and Oh (2004). Jeong classified adjective superlatives using certain measures, and Oh discussed the gradability of comparatives.

In computer engineering, we found five previous studies related to comparison mining. Jindal and Liu (2006a; 2006b) studied to mine comparative relations from English text documents. They used comparative and superlative POS tags, and some additional keywords. Their methods applied Class Sequential Rules and Label Sequential Rules. Yang and Ko (2009; 2011) studied to extract comparative sentences in Korean text documents. Li et al. (2010) studied to mine comparable entities from English comparative questions that users posted online. They focused on finding a set of comparable entities given a user's input entity.

Opinion mining is also related to our work because many comparative sentences also contain the speaker's opinion/sentiment. Lee et al. (2008) surveyed various techniques that have been developed for the key tasks of opinion mining. Kim and Hovy (2006) introduced a methodology for analyzing judgment opinion. Riloff and Wiebe (2003) presented a bootstrapping process that learns linguistically rich extraction patterns for subjective expressions.

In this study, three learning techniques are employed: the maximum entropy method (MEM) as a representative probabilistic model, the support vector machine (SVM) as a kernel model, and transformation-based learning (TBL) as a rule-based model. Berger et al. (1996) presented a Maximum Entropy Approach to natural language processing. Joachims (1998) introduced SVM for text classification. Various TBL studies have been performed. Brill (1992; 1995) first introduced TBL and presented a case study on part-of-speech

tagging. Ramshaw and Marcus (1995) applied TBL for locating chunks in tagged texts. Black and Vasilakopoulos (2002) used a modified TBL technique for Named Entity Recognition.

3 Classifying Comparative Sentences (Task 1)

We first classify the sentences into *comparatives* and *non-comparatives* by extracting only comparatives from text documents. Then we classify the comparatives into seven types.

3.1 Extracting comparative sentences from text documents

Our strategy is to first detect *Comparative Sentence candidates* (CS-candidates), and then eliminate non-comparative sentences from the candidates. As mentioned in the introduction section, we easily construct a linguistic-based keyword set, K_{ling}. However, we observe that K_{ling} is not enough to capture all the actual comparison expressions. Hence, we build a comparison lexicon as follows:

- *Comparison Lexicon* = K_{ling} U {Additional keywords that are frequently used for actual comparative expressions}

This lexicon is composed of three parts. The first part includes the elements of K_{ling} and their synonyms. The second part consists of idioms. For example, an idiom "X가 먼저 웃었다 [X-ga meon-jeo u-seot-da]" commonly means "The winner is X" while it literally means "X laughed first". The last part consists of long-distance-words sequences, e.g., "<X 는 [X-neun], 지만 [ji-man], Y 는 [Y-neun], 다 [da]>". This sequence means that the sentence is formed as < S(X) + V + but + S(Y) + V > in English (S: subject phrase; V: verb phrase; X, Y: proper nouns). We could regard a word, "지만 ([ji-man]: but)," as a single keyword. However, this word also captures numerous non-comparative sentences. Namely, the precision value can fall too much due to this word. By using long-distance-words sequences instead of single keywords, we can keep the precision value from dropping seriously low.

The comparison lexicon finally has a total of 177 elements. We call each element "CK" hereafter. Note that our lexicon does not include comparative/superlative POS tags. Unlike English, there is no Korean comparative/superlative POS tag from POS tagger commonly. Our lexicon covers 95.96% of the comparative sentences in our corpus. It means that we successfully defined a comparison lexicon for CS-candidate detection. However, the lexicon shows a relatively low precision of 68.39%. While detecting CS-candidates, the lexicon also captures many non-comparative sentences, e.g., following Ex1:

- Ex1. "내일은 주식이 오를 것 같다." ([nai-il-eun ju-sik-i o-reul-geot gat-da]: I think stock price will rise tomorrow.)

This sentence is a non-comparative sentence even though it contains a CK, "같[gat]." This CK generally means "*same*," but it often expresses "*conjecture*." Since it is an adjective in both cases, it is difficult to distinguish the difference.

To effectively filter out non-comparative sentences from CS-candidates, we use the sequences of "continuous POS tags within a radius of 3 words from each CK" as features. Each word in the sequence is replaced with its POS tag in order to reflect various expressions. However, as CKs play the most important role, they are represented as a combination of their lexicalization and POS tag, e.g., "같/pa[1]." Finally, the feature has the form of "X → y" ("X" means a sequence and "y" means a class; y_1: comparative, y_2: non-comparative). For instance, "<pv etm nbn 같/pa ef sf[2] >→ y_2" is one of the features from Ex1 sentence. Finally, we achieved an f1-score of 90.23% using SVM.

3.2 Classifying comparative sentences into seven types

As we extract comparative sentences successfully, the next step is to classify the comparatives into different types. We define seven comparative types and then employ TBL for comparative sentence classification.

We first define six broad comparative types based on modern Korean linguistics: *1) Equality, 2) Similarity, 3) Difference, 4) Greater or lesser, 5) Superlative, 6) Pseudo* comparisons. The first five types can be understood intuitively, whereas

[1] The POS tag "*pa*" means "the stem of an adjective".
[2] The labels such as "*pv*", "*etm*" are Korean POS Tags.

the sixth type needs more explanation. "*6) Pseudo*" comparison includes comparative sentences that compare two (or more) properties of one entity such as "Smartphone-X is a computer rather than a phone." This type of sentence is often classified into "*4) Greater or lesser*." However, since this paper focuses on comparisons between different entities, we separate "*6) Pseudo*" type from "*4) Greater or lesser*" type.

The seventh type is "*7) Implicit*" comparison. It is added with the goal of covering literally "implicit" comparisons. For example, the sentence "Shopping Mall X guarantees no fee full refund, but Shopping Mall Y requires refund-fee" does not directly compare two shopping malls. It implicitly gives a hint that X is more beneficial to use than Y. It can be considered as a non-comparative sentence from a linguistic point of view. However, we conclude that this kind of sentence is as important as the other explicit comparisons from an engineering point of view.

After defining the seven comparative types, we simply match each sentences to a particular type based on the CK types; e.g., a sentence which contains the word "가장 ([ga-jang]: most)" is matched to "*Superlative*" type. However, a method that uses just the CK information has a serious problem. For example, although we easily match the CK "보다 ([bo-da]: than)" to "*Greater or lesser*" without doubt, we observe that the type of CK itself does not guarantee the correct type of the sentence as we can see in the following three sentences:

- Ex2. "X 의 품질은 Y **보다** 좋지도 나쁘지도 않다." ([X-eui pum-jil-eun Y-bo-da jo-chi-do na-ppeu-ji-do an-ta]: The quality of X is neither better nor worse than that of Y.) → It can be interpreted as "The quality of X is **similar** to that of Y." (*Similarity*)
- Ex3. "X 가 Y **보다** 품질이 좋다." ([X-ga Y-bo-da pum-jil-I jo-ta]: The quality of X is better than that of Y.) → It is consistent with the CK type (*Greater or lesser*)
- Ex4. "X 는 다른 어떤 카메라**보다** 품질이 좋다." ([X-neun da-reun eo-tteon ka-me-ra-bo-da pum-jil-i jo-ta]: X is better than any other cameras in quality.) → It can be interpreted as "X is the **best** camera in quality." (*Superlative*)

If we only rely on the CK type, we should label the above three sentences as "*Greater or lesser*". However, each of these three sentences belongs to a different type. This fact addresses that many CKs could have an ambiguity problem just like the CK of "보다 ([bo-da]: than)."

To solve this ambiguity problem, we employ TBL. We first roughly annotate the type of sentences using the type of CK itself. After this initial annotating, TBL generates a set of error-driven transformation rules, and then a scoring function ranks the rules. We define our scoring function as Equation (1):

$$\boldsymbol{Score(r_i) = C_i - E_i} \quad (1)$$

Here, r_i is the *i*-th transformation rule, $\boldsymbol{C_i}$ is the number of corrected sentences after r_i is applied, and $\boldsymbol{E_i}$ is the number of the opposite case. The ranking process is executed iteratively. The iterations stop when the scoring function reaches a certain threshold. We finally set up the threshold value as 1 after tuning. This means that we use only the rules whose score is 2 or more.

4 Mining Comparative Entities and Predicates (Task 2)

This section explains how to extract comparative entities and predicates. Our strategy is to first detect *Comparative Element candidates* (CE-candidates), and then choose the answer among the candidates.

In this paper, we only present the results of two types: "*Greater or lesser*" and "*Superlative*." As we will see in the experiment section, these two types cover 65.8% of whole comparative sentences. We are still studying the other five types and plan to report their results soon.

4.1 Comparative elements

We extract three kinds of comparative elements in this paper: SE, OE and PR

- Ex5. "X 파이가 Y 파이보다 싸고 맛있다." ([X-pa-i-ga Y-pa-i-bo-da ssa-go mas-it-da]: Pie X is cheaper and more delicious than Pie Y.)
- Ex6. "대선 후보들 중 Z 가 가장 믿음직하다." ([dai-seon hu-bo-deul jung Z-ga ga-jang mit-eum-jik-ha-da]: "Z is the most trustworthy among the presidential candidates.")

In Ex5 sentence, "X 파이 (Pie X)" is a SE, "Y 파이 (Pie Y)" is an OE, and "싸고 맛있다 (cheaper and more delicious)" is a PR. In Ex6 sentence, "Z" is a SE, "대선 후보들 (the presidential candidates)" is an OE, and "믿음직하다 (trustworthy)" is a PR.

Note that comparative elements are not limited to just one word. For example, "싸고 맛있다 (cheaper and more delicious)" and "대선 후보들 (the presidential candidates)" are composed of multiple words. After investigating numerous actual comparison expressions, we conclude that SEs, OEs, and PRs should not be limited to a single word. It can miss a considerable amount of important information to restrict comparative elements to only one word. Hence, we define as follows:

- *Comparative elements (SE, OE, and PR) are composed of one or more consecutive words.*

It should also be noted that a number of superlative sentences are expressed without OE. In our corpus, the percentage of the *Superlative* sentences without any OE is close to 70%. Hence, we define as follows:

- *OEs can be omitted in the Superlative sentences.*

4.2 Detecting CE-candidates

As comparative elements are allowed to have multiple words, we need some preprocessing steps for easy detection of CE-candidates. We thus apply some simplification processes. Through the simplification processes, we represent potential SEs/OEs as one "N" and potential PRs as one "P". The following process is one of the simplification processes for making "N"

- Change each noun (or each noun compound) to a symbol "N".

And, the following two example processes are for "P".

- Change "pa (adjective)" and "pv (verb)" to a symbol "P".
- Change "P + ecc (a suffix whose meaning is "and") + P" to one "P", e.g., "cheaper and more delicious" is tagged as one "P".

In addition to the above examples, several processes are performed. We regard all the "N"s as CE-candidates for SE/OE and all the "P"s as CE-candidates for PR. It is possible that a more analytic method is used instead of this simplification task, e.g., by a syntactic parser. We leave this to our future work.

4.3 Finding final answers

We now generate features. The patterns that consist of POS tags, CKs, and "P"/"N" sequences within a radius of 4 POS tags from each "N" or "P" are considered as features.

Original sentence	"X 파이가 Y 파이보다 싸고 맛있다." (Pie X is cheaper and more delicious than Pie Y.)
After POS tagging	X 파이/nq + 가/jcs + Y 파이/nq + 보다/jca + 싸/pa + 고/ecc + 맛있/pa + 다/ef +./sf
After simplification process	X 파이/N**(SE)** + 가/jcs + Y 파이/N**(OE)** + 보다/jca + 싸고맛있다/P**(PR)** + ./sf
Patterns for SE	<**N(SE)**, jcs, N, 보다/jca,P>, …, <**N(SE)**, jcs>
Patterns for OE	<N, jcs, **N(OE)**, 보다/jca,P, sf>, …, <**N(OE)**, 보다/jca >
Patterns for PR	<N, jcs, N, 보다/jca,**P(PR)**, sf>, …, <**P(PR)**, sf>

Table 1: Feature examples for mining comparative elements

Table 1 lists some examples. Since the CKs play an important role, they are represented as a combination of their lexicalization and POS tag. After feature generation, we calculate each probability value of all CE-candidates using SVM. For example, if a sentence has three "P"s, one "P" with the highest probability value is selected as the answer PR.

5 Experimental Evaluation

5.1 Experimental Settings

The experiments are conducted on 7,384 sentences collected from the web by three trained human labelers. Firstly, two labelers annotated the corpus. A Kappa value of 0.85 showed that it was safe to say that the two labelers agreed in their judgments.

Secondly, the third labeler annotated the conflicting part of the corpus. All three labelers discussed any conflict, and finally reached an agreement. Table 2 lists the distribution of the corpus.

Comparative Types		Sentence Portion
Non-comparative:		5,001 (67.7%)
Comparative:		2,383 (32.3%)
Total (Corpus)		7,384 (100%)
Among Comparative Sentences	*1) Equality*	3.6%
	2) Similarity	7.2%
	3) Difference	4.8%
	4) Greater or lesser	54.5%
	5) Superlative	11.3%
	6) Pseudo	1.3%
	7) Implicit	17.5%
	Total (Comparative)	100%

Table 2: Distribution of the corpus

5.2 Classifying comparative sentences

Our experimental results for Task 1 showed an f1-score of 90.23% in extracting comparative sentences from text documents and an accuracy of 81.67% in classifying the comparative sentences into seven comparative types.

The integrated results showed an accuracy of 88.59%. Non-comparative sentences were regarded as an eighth comparative type in this integrated result. It means that we classify entire sentences into eight types (seven comparative types and one non-comparative type).

5.2.1 Extracting comparative sentences.

Before evaluating our proposed method for comparative sentence extraction, we conducted four experiments with all of the lexical unigrams and bigrams using MEM and SVM. Among these four cases, SVM with lexical unigrams showed the highest performance, an f1-score of 79.49%. We regard this score as our baseline performance.

Next, we did experiments using all of the continuous lexical sequences and using all of the POS tags sequences within a radius of *n* words from each CK as features (*n*=1,2,3,4,5). Among these ten cases, "the POS tags sequences within a radius of 3" showed the best performance. Besides, as SVM showed the better performance than MEM in overall experiments, we employ SVM as our proposed learning technique. Table 3 summarizes the overall results.

Systems	Precision	Recall	F1-score
baseline	87.86	72.57	79.49
comparison lexicon only	68.39	95.96	79.87
comparison lexicon & SVM (proposed)	92.24	88.31	***90.23***

Table 3: Final results in comparative sentence extraction (%)

As given above, we successfully detected CS-candidates with considerably high recall by using the comparison lexicon. We also successfully filtered the candidates with high precision while still preserving high recall by applying machine learning technique. Finally, we could achieve an outstanding performance, an f1-score of 90.23%.

5.2.2 Classifying comparative sentences into seven types.

Like the previous comparative sentence extraction task, we also conducted experiments for type classification using the same features (continuous POS tags sequences within a radius of 3 words from each CK) and the same learning technique (SVM). Here, we achieved an accuracy of 73.64%. We regard this score as our baseline performance.

Next, we tested a completely different technique, the TBL method. TBL is well-known to be relatively strong in sparse problems. We observed that the performance of type classification can be influenced by very subtle differences in many cases. Hence, we think that an error-driven approach can perform well in comparative type classification. Experimental results showed that TBL actually performed better than SVM or MEM.

In the first step, we roughly annotated the type of a sentence using the type of the CK itself. Then, we generated error-driven transformation rules from the incorrectly annotated sentences. Transformation templates we defined are given in Table 4. Numerous transformation rules were generated on the basis of the templates. For example, "Change the type of the current sentence from "*Greater or lesser*" to "*Superlative*" if this sentence holds the CK of "보다 ([bo-da]: than)",

and the second preceding word of the CK is tagged as *mm*" is a transformation rule generated by the third template.

Change the type of the current sentence from *x* to *y* if this sentence holds the CK of ***k***, and …
1. the preceding word of ***k*** is tagged ***z***.
2. the following word of ***k*** is tagged ***z***.
3. the second preceding word of ***k*** is tagged ***z***.
4. the second following word of ***k*** is tagged ***z***.
5. the preceding word of ***k*** is tagged ***z,*** and the following word of ***k*** is tagged ***w***.
6. the preceding word of ***k*** is tagged ***z,*** and the second preceding word of ***k*** is tagged ***w***.
7. the following word of ***k*** is tagged ***z,*** and the second following word of ***k*** is tagged ***w***.

Table 4: Transformation templates

For evaluation of threshold values, we performed experiments with three options as given in Table 5.

Threshold	0	1	2
Accuracy	79.99	***81.67***	80.04

Table 5: Evaluation of threshold option (%);
Threshold *n* means that the learning iterations continues while C_i-$E_i \geq n$+1

We achieved the best performance with the threshold option 1. Finally, we classified comparative sentences into seven types using TBL with an accuracy of 81.67%.

5.2.3 Integrated results of Task 1

We sum up our proposed method for Task 1 as two steps as follows;

1) The comparison lexicon detects CS-candidates in text documents, and then SVM eliminates the non-comparative sentences from the candidates. Thus, all of the sentences are divided into two classes: a comparative class and a non-comparative class.
2) TBL then classifies the sentences placed in the comparative class in the previous step into seven comparative types.

The integrated results showed an overall accuracy of 88.59% for the eight-type classification. To evaluate the effectiveness of our two-step processing, we performed one-step processing experiments using SVM and TBL. Table 6 shows a comparison of the results.

Processing		**Accuracy**
One-step processing (classifying eight types at a time)	comparison lexicon & SVM	75.64
	comparison lexicon & TBL	72.49
Two-step processing (proposed)		***88.59***

Table 6: Integrated results for Task 1 (%)

As shown above, Task 1 was successfully divided into two steps.

5.3 Mining comparative entities and predicates

For the mining task of comparative entities and predicates, we used 460 comparative sentences (*Greater or lesser*: 300, *Superlative*: 160). As previously mentioned, we allowed multiple-word comparative elements. Table 7 lists the portion of multiple-word comparative elements.

Multi-word rate	**SE**	**OE**	**PR**
Greater or lesser	30.0	31.3	8.3
Superlative	24.4	9.4 (32.6)	8.1

Table 7: Portion (%) of multiple-word comparative elements

As given above, each multiple-word portion, especially in SEs and OEs, is quite high. This fact proves that it is absolutely necessary to allow multiple-word comparative elements. Relatively lower rate of 9.4% in *Superlative*-OEs is caused by a number of omitted OEs. If sentences that do not have any OEs are excluded, the portion of multiple-words becomes 32.6% as written in parentheses.

Table 8 shows the effectiveness of simplification processes. We calculated the error rates of CE-candidate detection before and after simplification processes.

Simplification processes		SE	OE	PR
Greater or lesser	Before	34.7	39.3	10.0
	After	***4.7***	***8.0***	***1.7***
Superlative	Before	26.3	85.0 (38.9)	9.4
	After	***1.9***	***75.6 (6.3)***	***1.3***

Table 8: Error rate (%) in CE-candidate detection

Here, the first value of 34.7% means that the real SEs of 104 sentences (among total 300 *Greater or lesser* sentences) were not detected by CE-candidate detection before simplification processes. After the processes, the error rate decreased to 4.7%. The significant differences between before and after indicate that we successfully detect CE-candidates through the simplification processes. Although the *Superlative*-OEs still show the seriously high rate of 75.6%, it is also caused by a number of omitted OEs. If sentences that do not have any OEs are excluded, the error rate is only 6.3% as written in parentheses.

The final results for Task 2 are reported in Table 9. We calculated each probability of CE-candidates using MEM and SVM. Both MEM and SVM showed outstanding performance; there was no significant difference between the two machine learning methods (SVM and MEM). Hence, we only report the results of SVM. Note that many sentences do not contain any OE. To identify such sentences, if SVM tagged every "N" in a sentence as "not OE", we tagged the sentence as "no OE".

Final Results	SE	OE	PR
Greater or lesser	86.00	89.67	92.67
Superlative	84.38	71.25	90.00
Total	**85.43**	**83.26**	**91.74**

Table 9: Final results of Task 2 (Accuracy, %)

As shown above, we successfully extracted the comparative entities and predicates with outstanding performance, an overall accuracy of 86.81%.

6 Conclusions and Future Work

This paper has studied a Korean comparison mining system. Our proposed system achieved an accuracy of 88.59% for classifying comparative sentences into eight types (one non-comparative type and seven comparative types), and an accuracy of 86.81% for mining comparative entities and predicates. These results demonstrated that our proposed method could be used effectively in practical applications. Since the comparison mining is an area of increasing interest around the world, our study can contribute greatly to text mining research.

In our future work, we have the following plans. Our first plan is to complete the mining process on all the types of sentences. The second one is to conduct more experiments for obtaining better performance. The final one is about an integrated system. Since we perform Task 1 and Task 2 separately, we need to build an end-to-end system.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0015613)

References

Adam L. Berger, Stephen A. Della Pietra and Vicent J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39-71.

William J. Black and Argyrios Vasilakopoulos. 2002. Language-Independent named Entity Classification by modified Transformation-based Learning and by Decision Tree Induction. In *Proceedings of CoNLL'02*, 24:1-4.

Eric Brill. 1992. A simple rule-based part of speech tagger. In *Proceedings of ANLP'92*, 152-155.

Eric Brill. 1995. Transformation-based Error-Driven Learning and Natural language Processing: A Case Study in Part-of-Speech tagging. *Computational Linguistics*, 543-565.

Gil-jong Ha. 1999a. *Korean Modern Comparative Syntax*, Pijbook Press, Seoul, Korea.

Gil-jong Ha. 1999b. Research on Korean Equality Comparative Syntax, *Association for Korean Linguistics*, 5:229-265.

In-su Jeong. 2000. Research on Korean Adjective Superlative Comparative Syntax. *Korean Han-min-jok Eo-mun-hak*, 36:61-86.

Nitin Jindal and Bing Liu. 2006. Identifying Comparative Sentences in Text Documents, In *Proceedings of SIGIR'06*, 244-251.

Nitin Jindal and Bing Liu. 2006. Mining Comparative Sentences and Relations, In *Proceedings of AAAI'06*, 1331-1336.

Thorsten Joachims. 1998. Text Categorization with Support Vector Machines: Learning with Many relevant Features. In *Proceedings of ECML'98*, 137-142

Soomin Kim and Eduard Hovy. 2006. Automatic Detection of Opinion Bearing Words and Sentences. In *Proceedings of ACL'06*.

Dong-joo Lee, OK-Ran Jeong and Sang-goo Lee. 2008. Opinion Mining of Customer Feedback Data on the Web. In *Proceedings of ICUIMC'08*, 247-252.

Shasha Li, Chin-Yew Lin, Young-In Song and Zhoujun Li. 2010. Comparable Entity Mining from Comparative Questions. In *Proceedings of ACL'10*, 650-658.

Kyeong-sook Oh. 2004. The Difference between 'Man-kum' Comparative and 'Cheo-rum' Comparative. *Society of Korean Semantics*, 14:197-221.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text Chunking using Transformation-Based Learning. In *Proceedings of NLP/VLC'95*, 82-94.

Ellen Riloff and Janyce Wiebe. 2003. Learning Extraction Patterns for Subjective Expressions. In *Proceedings of EMNLP'03*.

Seon Yang and Youngjoong Ko. 2009. Extracting Comparative Sentences from Korean Text Documents Using Comparative Lexical Patterns and Machine Learning Techniques. In *Proceedings of ACL-IJNLP:Short Papers*, 153-156

Seon Yang and Youngjoong Ko. 2011. Finding relevant features for Korean comparative sentence extraction. *Pattern Recognition Letters*, 32(2):293-296

Author Index