ACL-IJCNLP 2009

Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and 4th International Joint Conference on Natural Language Processing of the AFNLP

Proceedings of the Conference

Volume 1

2-7 August 2009
Suntec, Singapore

Production and Manufacturing by
World Scientific Publishing Co Pte Ltd
5 Toh Tuck Link
Singapore 596224

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-932432-45-9 / 1-932432-45-0

Preface: General Chair

Welcome to the ACL-IJCNLP 2009, the first joint conference sponsored by the ACL (Association for Computational Linguistics) and the AFNLP (Asian Federation of Natural Language Processing). The idea to have a joint conference for ACL and AFNLP was first discussed at ACL-05 (Ann Arbor, Michigan) among Martha Palmer (ACL President), Benjamin T'sou (AFNLP President), Jun'ichi Tsujii (AFNLP Vice President) and Keh-Yih Su (AFNLP Conference Coordinating Committee Chair, also the Secretary General). We are glad that the original idea has come true four years later, and even the affiliation relationship between these two organizations has been built up now.

In this joint conference, we have tried to mix the spirit from both ACL and AFNLP; and, Singapore, which itself has a mixture of diversified cultures from eastern and western regions, is certainly a wonderful place to see how different languages meet each other. We hope you will enjoy this big event held in this garden city, which is brought to you via the efforts from each member of the conference organization team.

Among our hard working organizers, I would like to thank the Program Chairs, Jan Wiebe and Jian Su, who has carefully selected papers from our record high submissions, and the Local Arrangements Chair, Haizhou Li, who has shown his excellent capability in smoothly organizing various events and details. My thanks will also go to other chairs for their competent and hard work: The Webmaster, Minghui Dong; the Demo Chairs, Gary Geunbae Lee and Sabine Schulte im Walde; the Exhibits Chairs, Timothy Baldwin and Philipp Koehn; the Mentoring Service Chairs, Hwee Tou Ng and Florence Reeder; the Publication Chairs, Jing-Shin Chang and Regina Barzilay; the Publicity Chairs, Min-Yen Kan and Andy Way; the Sponsorship Chairs, Hitoshi Isahara and Kim-Teng Lua; the Student Research Workshop Chairs, Davis Dimalen, Jenny Rose Finkel, and Blaise Thomson; also the Faculty Advisors, Grace Ngai and Brian Roark; the Tutorial Chairs, Diana McCarthy and Chengqing Zong; the Workshop Chairs, Jimmy Lin and Yuji Matsumoto; last, the ACL Business Manager, Priscilla Rasmussen, who not only provides useful advice but also helps to contact more sponsors and get their support.

Besides, I need to express my gratitude to the Conference Coordination Committee for their valuable advice and support: in which Bonnie Dorr (chair), Steven Bird, Graeme Hirst, Kathleen McCoy, Martha Palmer, Dragomir Radev, Priscilla Rasmussen, Mark Steedman are from ACL; and Yuji Matsumoto, Keh-Yih Su, Jun'ichi Tsujii, Benjamin T'sou, Kam-Fai Wong are from AFNLP.

Last, I sincerely thank all the authors, reviewers, presenters, invited speakers, sponsors, exhibitors, local supporting staff, and all the conference attendants. It is you that make this conference possible. Wish you all enjoy the program that we provide.

Keh-Yih Su
ACL-IJCNLP 2009 General Chair
August 2009

Preface: Program Committee Co-Chairs

For the first time, the flagship conferences of the Association for Computational Linguistics (ACL) and the Asian Federation of Natural Language Processing (AFNLP) – the ACL and IJCNLP – are jointly organized as a single event. ACL-IJCNLP 2009 covers a broad spectrum of technical areas related to natural language and computation, representing a rich array of the state of the art. The conference includes full papers, short papers, demonstrations, a student research workshop, as well as pre- and post-conference tutorials and workshops.

This year, we again received a record number of submissions: 925 total valid paper submissions, a 24% increase over ACL-08: HLT. This includes 569 full-paper submissions and 356 short-paper submissions from more than 40 countries – approximately 51% from 20 countries in Asia Pacific, 27% from Canada, Cuba and the United States, 22% from 15 countries in Europe, fewer than 1% from Argentina, and one paper submitted anonymously. We thank all of the authors for submitting papers describing their recent work. The significant submission increase is a trend extending over multiple years, and shows how vigorous our field is. We also thank Hwee Tou Ng and Florence Reeder, the Mentoring Service Co-Chairs, for organizing a 19-mentor team who provided English scientific paper writing support.

20 Area Chairs worked with 489 Program Committee members and 85 additional reviewers to come up with 2551 reviews, in total, for the final paper selection. 21% of the full-paper submissions were accepted; all will be presented orally. 26% of the short-paper submissions were accepted; some will be presented orally and some as poster presentations. While short papers are distinguished from full papers in the proceedings, there are no distinctions in the proceedings between short papers presented orally and those presented as posters. We are absolutely indebted to the Area Chairs, Program Committee members, and additional reviewers for their intensive efforts.

We are delighted to have two keynote speakers: Qiang Yang, who will talk about heterogeneous transfer learning, and Bonnie Webber, who will address discourse and genre. Best (student) paper awards and the ACL Lifetime Achievement Award will be announced in the last session of the conference as well.

We thank General Conference Chair Keh Yih Su, the Local Arrangements Committee headed by Haizhou Li, and the ACL-AFNLP Conference Coordination Committee chaired by Bonnie Dorr, for their help and advice, as well as last years PC Co-Chairs, Johanna Moore and Simone Teufel, for sharing their experiences, Jason Eisner for his How to Serve as Program Chair of a Conference website and corresponding emails, Jing-Shin Chang and Regina Barzilay, the Publication Co-Chairs for putting the proceedings together, and all the other committee chairs for their work. Our thanks go to our assistant Chen Bin, who worked tirelessly throughout the entire process, and who made our work with START much easier. Together, everyone made such a wonderful event possible.

We hope that you enjoy the conference!

Jian Su, Institute for Infocomm Research
Jan Wiebe, University of Pittsburgh

Organizing Committee

General Conference Chair:

Su, Keh-Yih (Behavior Design Corp., Taiwan; kysu@bdc.com.tw)

Program Chairs:

Su, Jian (Institute for Infocomm Research, Singapore; sujian@i2r.a-star.edu.sg)
Wiebe, Janyce (University of Pittsburgh, USA; janycewiebe@gmail.com)

Local Organizing Chair:

Li, Haizhou (Institute for Infocomm Research, Singapore; hli@i2r.a-star.edu.sg)

Demo Chairs:

Lee, Gary Geunbae (POSTECH, Korea; gblee@postech.ac.kr)
Schulte im Walde, Sabine (University of Stuttgart, Germany; schulte@ims.uni-stuttgart.de)

Exhibits Chairs:

Baldwin, Timothy (University of Melbourne, Australia; tim@csse.unimelb.edu.au)
Koehn, Philipp (University of Edinburgh, UK; pkoehn@inf.ed.ac.uk)

Mentoring Service Chairs:

Ng, Hwee Tou (National University of Singapore, Singapore; nght@comp.nus.edu.sg)
Reeder, Florence (Mitre, USA; freeder@mitre.org)

Publication Chairs:

Barzilay, Regina (MIT, USA; regina@csail.mit.edu)
Chang, Jing-Shin (National Chi Nan University, Taiwan; jshin@csie.ncnu.edu.tw)

Publicity Chairs:

Kan, Min-Yen (National University of Singapore, Singapore; kanmy@comp.nus.edu.sg)
Way, Andy (Dublin City University, Ireland; away@computing.dcu.ie)

Sponsorship Chairs:

Americas Sponsorship Co-Chairs:
Bangalore, Srinivas
Doran, Christine

European Sponsorship Co-Chairs:
Koehn, Philipp
Genabith, Josef van

Asia Sponsorship Co-Chairs:
Isahara, Hitoshi (NICT, Japan; isahara@nict.go.jp)
Lua, Kim-Teng (COLIPS, Singapore; luakt@colips.org)

Student Chairs:

Dimalen, Davis (Academia Sinica, Taiwan; d_dimalen@yahoo.com)
Finkel, Jenny Rose (Stanford University, USA; jrfinkel@cs.stanford.edu)
Thomson, Blaise (Cambridge University, UK; brmt2@cam.ac.uk)

Student Workshop Faculty Advisors:

Ngai, Grace (Polytechnic University, Hong Kong; csgngai@polyu.edu.hk)
Roark, Brian (Oregon Health & Science University, USA; roark@cslu.ogi.edu)

Tutorial Chairs:

McCarthy, Diana (University of Sussex, UK; dianam@sussex.ac.uk)
Zong, Chengqing (Chinese Academy of Sciences, China; cqzong@gmail.com)

Workshop Chairs:

Lin, Jimmy (University of Maryland, USA; jimmylin@umd.edu)
Matsumoto, Yuji (NAIST, Japan; matsu@is.naist.jp)

Webmaster:

Dong, Minghui (Institute for Infocomm Research, Singapore; mhdong@i2r.a-star.edu.sg)

Registration:

Rasmussen, Priscilla(ACL; rasmusse@ptd.net)

Program Committee

Program Chairs:

Su, Jian (Institute for Infocomm Research, Singapore; sujian@i2r.a-star.edu.sg)
Wiebe, Janyce (University of Pittsburgh, USA; janycewiebe@gmail.com)

Area Chairs:

Agirre, Eneko (University of Basque Country, Spain; e.agirre@ehu.es)
Ananiadou, Sophia (University of Manchester, UK; sophia.ananiadou@manchester.ac.uk)
Belz, Anja (University of Brighton, UK; a.s.belz@itri.brighton.ac.uk)
Carenini, Giuseppe (University of British Columbia, Canada; carenini@cs.ubc.ca)
Chen, Hsin-Hsi (National Taiwan University, Taiwan, hh_chen@csie.ntu.edu.tw)
Chen, Keh-Jiann (Sinica, Taiwan, kchen@iis.sinica.edu.tw)
Curran, James (University of Sydney, Australia; james@it.usyd.edu.au)
Gao, Jian Feng (MSR, USA; jfgao@microsoft.com)
Harabagiu, Sanda (University of Texas at Dallas, USA, sanda@hlt.utdallas.edu)
Koehn, Philipp (University of Edinburgh, UK; pkoehn@inf.ed.ac.uk)
Kondrak, Grzegorz (University of Alberta, Canada; kondrak@cs.ualberta.ca)
Meng, Helen Mei-Ling (Chinese University of Hong Kong, HK; hmmeng@se.cuhk.edu.hk)
Mihalcea, Rada (University of North Texas, USA; rada@cs.unt.edu)
Poesio, Massimo(University of Trento, Italy; poesio@disi.unitn.it)
Riloff, Ellen (University of Utah, USA; riloff@cs.utah.edu)
Sekine, Satoshi (NewYork University, USA; sekine@cs.nyu.edu)
Smith, Noah (CMU, USA; nasmith@cs.cmu.edu)
Strube, Michael (EML Research, Germany; strube@eml-research.de)
Suzuki, Jun (NTT, Japan; jun@cslab.kecl.ntt.co.jp)
Wang, Hai Feng (Toshiba, China; wanghaifeng@rdc.toshiba.com.cn)

Program Committee Members:

Eugene Agichtein, Gregory Aist, Salah Ait-Mokhtar, Enrique Alfonseca, Yaser Al-Onaizan, Sophia Ananiadou, Alina Andreevskaia, Ion Androutsopoulos, Doug Appelt, Xabier Arregi, Abhishek Arun, Masayuki Asahara, Jordi Atserias, Michaela Atterer

Jason Baldridge, Srinivas Bangalore, Michele Banko, Marco Baroni, Regina Barzilay, Roberto Basili, Sabine Bergler, Shane Bergsma, Steven Bethard, Pushpak Bhattacharyya, Mikhail Bilenko, Philippe Blache, Alan Black, Sasha Blair-Goldensohn, John Blitzer, Phil Blunsom, Philip Blunsom, Bernd Bohnet, Johan Bos, Pierre Boullier, Thorsten Brants, Eric Breck, Chris Brew, Ted Briscoe, Chris Brockett, Paul Buitelaar, Razvan Bunescu, Harry Bunt, Stephan Busemann, Donna Byron

Aoife Cahill, Lynne Cahill Cahill, Nicoletta Calzolari, Nick Campbell, Claire Cardie, Giuseppe Carenini, Michael Carl, Xavier Carreras, John Carroll, Francisco Casacuberta, Jon Chamberlain, Ciprian Chelba, John Chen, Keh-Jiann Chen, Pu-Jen Cheng, Colin Cherry, David Chiang, Key-Sun Choi, Yejin Choi, Monojit Choudhury, Tat-Seng Chua, Jennifer Chu-Carroll, Philip Cimiano, Stephen Clark, James Clarke, Kevin Cohen, Shay Cohen, Trevor Cohn, Nigel Collier, John Conroy, Mark Craven, Dan Cristea, Andras Csomai, Silviu-PetrCucerzan, Hang Cui, Aron Culotta, James Curran

Robert Dale, Hal Daume, Eric de la Clergerie, Maarten de Rijke, Vera Demberg, Dina Demner, Dina Demner-Fushman, John DeNero, Li Deng, Yonggang Deng, Pascal Denis, Ann Devitt, Mona Diab, Arantza Diaz, Bill Dolan, John Dowding, Mark Dras, Mark Dredze, Jasha Droppo, Amit Dubey, Kevin Duh, Kenneth Dwyer, Chris Dyer

Phil Edmonds, Andreas Eisele, Jacob Eisenstein, Jason Eisner, Michael Elhadad, Mark Ellison, Micha Elsner, Katrin Erk, Andrea Esuli, Marc Ettlinger, Stefan Evert

Ronen Feldman, Christiane Fellbaum, Raquel Fernandez, Elena Filatova, Katja Filippova, Jenny Finkel, Dan Flickinger, Radu Florian, Juliane Fluck, Eric Fosler-Lussier, George Foster, Alex Fraser, Marjorie Freedman, Carol Friedman, Qiang Fu

Evgeniy Gabrilovich, Rob Gaizauskas, Michael Gamon, Albert Gatt, Ulrich Germann, Daniel Gildea, Jesus Gimenez, Jonathan GInzburg, Roxana Girju, Amir Globerson, Andrew Goldberg, John Goldsmith, Sharon Goldwater, Julio Gonzalo, Ralph Grishman, Rodrigo Guido, Tunga Gungor, Iryna Gurevych

Stephanie Haas, Aria Haghighi, Dilek Hakkani-Tur, Keith Hall, John Hansen, Sanda Harabagiu, Donna Harman, Saša Hasan, Samer Hassan, Timothy Hazen, Xiaodong He, Jeff Heinz, James Henderson, John Henderson, Andrew Hickl, Erhard Hinrichs, Keikichi Hirose, Julia Hirschberg, Graeme Hirst, Hieu Hoang, Julia Hockenmaier, Beth Ann Hockey, Mark Hopkins, Veronique Hoste, Churen Huang, Liang Huang, Sarmad Hussain, Rebecca Hwa, Mei-Yuh Hwang

Nancy Ide, Ryu Iida, Diana Inkpen, Kentaro Inui, Hitoshi Isahara, Abe Ittycheriah, Tatsuya Izuha

Heng Ji, Sittichai Jiampojamarn, Jing Jiang, Mark Johnson, Doug Jones, Gareth Jones

Mijail Kabadjov, Laura Kallmeyer, Nanda Kambhatla, Hiroshi Kanayama, Nikiforos Karamanis, Tsuneaki Kato, Hisashi Kawai, Junichi Kazama, Frank Keller, Andre Kempe, Brett Kessler, Mitesh Khapra, Bernd Kiefer, Adam Kilgarriff, Brian Kingsbury, James Kirby, Ewan Klein, Alexandre Klementiev, Kevin Knight, Rob Koeling, Terry Koo, Moshe Koppel, Wessel Kraaij, Emiel Krahmer, Ivana Kruijff-Korbayova, Lun-Wei Ku, Sandra Kuebler, Marco Kuhlmann, Jonas Kuhn, Seth Kulick, Akira Kumano, Shankar Kumar, A. Kumaran, June-Jei Kuo, Sadao Kurohashi, Kui-Lam Kwok

Philippe Langlais, Mirella Lapata, Alberto Lavelli, Gary Lee, Lillian Lee, Yoong Keok Lee, Yue-Shi Lee, Haizhou Li, Hang Li, Xiaolong Li, Percy Liang, Elizabeth Liddy, Dekang Lin, Lucian Lita, Ken Litkowski, Diane Litman, Bing Liu, Qun Liu, Tie-Yan Liu, Yang Liu, Zhanyi Liu, Adam Lopez, Xiaoqiang Luo, Yajuan Lv

Yanjun Ma, Lluís Màrquez, Karin Müller, Bernardo Magnini, Brian Mak, Rob Malouf, Gideon Mann, Daniel Marcu, Katja Markert, David Martínez, Andre Martins, Mstislav Maslennikov, Tomoko Matsui, Yuji Matsumoto, Takuya Matsuzaki, Evgeny Matusov, Arne Mauser, Diana McCarthy, David McClosky, Mark McConnville, Ryan McDonald, Michael McTear, Qiaozhu Mei, Chris Mellish, Arul Menezes, Paola Merlo, Detmar Meurers, David Mimno, Mandar Mitra, Vibhu Mittal, Yusuke Miyao, Daichi Mochihashi, Saif Mohammad, Rajat Mohanty, Diego Molla-Aliod, Christian Monson, Simonetta Montemagni, Bob Moore, Alex Morgan, Glyn Morrill, Alessandro Moschitti, Dragos Munteanu, Gabriel Murray, Sung Hyon Myaeng

Vivi Nastase, Tetsuya Nasukawa, Roberto Navigli, Mark-Jan Nederhof, Ani Nenkova, John Nerbonne, Hwee Tou Ng, Vincent Ng, Patrick Nguyen, Jian-Yun Nie, Zaiqing Nie, Takashi Ninomiya, Joakim Nivre, Chikashi Nobata, David Novick, Adrian Novischi

Franz Och, Stephan Oepen, Kemal Oflazer, Alice Oh, Jong-Hoon Oh, Daisuke Okanohara, Naoaki Okazaki, Fredrik Olsson, Constantin Orasan, Miles Osborne, Jahna Otterbacher

Tim Paek, Martha Palmer, Bo Pang, Patrick Pantel, Cecile Paris, Rebecca Passonneau, Michael Paul, Matthias Paulik, Anselmo Peñas, Adam Pease, Ted Pedersen, Catherine Pelachaud, Slav Petrov, Christopher Pinchak, Maja Popovic, Sameer Pradhan, John Prager, Rashmi Prasad, Detlef Prescher, Stephen Pulman, Sampo Pyysalo

Long Qiu, Silvia Quarteroni, Chris Quirk

Bhuvana Ramabhadran, Ganesh Ramakrishnan, Lance Ramshaw, Giuseppe Riccardi, Verena Rieser, German Rigau, Hae-Chang Rim, Brian Roark, James Rogers, Barbara Rosario, Carolyn Rose, Antti-Veikk Rosti, Patrick Ruch, Marta Ruiz, Andrey Rzhetsky

Kenji Sagae, Horacio Saggion, Benoit Sagot, Tetsuya Sakai, Baskaran Sankaran, Murat Saraclar, Ruhi Sarikaya, Yutaka Sasaki, Giorgio Satta, Anne Schiller, Helmut Schmid, Marc Schroeder, Holger Schwenk, Yohei Seki, Satoshi Sekine, Mike Seltzer, Jungyun Seo, Vijay Shanker, Hagit Shatkay, Libin Shen, Nobuyuki Shimizu, Luo Si, Advaith Siddharthan, Candy Sidner, Khalil Simaan, Michel Simard, Gabriel Skantze, David Smith, Noah Smith, Rion Snow, Ian Soboroff, Swapna Somasundaran, Radu Soricut, Caroline Sporleder, Rohini Srihari, Mark Steedman, Josef Steinberger, Amanda Stent, Mark Stevenson, Veselin Stoyanov, Carlo Strapparava, Michael Strube, Tomek Strzalkowski, Jana Sukkarieh, Maoson Sun, Mihai Surdeanu, Richard Sutcliffe, Stan Szpakowicz, Idan Szpektor

Maite Taboada, John Tait, Hiroya Takamura, David Talbot, Ben Taskar, Joel Tetreault, Simone Teufel, Joerg Tiedemann, Christoph Tillmann, Ivan Titov, Roberto Togneri, Keiichi Tokuda, Kristina Toutanova, Roy Tromble, Yuen-Hsien Tseng, Jun'ichi Tsujii, Yoshimasa Tsuruoka, Dan Tufis, Gokhan Tur

Antal van den Bosch, Josef van Genabith, Keith Vander Linden, Sebastian Varges, Tony Veale, Ashish Venugopal, Paola Verlardi, Yannick Versley, David Vilar, Piek Vossen

Michael Walsh, Stephen Wan, Xiaojun Wan, Qin Wang, Shaojun Wang, Wei Wang, Xinglong Wang, Taro Watanabe, Andy Way, Bonnie Webber, David Weir, Fuiliang Weng, Janyce Wiebe, Theresa Wilson, Shuly Wintner, Yuk Wah Wong, Johan Wouters, Dekai Wu, Hua Wu

Zhuli Xie, Deyi Xiong, Jun Xu, Peng Xu, Jian Xue

Kazuhide Yamamoto, Christopher Yang, Jianwu Yang, Muyun Yang, Kaisheng Yao, Umit Yapanel, Scott Wen-tau Yih, Deniz Yuret

Fabio Zanzotto, Dmitry Zelenko, Heiga Zen, Richard Zens, Luke Zettlemoyer, Hao Zhang, Min Zhang, Rong Zhang, Tong Zhang, Yue Zhang, Tiejun Zhao, Bowen Zhou, Denny Zhou, Ming Zhou, Jerry Zhu, Andreas Zollmann, Chengqing Zong, Pierre Zweigenbaum

Additional Reviewers:

Shlomo Argamon, Javier Artiles, Giuseppe Attardi, S.R.K. Branavan, Julian Brooke, David Burkett, Yong Cao, Yufeng Chen, Yu Chen, Ying Chen, Bonaventura Coppola, Sajib Dasgupta, Anirban Dasgupta, Diego De Cao, Oier Lopez de Lacalle, Adi Eyal, Jung-wei Fan, Benoit Favre, Moshe Fresko, Oana Frunza, Bin Gao, Xi-Wu Han, Zhongjun He, Carmen Heger, Zhongjun He, Wenbin Jiang, Richard Johansson, Anna Kazantseva, Alistair Kennedy, Fazel Keshtkar, Gerhard Kremer, Patrik Lambert, Greg Langmead, Oliver Lemon, Gregor Leusch, Xiao Li, Shoushan Li, Wei Li, Sujian Li, Xiaojiang Liu, Ting Liu, Yang Liu, Yue Lu, Jia Lu, Weihua Luo, Gang Luo, Yong-Liang Ma, Saab Mansour, Haitao Mi, Peter Nabenda, Peter Nabende, Ramesh Nallapati, Dipasree Pal, Adam Pauls, Emanuele Pianta, Daniele Pighin, Guilin Qi, Wei Qiao, Tao Qin, Jason Riesa, Felipe Sanchez-Martinez, Steven Schockaert, Dou Shen, Kathrin Spreyer, Jun Sun, Milan Tofiloski, Lav Varshney, Ye-Yi

Wang, Yu-Chieh Wu, Gu Xu, Sibel Yaman, Shiren Ye, Huang Yun, Hui Zhang, Yi Zhang, Bing Zhao, Yu Zhou, Zhemin Zhu, Conghui Zhu

Mentoring Service Committee

Chairs:

Ng, Hwee Tou (National University of Singapore, Singapore; nght@comp.nus.edu.sg)
Reeder, Florence (Mitre, USA; freeder@mitre.org)

Members:

Razvan Bunescu, Yee Seng Chan, Chrys Chrystello, Ken Church, Walter Daelemans, Deborah Dahl, Robert Daland, Janet Hitzeman, Eduard Hovy, Marilyn Kupetz, Preslav Nakov, Jian-Yun Nie, Kemal Oflazer, Bea Oshika, Dan Roth, Kenneth Samuel, Antal van den Bosch, John White, Yuk Wah Wong

Invited Talk:

Heterogeneous Transfer Learning with Real-world Applications

Qiang Yang
Hong Kong University of Science and Technology
qyang@cse.ust.hk

Abstract

In many real-world machine learning and data mining applications, we often face the problem where the training data are scarce in the feature space of interest, but much data are available in other feature spaces. Many existing learning techniques cannot make use of these auxiliary data, because these algorithms are based on the assumption that the training and test data must come from the same distribution and feature spaces. When this assumption does not hold, we have to seek novel techniques for 'transferring' the knowledge from one feature space to another. In this talk, I will present our recent works on heterogeneous transfer learning. I will describe how to identify the common parts of different feature spaces and learn a bridge between them to improve the learning performance in target task domains. I will also present several interesting applications of heterogeneous transfer learning, such as image clustering and classification, cross-domain classification and collaborative filtering.

Biography

Qiang Yang is a professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research interests are artificial intelligence, including automated planning, machine learning and data mining. He graduated from Peking University in 1982 with BSc. in Astrophysics, and obtained his MSc. degrees in Astrophysics and Computer Science from the University of Maryland, College Park in 1985 and 1987, respectively. He obtained his PhD in Computer Science from the University of Maryland, College Park in 1989. He was an assistant/associate professor at the University of Waterloo between 1989 and 1995, and a professor and NSERC Industrial Research Chair at Simon Fraser University in Canada from 1995 to 2001.

Qiang Yang has been active in research on artificial intelligence planning, machine learning and data mining. His research teams won the 2004 and 2005 ACM KDDCUP international competitions on data mining. He has been on several editorial boards of international journals, including IEEE Intelligent Systems, IEEE Transactions on Knowledge and Data Engineering and Web Intelligence. He has been an organizer for several international conferences in AI and data mining, including being the conference co-chair for ACM IUI 2010 and ICCBR 2001, program co-chair for PRICAI 2006 and PAKDD 2007, workshop chair for ACM KDD 2007, AAAI tutorial chair for AAAI 2005 and 2006, data mining contest chair for IEEE ICDM 2007 and 2009, and vice chair for ICDM 2006 and CIKM 2009. He is a fellow of IEEE and a member of AAAI and ACM. His home page is at http://www.cse.ust.hk/~qyang

Invited Talk:

Discourse - Early Problems, Current Successes, Future Challenges

Bonnie Webber
University of Edinburgh, UK
bonnie.webber@ed.ac.uk

Abstract

I will look back through nearly forty years of computational research on discourse, noting some problems (such as context-dependence and inference) that were identified early on as a hindrance to further progress, some admirable successes that we have achieved so far in the development of algorithms and resources, and some challenges that we may want to (or that we may have to!) take up in the future, with particular attention to problems of data annotation and genre dependence.

Biography

Bonnie Webber was a researcher at Bolt Beranek and Newman while working on the PhD she received from Harvard University in 1978. She then taught in the Department of Computer and Information Science at the University of Pennsylvania for 20 years before joining the School of Informatics at the University of Edinburgh. Known for research on discourse and on question answering, she is a Past President of the Association for Computational Linguistics, co-developer (with Aravind Joshi, Rashmi Prasad, Alan Lee and Eleni Miltsakaki) of the Penn Discourse TreeBank, and co-editor (with Annie Zaenen and Martha Palmer) of the journal, *Linguistic Issues in Language Technology*.

Table of Contents

Conference Program

Monday, August 3, 2009

08:30–08:40 Opening Session

08:40–09:40 Invited Talk: Qiang Yang, Heterogeneous Transfer Learning with Real-world Applications

Invited Talk

08:40–09:40 *Heterogeneous Transfer Learning for Image Clustering via the SocialWeb*
Qiang Yang, Yuqiang Chen, Gui-Rong Xue, Wenyuan Dai and Yong Yu

09:40–10:10 Break

Session 1A: Semantics 1
Chaired by Graeme Hirst

10:10–10:35 *Investigations on Word Senses and Word Usages*
Katrin Erk, Diana McCarthy and Nicholas Gaylord

10:35–11:00 *A Comparative Study on Generalization of Semantic Roles in FrameNet*
Yuichiroh Matsubayashi, Naoaki Okazaki and Jun'ichi Tsujii

11:00–11:25 *Unsupervised Argument Identification for Semantic Role Labeling*
Omri Abend, Roi Reichart and Ari Rappoport

11:25–11:50 *Brutus: A Semantic Role Labeling System Incorporating CCG, CFG, and Dependency Features*
Stephen Boxwell, Dennis Mehay and Chris Brew

Monday, August 3, 2009 (continued)

Session 1B: Syntax and Parsing 1
Chaired by Christopher Manning

10:10–10:35 *Exploiting Heterogeneous Treebanks for Parsing*
Zheng-Yu Niu, Haifeng Wang and Hua Wu

10:35–11:00 *Cross Language Dependency Parsing using a Bilingual Lexicon*
Hai Zhao, Yan Song, Chunyu Kit and Guodong Zhou

11:00–11:25 *Topological Field Parsing of German*
Jackie Chi Kit Cheung and Gerald Penn

11:25–11:50 *Unsupervised Multilingual Grammar Induction*
Benjamin Snyder, Tahira Naseem and Regina Barzilay

Session 1C:Statistical and Machine Learning Methods 1
Chaired by Jun Suzuki

10:10–10:35 *Reinforcement Learning for Mapping Instructions to Actions*
S.R.K. Branavan, Harr Chen, Luke Zettlemoyer and Regina Barzilay

10:35–11:00 *Learning Semantic Correspondences with Less Supervision*
Percy Liang, Michael Jordan and Dan Klein

11:00–11:25 *Bayesian Unsupervised Word Segmentation with Nested Pitman-Yor Language Modeling*
Daichi Mochihashi, Takeshi Yamada and Naonori Ueda

11:25–11:50 *Knowing the Unseen: Estimating Vocabulary Size over Unseen Samples*
Suma Bhat and Richard Sproat

Monday, August 3, 2009 (continued)

Session 1D: Phonology and Morphology
Chaired by Jason Eisner

10:10–10:35 *A Ranking Approach to Stress Prediction for Letter-to-Phoneme Conversion*
Qing Dou, Shane Bergsma, Sittichai Jiampojamarn and Grzegorz Kondrak

10:35–11:00 *Reducing the Annotation Effort for Letter-to-Phoneme Conversion*
Kenneth Dwyer and Grzegorz Kondrak

11:00–11:25 *Transliteration Alignment*
Vladimir Pervouchine, Haizhou Li and Bo Lin

11:25–11:50 *Automatic training of lemmatization rules that handle morphological changes in pre-, in- and suffixes alike*
Bart Jongejan and Hercules Dalianis

11:50–13:20 Lunch

Session 2A: Machine Translation 1
Chaired by Qun Liu

13:20–13:45 *Revisiting Pivot Language Approach for Machine Translation*
Hua Wu and Haifeng Wang

13:45–14:10 *Efficient Minimum Error Rate Training and Minimum Bayes-Risk Decoding for Translation Hypergraphs and Lattices*
Shankar Kumar, Wolfgang Macherey, Chris Dyer and Franz Och

14:10–14:35 *Forest-based Tree Sequence to String Translation Model*
Hui Zhang, Min Zhang, Haizhou Li, Aiti Aw and Chew Lim Tan

14:35–15:00 *Active Learning for Multilingual Statistical Machine Translation*
Gholamreza Haffari and Anoop Sarkar

Monday, August 3, 2009 (continued)

Session 2B: Generation and Summariation 1
Chaired by Anja Belz

13:20–13:45 *DEPEVAL(summ): Dependency-based Evaluation for Automatic Summaries*
Karolina Owczarzak

13:45–14:10 *Summarizing Definition from Wikipedia*
Shiren Ye, Tat-Seng Chua and Jie LU

14:10–14:35 *Automatically Generating Wikipedia Articles: A Structure-Aware Approach*
Christina Sauper and Regina Barzilay

14:35–15:00 *Learning to Tell Tales: A Data-driven Approach to Story Generation*
Neil McIntyre and Mirella Lapata

Session 2C: Sentiment Analysis and Text Categorization 1
Chaired by Katja Markert

13:20–13:45 *Recognizing Stances in Online Debates*
Swapna Somasundaran and Janyce Wiebe

13:45–14:10 *Co-Training for Cross-Lingual Sentiment Classification*
Xiaojun Wan

14:10–14:35 *A Non-negative Matrix Tri-factorization Approach to Sentiment Classification with Lexical Prior Knowledge*
Tao Li, Yi Zhang and Vikas Sindhwani

14:35–15:00 *Discovering the Discriminative Views: Measuring Term Weights for Sentiment Analysis*
Jungi Kim, Jin-Ji Li and Jong-Hyeok Lee

Monday, August 3, 2009 (continued)

Session 2D: Language Resources
Chaired by Nicoletta Calzolari

13:20–13:45 *Compiling a Massive, Multilingual Dictionary via Probabilistic Inference*
Mausam, Stephen Soderland, Oren Etzioni, Daniel Weld, Michael Skinner and Jeff Bilmes

13:45–14:10 *A Metric-based Framework for Automatic Taxonomy Induction*
Hui Yang and Jamie Callan

14:10–14:35 *Learning with Annotation Noise*
Eyal Beigman and Beata Beigman Klebanov

14:35–15:00 *Abstraction and Generalisation in Semantic Role Labels: PropBank, VerbNet or both?*
Paola Merlo and Lonneke van der Plas

15:00–15:30 Break

Session 3A: Machine Translation 2
Chaired by Haifeng Wang

15:30–15:55 *Robust Machine Translation Evaluation with Entailment Features*
Sebastian Pado, Michel Galley, Dan Jurafsky and Christopher D. Manning

15:55–16:20 *The Contribution of Linguistic Features to Automatic Machine Translation Evaluation*
Enrique Amigó, Jesús Giménez, Julio Gonzalo and Felisa Verdejo

16:20–16:45 *A Syntax-Driven Bracketing Model for Phrase-Based Translation*
Deyi Xiong, Min Zhang, Aiti Aw and Haizhou Li

16:45–17:10 *Topological Ordering of Function Words in Hierarchical Phrase-based Translation*
Hendra Setiawan, Min Yen Kan, Haizhou Li and Philip Resnik

17:10–17:35 *Phrase-Based Statistical Machine Translation as a Traveling Salesman Problem*
Mikhail Zaslavskiy, Marc Dymetman and Nicola Cancedda

Monday, August 3, 2009 (continued)

Session 3B: Syntax and Parsing 2
Chaired by Dan Klein

15:30–15:55 *Concise Integer Linear Programming Formulations for Dependency Parsing*
Andre Martins, Noah Smith and Eric Xing

15:55–16:20 *Non-Projective Dependency Parsing in Expected Linear Time*
Joakim Nivre

16:20–16:45 *Semi-supervised Learning of Dependency Parsers using Generalized Expectation Criteria*
Gregory Druck, Gideon Mann and Andrew McCallum

16:45–17:10 *Dependency Grammar Induction via Bitext Projection Constraints*
Kuzman Ganchev, Jennifer Gillenwater and Ben Taskar

17:10–17:35 *Cross-Domain Dependency Parsing Using a Deep Linguistic Grammar*
Yi Zhang and Rui Wang

Session 3C: Information Extraction 1
Chaired by Eduard Hovy

15:30–15:55 *A Chinese-English Organization Name Translation System Using Heuristic Web Mining and Asymmetric Alignment*
Fan Yang, Jun Zhao and Kang Liu

15:55–16:20 *Reducing Semantic Drift with Bagging and Distributional Similarity*
Tara McIntosh and James R. Curran

16:20–16:45 *Jointly Identifying Temporal Relations with Markov Logic*
Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara and Yuji Matsumoto

16:45–17:10 *Profile Based Cross-Document Coreference Using Kernelized Fuzzy Relational Clustering*
Jian Huang, Sarah M. Taylor, Jonathan L. Smith, Konstantinos A. Fotiadis and C. Lee Giles

17:10–17:35 *Who, What, When, Where, Why? Comparing Multiple Approaches to the Cross-Lingual 5W Task*
Kristen Parton, Kathleen R. McKeown, Bob Coyne, Mona T. Diab, Ralph Grishman, Dilek Hakkani-Tür, Mary Harper, Heng Ji, Wei Yun Ma, Adam Meyers, Sara Stolbach, Ang Sun, Gokhan Tur, Wei Xu and Sibel Yaman

Monday, August 3, 2009 (continued)

Session 3D: Semantics 2
Chaired by Patrick Pantel

15:30–15:55 *Bilingual Co-Training for Monolingual Hyponymy-Relation Acquisition*
Jong-Hoon Oh, Kiyotaka Uchimoto and Kentaro Torisawa

15:55–16:20 *Automatic Set Instance Extraction using the Web*
Richard C. Wang and William W. Cohen

16:20–16:45 *Extracting Lexical Reference Rules from Wikipedia*
Eyal Shnarch, Libby Barak and Ido Dagan

16:45–17:10 *Employing Topic Models for Pattern-based Semantic Class Discovery*
Huibin Zhang, Mingjie Zhu, Shuming Shi and Ji-Rong Wen

17:10–17:35 *Paraphrase Identification as Probabilistic Quasi-Synchronous Recognition*
Dipanjan Das and Noah A. Smith

Tuesday, August 4, 2009

Session 4A: Statistical and Machine Learning Methods 2
Chaired by Hal Daumé III

08:30–08:55 *Stochastic Gradient Descent Training for L1-regularized Log-linear Models with Cumulative Penalty*
Yoshimasa Tsuruoka, Jun'ichi Tsujii and Sophia Ananiadou

08:55–09:20 *A global model for joint lemmatization and part-of-speech prediction*
Kristina Toutanova and Colin Cherry

09:20–09:45 *Distributional Representations for Handling Sparsity in Supervised Sequence-Labeling*
Fei Huang and Alexander Yates

Tuesday, August 4, 2009 (continued)

Session 4B: Word Segmentation and POS Tagging
Chaired by Hwee Tou Ng

08:30–08:55 *Minimized Models for Unsupervised Part-of-Speech Tagging*
Sujith Ravi and Kevin Knight

08:55–09:20 *An Error-Driven Word-Character Hybrid Model for Joint Chinese Word Segmentation and POS Tagging*
Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa and Hitoshi Isahara

09:20–09:45 *Automatic Adaptation of Annotation Standards: Chinese Word Segmentation and POS Tagging – A Case Study*
Wenbin Jiang, Liang Huang and Qun Liu

Session 4C: Spoken Language Processing 1
Chaired by Brian Roark

08:30–08:55 *Linefeed Insertion into Japanese Spoken Monologue for Captioning*
Tomohiro Ohno, Masaki Murata and Shigeki Matsubara

08:55–09:20 *Semi-supervised Learning for Automatic Prosodic Event Detection Using Co-training Algorithm*
Je Hun Jeon and Yang Liu

09:20–09:45 *Summarizing multiple spoken documents: finding evidence from untranscribed audio*
Xiaodan Zhu, Gerald Penn and Frank Rudzicz

Session 4DI: Short Paper 1 (Syntax and Parsing)

Session 4DII: Short Paper 2 (Discourse and Dialogue)

09:45–10:15 Break

Tuesday, August 4, 2009 (continued)

Session 5A: Machine Translation 3
Chaired by Dan Gildea

10:15–10:40 *Improving Tree-to-Tree Translation with Packed Forests*
Yang Liu, Yajuan Lü and Qun Liu

10:40–11:05 *Fast Consensus Decoding over Translation Forests*
John DeNero, David Chiang and Kevin Knight

11:05–11:30 *Joint Decoding with Multiple Translation Models*
Yang Liu, Haitao Mi, Yang Feng and Qun Liu

11:30–11:55 *Collaborative Decoding: Partial Hypothesis Re-ranking Using Translation Consensus between Decoders*
Mu Li, Nan Duan, Dongdong Zhang, Chi-Ho Li and Ming Zhou

11:55–12:20 *Variational Decoding for Statistical Machine Translation*
Zhifei Li, Jason Eisner and Sanjeev Khudanpur

Session 5B: Semantics 3
Chaired by Diana McCarthy

10:15–10:40 *Unsupervised Learning of Narrative Schemas and their Participants*
Nathanael Chambers and Dan Jurafsky

10:40–11:05 *Learning a Compositional Semantic Parser using an Existing Syntactic Parser*
Ruifang Ge and Raymond Mooney

11:05–11:30 *Latent Variable Models of Concept-Attribute Attachment*
Joseph Reisinger and Marius Pasca

11:30–11:55 *The Chinese Aspect Generation Based on Aspect Selection Functions*
Guowen Yang and John Bateman

11:55–12:20 *Quantitative modeling of the neural representation of adjective-noun phrases to account for fMRI activation*
Kai-min K. Chang, Vladimir L. Cherkassky, Tom M. Mitchell and Marcel Adam Just

Tuesday, August 4, 2009 (continued)

Session 5C: Discourse and Dialogue 1
Chaired by Kathy McKeown

10:15–10:40 *Capturing Salience with a Trainable Cache Model for Zero-anaphora Resolution*
Ryu Iida, Kentaro Inui and Yuji Matsumoto

10:40–11:05 *Conundrums in Noun Phrase Coreference Resolution: Making Sense of the State-of-the-Art*
Veselin Stoyanov, Nathan Gilbert, Claire Cardie and Ellen Riloff

11:05–11:30 *A Novel Discourse Parser Based on Support Vector Machine Classification*
David duVerle and Helmut Prendinger

11:30–11:55 *Genre distinctions for discourse in the Penn TreeBank*
Bonnie Webber

11:55–12:20 *Automatic sense prediction for implicit discourse relations in text*
Emily Pitler, Annie Louis and Ani Nenkova

Session 5D: Student Research Workshop

12:20–14:20 Short Paper Poster / SRW Poster Session (Lunch)

Session 6A: Short Paper 3 (Machine Translation)

Session 6B: Short Paper 4 (Semantics)

Session 6C: Short Paper 5 (Spoken Language Processing)

Tuesday, August 4, 2009 (continued)

Session 6D: Short Paper 6 (Statistical and Machine Learning Methods 1)

Session 6E: Short Paper 7 (Summarization and Generation)

Session 6F: Short Paper 8 (Sentiment Analysis)

Session 6G: Short Paper 9 (Question Answering)

Session 6H: Short Paper 10 (Statistical and Machine Learning Methods 2)

16:25–16:50 Break

Session 7A: Sentiment Analysis and Text Categorization 2
Chaired by Bing Liu

16:50–17:15 *A Framework of Feature Selection Methods for Text Categorization*
Shoushan Li, Rui Xia, Chengqing Zong and Chu-Ren Huang

17:15–17:40 *Mine the Easy, Classify the Hard: A Semi-Supervised Approach to Automatic Sentiment Classification*
Sajib Dasgupta and Vincent Ng

17:40–18:05 *Modeling Latent Biographic Attributes in Conversational Genres*
Nikesh Garera and David Yarowsky

Session 7B: Question Answering
Chaired by Hae-Chang Rim

16:50–17:15 *A Graph-based Semi-Supervised Learning for Question-Answering*
Asli Celikyilmaz, Marcus Thint and Zhiheng Huang

17:15–17:40 *Combining Lexical Semantic Resources with Question & Answer Archives for Translation-Based Answer Finding*
Delphine Bernhard and Iryna Gurevych

17:40–18:05 *Answering Opinion Questions with Random Walks on Graphs*
Fangtao Li, Yang Tang, Minlie Huang and Xiaoyan Zhu

Tuesday, August 4, 2009 (continued)

Session 7C: Spoken Language Processing 2
Chaired by Yang Liu

16:50–17:15 *What lies beneath: Semantic and syntactic analysis of manually reconstructed spontaneous speech*
Erin Fitzgerald, Frederick Jelinek and Robert Frank

17:15–17:40 *Discriminative Lexicon Adaptation for Improved Character Accuracy - A New Direction in Chinese Language Modeling*
Yi-cheng Pan, Lin-shan Lee and Sadaoki Furui

17:40–18:05 *Improving Automatic Speech Recognition for Lectures through Transformation-based Rules Learned from Minimal Data*
Cosmin Munteanu, Gerald Penn and Xiaodan Zhu

Session 7D: Short Paper 11 (Information Extraction)

Wednesday, August 5, 2009

08:30–09:30 Invited Talk: Bonnie Webber, Discourse – Early Problems, Current Successes, Future Challenges

09:30–10:00 Break

Session 8A: Machine Translation 4
Chaired by Dekai Wu

09:55–10:20 *Quadratic-Time Dependency Parsing for Machine Translation*
Michel Galley and Christopher D. Manning

10:20–10:45 *A Gibbs Sampler for Phrasal Synchronous Grammar Induction*
Phil Blunsom, Trevor Cohn, Chris Dyer and Miles Osborne

10:45–11:10 *Source-Language Entailment Modeling for Translating Unknown Terms*
Shachar Mirkin, Lucia Specia, Nicola Cancedda, Ido Dagan, Marc Dymetman and Idan Szpektor

11:10–11:35 *Case markers and Morphology: Addressing the crux of the fluency problem in English-Hindi SMT*
Ananthakrishnan Ramanathan, Hansraj Choudhary, Avishek Ghosh and Pushpak Bhattacharyya

Session 8B: Generation and Summarization 2
Chaired by Regina Barzilay

09:55–10:20 *Dependency Based Chinese Sentence Realization*
Wei He, Haifeng Wang, Yuqing Guo and Ting Liu

10:20–10:45 *Incorporating Information Status into Generation Ranking*
Aoife Cahill and Arndt Riester

10:45–11:10 *A Syntax-Free Approach to Japanese Sentence Compression*
Tsutomu Hirao, Jun Suzuki and Hideki Isozaki

11:10–11:35 *Application-driven Statistical Paraphrase Generation*
Shiqi Zhao, Xiang Lan, Ting Liu and Sheng Li

Session 8C: Text Mining and NLP applications
Chaired by Sophia Ananiodou

09:55–10:20 *Semi-Supervised Cause Identification from Aviation Safety Reports*
Isaac Persing and Vincent Ng

10:20–10:45 *SMS based Interface for FAQ Retrieval*
Govind Kothari, Sumit Negi, Tanveer A. Faruquie, Venkatesan T. Chakaravarthy and L. Venkata Subramaniam

10:45–11:10 *Semantic Tagging of Web Search Queries*
Mehdi Manshadi and Xiao Li

11:10–11:35 *Mining Bilingual Data from the Web with Adaptively Learnt Patterns*
Long Jiang, Shiquan Yang, Ming Zhou, Xiaohua Liu and Qingsheng Zhu

Wednesday, August 5, 2009 (continued)

Session 8D: Discourse and Dialogue 2
Chaired by Gary Geunbae Lee

09:55–10:20 *Comparing Objective and Subjective Measures of Usability in a Human-Robot Dialogue System*
Mary Ellen Foster, Manuel Giuliani and Alois Knoll

10:20–10:45 *Setting Up User Action Probabilities in User Simulations for Dialog System Development*
Hua Ai and Diane Litman

10:45–11:10 *Dialogue Segmentation with Large Numbers of Volunteer Internet Annotators*
T. Daniel Midgley

11:10–11:35 *Robust Approach to Abbreviating Terms: A Discriminative Latent Variable Model with Global Information*
Xu Sun, Naoaki Okazaki and Jun'ichi Tsujii

11:35–12:30 Lunch

12:30–14:00 Business Meeting

14:00–14:25 Break

Session 9A: Machine Translation 5
Chaired by Kevin Knight

14:25–14:50 *A non-contiguous Tree Sequence Alignment-based Model for Statistical Machine Translation*
Jun Sun, Min Zhang and Chew Lim Tan

14:50–15:15 *Better Word Alignments with Supervised ITG Models*
Aria Haghighi, John Blitzer, John DeNero and Dan Klein

15:15–15:40 *Confidence Measure for Word Alignment*
Fei Huang

15:40–16:05 *A Comparative Study of Hypothesis Alignment and its Improvement for Machine Translation System Combination*
Boxing Chen, Min Zhang, Haizhou Li and Aiti Aw

16:05–16:30 *Incremental HMM Alignment for MT System Combination*
Chi-Ho Li, Xiaodong He, Yupeng Liu and Ning Xi

Wednesday, August 5, 2009 (continued)

Session 9B: Syntax and Parsing 3
Chaired by James Curran

14:25–14:50 *K-Best A* Parsing*
Adam Pauls and Dan Klein

14:50–15:15 *Coordinate Structure Analysis with Global Structural Constraints and Alignment-Based Local Features*
Kazuo Hara, Masashi Shimbo, Hideharu Okuma and Yuji Matsumoto

15:15–15:40 *Learning Context-Dependent Mappings from Sentences to Logical Form*
Luke Zettlemoyer and Michael Collins

15:40–16:05 *An Optimal-Time Binarization Algorithm for Linear Context-Free Rewriting Systems with Fan-Out Two*
Carlos Gómez-Rodríguez and Giorgio Satta

16:05–16:30 *A Polynomial-Time Parsing Algorithm for TT-MCTAG*
Laura Kallmeyer and Giorgio Satta

Session 9C: Information Extraction 2
Chaired by Jian Su

14:25–14:50 *Distant supervision for relation extraction without labeled data*
Mike Mintz, Steven Bills, Rion Snow and Daniel Jurafsky

14:50–15:15 *Multi-Task Transfer Learning for Weakly-Supervised Relation Extraction*
Jing Jiang

15:15–15:40 *Unsupervised Relation Extraction by Mining Wikipedia Texts Using Information from the Web*
Yulan Yan, Naoaki Okazaki, Yutaka Matsuo, Zhenglu Yang and Mitsuru Ishizuka

15:40–16:05 *Phrase Clustering for Discriminative Learning*
Dekang Lin and Xiaoyun Wu

16:05–16:30 *Semi-Supervised Active Learning for Sequence Labeling*
Katrin Tomanek and Udo Hahn

Wednesday, August 5, 2009 (continued)

Sssion 9D: Information Retrieval
Chaired by Kam-Fai Wong

14:25–14:50 *Word or Phrase? Learning Which Unit to Stress for Information Retrieval*
Young-In Song, Jung-Tae Lee and Hae-Chang Rim

14:50–15:15 *A Generative Blog Post Retrieval Model that Uses Query Expansion based on External Collections*
Wouter Weerkamp, Krisztian Balog and Maarten de Rijke

15:15–15:40 *Language Identification of Search Engine Queries*
Hakan Ceylan and Yookyung Kim

15:40–16:05 *Exploiting Bilingual Information to Improve Web Search*
Wei Gao, John Blitzer, Ming Zhou and Kam-Fai Wong

16:35-18:00 Best Paper Awards, Lifetime Achievement Award and Presentation

18:00–18:30 Closing Session

Heterogeneous Transfer Learning for Image Clustering via the Social Web

Qiang Yang
Hong Kong University of Science and Technology, Clearway Bay, Kowloon, Hong Kong
qyang@cs.ust.hk

Yuqiang Chen **Gui-Rong Xue** **Wenyuan Dai** **Yong Yu**
Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai 200240, China
{yuqiangchen,grxue,dwyak,yyu}@apex.sjtu.edu.cn

Abstract

In this paper, we present a new learning scenario, *heterogeneous transfer learning*, which improves learning performance when the data can be in different feature spaces and where no correspondence between data instances in these spaces is provided. In the past, we have classified Chinese text documents using English training data under the heterogeneous transfer learning framework. In this paper, we present image clustering as an example to illustrate how unsupervised learning can be improved by transferring knowledge from auxiliary heterogeneous data obtained from the social Web. Image clustering is useful for image sense disambiguation in query-based image search, but its quality is often low due to image-data sparsity problem. We extend PLSA to help transfer the knowledge from social Web data, which have mixed feature representations. Experiments on image-object clustering and scene clustering tasks show that our approach in heterogeneous transfer learning based on the auxiliary data is indeed effective and promising.

1 Introduction

Traditional machine learning relies on the availability of a large amount of data to train a model, which is then applied to test data in the same feature space. However, labeled data are often scarce and expensive to obtain. Various machine learning strategies have been proposed to address this problem, including semi-supervised learning (Zhu, 2007), domain adaptation (Wu and Dietterich, 2004; Blitzer et al., 2006; Blitzer et al., 2007; Arnold et al., 2007; Chan and Ng, 2007; Daume, 2007; Jiang and Zhai, 2007; Reichart and Rappoport, 2007; Andreevskaia and Bergler, 2008), multi-task learning (Caruana, 1997; Reichart et al., 2008; Arnold et al., 2008), self-taught learning (Raina et al., 2007), etc. A commonality among these methods is that they all require the training data and test data to be in the same feature space. In addition, most of them are designed for supervised learning. However, in practice, we often face the problem where the labeled data are scarce in their own feature space, whereas there may be a large amount of labeled heterogeneous data in another feature space. In such situations, it would be desirable to transfer the knowledge from heterogeneous data to domains where we have relatively little training data available.

To learn from heterogeneous data, researchers have previously proposed multi-view learning (Blum and Mitchell, 1998; Nigam and Ghani, 2000) in which each instance has multiple views in different feature spaces. Different from previous works, we focus on the problem of *heterogeneous transfer learning*, which is designed for situation when the training data are in one feature space (such as text), and the test data are in another (such as images), and there may be no correspondence between instances in these spaces. The type of heterogeneous data can be very different, as in the case of text and image. To consider how heterogeneous transfer learning relates to other types of learning, Figure 1 presents an intuitive illustration of four learning strategies, including traditional machine learning, transfer learning across different distributions, multi-view learning and heterogeneous transfer learning. As we can see, an important distinguishing feature of heterogeneous transfer learning, as compared to other types of learning, is that more constraints on the problem are relaxed, such that data instances do not need to correspond anymore. This allows, for example, a collection of Chinese text documents to be classified using another collection of English text as the

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 1–9,
Suntec, Singapore, 2-7 August 2009.

training data (c.f. (Ling et al., 2008) and Section 2.1).

In this paper, we will give an illustrative example of heterogeneous transfer learning to demonstrate how the task of image clustering can benefit from learning from the heterogeneous social Web data. A major motivation of our work is Web-based image search, where users submit textual queries and browse through the returned result pages. One problem is that the user queries are often ambiguous. An ambiguous keyword such as "Apple" might retrieve images of Apple computers and mobile phones, or images of fruits. Image clustering is an effective method for improving the accessibility of image search result. Loeff et al. (2006) addressed the image clustering problem with a focus on image sense discrimination. In their approach, images associated with textual features are used for clustering, so that the text and images are clustered at the same time. Specifically, spectral clustering is applied to the distance matrix built from a multimodal feature set associated with the images to get a better feature representation. This new representation contains both image and text information, with which the performance of image clustering is shown to be improved. A problem with this approach is that when images contained in the Web search results are very scarce and when the textual data associated with the images are very few, clustering on the images and their associated text may not be very effective.

Different from these previous works, in this paper, we address the image clustering problem as a *heterogeneous transfer learning* problem. We aim to leverage heterogeneous auxiliary data, social annotations, etc. to enhance image clustering performance. We observe that the World Wide Web has many annotated images in Web sites such as Flickr (`http://www.flickr.com`), which can be used as auxiliary information source for our clustering task. In this work, our objective is to cluster a small collection of images that we are interested in, where these images are not sufficient for traditional clustering algorithms to perform well due to data sparsity and the low level of image features. We investigate how to utilize the readily available socially annotated image data on the Web to improve image clustering. Although these auxiliary data may be irrelevant to the images to be clustered and cannot be directly used to solve the data sparsity problem, we show that they can still be used to estimate a good *latent feature representation*, which can be used to improve image clustering.

2 Related Works

2.1 Heterogeneous Transfer Learning Between Languages

In this section, we summarize our previous work on cross-language classification as an example of heterogeneous transfer learning. This example is related to our image clustering problem because they both rely on data from different feature spaces.

As the World Wide Web in China grows rapidly, it has become an increasingly important problem to be able to accurately classify Chinese Web pages. However, because the labeled Chinese Web pages are still not sufficient, we often find it difficult to achieve high accuracy by applying traditional machine learning algorithms to the Chinese Web pages directly. Would it be possible to make the best use of the relatively abundant labeled English Web pages for classifying the Chinese Web pages?

To answer this question, in (Ling et al., 2008), we developed a novel approach for classifying the Web pages in Chinese using the training documents in English. In this subsection, we give a brief summary of this work. The problem to be solved is: we are given a collection of labeled English documents and a large number of unlabeled Chinese documents. The English and Chinese texts are not aligned. Our objective is to classify the Chinese documents into the same label space as the English data.

Our key observation is that even though the data use different text features, they may still share many of the same semantic information. What we need to do is to uncover this latent semantic information by finding out what is common among them. We did this in (Ling et al., 2008) by using the information bottleneck theory (Tishby et al., 1999). In our work, we first translated the Chinese document into English automatically using some available translation software, such as Google translate. Then, we encoded the training text as well as the translated target text together, in terms of the information theory. We allowed all the information to be put through a 'bottleneck' and be represented by a limited number of *code-*

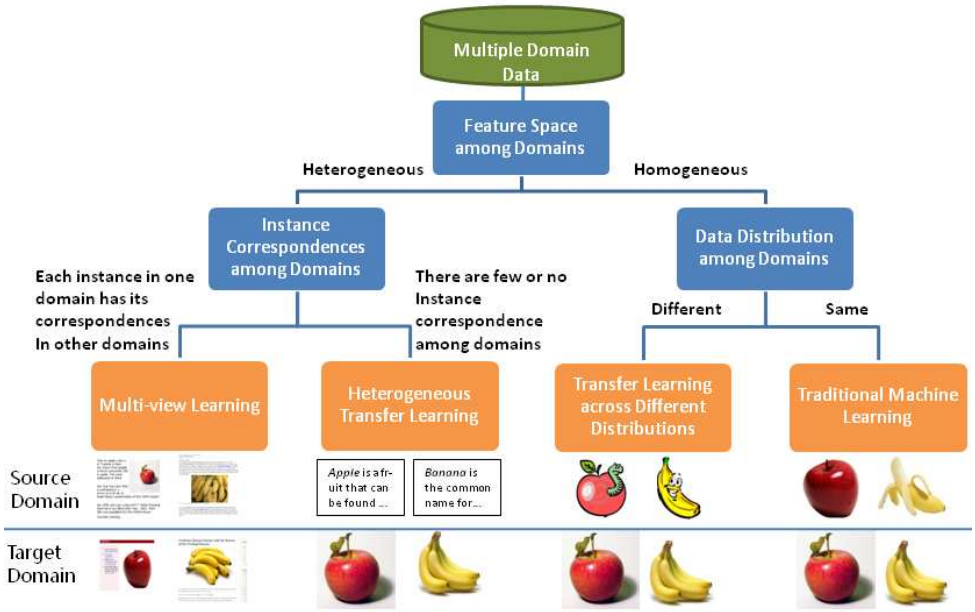


Figure 1: An intuitive illustration of different kinds learning strategies using classification/clustering of image `apple` and `banana` as the example.

words (i.e. labels in the classification problem). Finally, information bottleneck was used to maintain most of the common information between the two data sources, and discard the remaining irrelevant information. In this way, we can approximate the ideal situation where similar training and translated test pages shared in the common part are encoded into the same codewords, and are thus assigned the correct labels. In (Ling et al., 2008), we experimentally showed that heterogeneous transfer learning can indeed improve the performance of cross-language text classification as compared to directly training learning models (e.g., Naive Bayes or SVM) and testing on the translated texts.

2.2 Other Works in Transfer Learning

In the past, several other works made use of transfer learning for cross-feature-space learning. Wu and Oard (2008) proposed to handle the cross-language learning problem by translating the data into a same language and applying kNN on the latent topic space for classification. Most learning algorithms for dealing with cross-language heterogeneous data require a *translator* to convert the data to the same feature space. For those data that are in different feature spaces where no translator is available, Davis and Domingos (2008) proposed a Markov-logic-based transfer learning algorithm, which is called *deep transfer*, for transferring knowledge between biological domains and Web domains. Dai et al. (2008a) proposed a novel learning paradigm, known as translated learning, to deal with the problem of learning heterogeneous data that belong to quite different feature spaces by using a risk minimization framework.

2.3 Relation to PLSA

Our work makes use of PLSA. *Probabilistic latent semantic analysis* (PLSA) is a widely used probabilistic model (Hofmann, 1999), and could be considered as a probabilistic implementation of *latent semantic analysis* (LSA) (Deerwester et al., 1990). An extension to PLSA was proposed in (Cohn and Hofmann, 2000), which incorporated the hyperlink connectivity in the PLSA model by using a joint probabilistic model for connectivity and content. Moreover, PLSA has shown a lot of applications ranging from text clustering (Hofmann, 2001) to image analysis (Sivic et al., 2005).

2.4 Relation to Clustering

Compared to many previous works on image clustering, we note that traditional image clustering is generally based on techniques such as K-means (MacQueen, 1967) and hierarchical clustering (Kaufman and Rousseeuw, 1990). However, when the data are sparse, traditional clustering algorithms may have difficulties in obtaining high-quality image clusters. Recently, several researchers have investigated how to leverage the auxiliary information to improve target clustering

performance, such as supervised clustering (Finley and Joachims, 2005), semi-supervised clustering (Basu et al., 2004), self-taught clustering (Dai et al., 2008b), etc.

3 Image Clustering with Annotated Auxiliary Data

In this section, we present our *annotation-based probabilistic latent semantic analysis* algorithm (aPLSA), which extends the traditional PLSA model by incorporating annotated auxiliary image data. Intuitively, our algorithm aPLSA performs PLSA analysis on the target images, which are converted to an image instance-to-feature co-occurrence matrix. At the same time, PLSA is also applied to the annotated image data from social Web, which is converted into a text-to-image-feature co-occurrence matrix. In order to unify those two separate PLSA models, these two steps are done simultaneously with common latent variables used as a bridge linking them. Through these common latent variables, which are now constrained by both target image data and auxiliary annotation data, a better clustering result is expected for the target data.

3.1 Probabilistic Latent Semantic Analysis

Let $\mathcal{F} = \{f_i\}_{i=1}^{|\mathcal{F}|}$ be an image feature space, and $\mathcal{V} = \{v_i\}_{i=1}^{|\mathcal{V}|}$ be the image data set. Each image $v_i \in \mathcal{V}$ is represented by a *bag-of-features* $\{f|f \in v_i \wedge f \in \mathcal{F}\}$.

Based on the image data set $\mathcal{V}$, we can estimate an image instance-to-feature co-occurrence matrix $A^{|\mathcal{V}|\times|\mathcal{F}|} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{F}|}$, where each element A_{ij} ($1 \le i \le |\mathcal{V}|$ and $1 \le j \le |\mathcal{F}|$) in the matrix A is the frequency of the feature f_j appearing in the instance v_i.

Let $\mathcal{W} = \{w_i\}_{i=1}^{|\mathcal{W}|}$ be a text feature space. The annotated image data allow us to obtain the co-occurrence information between images v and text features $w \in \mathcal{W}$. An example of annotated image data is the Flickr (`http://www.flickr.com`), which is a social Web site containing a large number of annotated images.

By extracting image features from the annotated images v, we can estimate a text-to-image feature co-occurrence matrix $B^{|\mathcal{W}|\times|\mathcal{F}|} \in \mathbb{R}^{|\mathcal{W}|\times|\mathcal{F}|}$, where each element B_{ij} ($1 \le i \le |\mathcal{W}|$ and $1 \le j \le |\mathcal{F}|$) in the matrix B is the frequency of the text feature w_i and the image feature f_j occurring together in the annotated image data set.

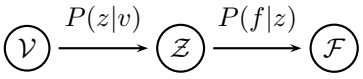


Figure 2: Graphical model representation of PLSA model.

Let $\mathcal{Z} = \{z_i\}_{i=1}^{|\mathcal{Z}|}$ be the latent variable set in our aPLSA model. In clustering, each latent variable $z_i \in \mathcal{Z}$ corresponds to a certain cluster.

Our objective is to estimate a clustering function $g : \mathcal{V} \mapsto \mathcal{Z}$ with the help of the two co-occurrence matrices A and B as defined above.

To formally introduce the aPLSA model, we start from the *probabilistic latent semantic analysis* (PLSA) (Hofmann, 1999) model. PLSA is a probabilistic implementation of *latent semantic analysis* (LSA) (Deerwester et al., 1990). In our image clustering task, PLSA decomposes the instance-feature co-occurrence matrix A under the assumption of conditional independence of image instances $\mathcal{V}$ and image features $\mathcal{F}$, given the latent variables $\mathcal{Z}$.

$$P(f|v) = \sum_{z\in\mathcal{Z}} P(f|z)P(z|v). \tag{1}$$

The graphical model representation of PLSA is shown in Figure 2.

Based on the PLSA model, the log-likelihood can be defined as:

$$\mathcal{L} = \sum_i \sum_j \frac{A_{ij}}{\sum_{j'} A_{ij'}} \log P(f_j|v_i) \tag{2}$$

where $A^{|\mathcal{V}|\times|\mathcal{F}|} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{F}|}$ is the image instance-feature co-occurrence matrix. The term $\frac{A_{ij}}{\sum_{j'} A_{ij'}}$ in Equation (2) is a normalization term ensuring each image is giving the same weight in the log-likelihood.

Using EM algorithm (Dempster et al., 1977), which locally maximizes the log-likelihood of the PLSA model (Equation (2)), the probabilities $P(f|z)$ and $P(z|v)$ can be estimated. Then, the clustering function is derived as

$$g(v) = \operatorname*{argmax}_{z\in\mathcal{Z}} P(z|v). \tag{3}$$

Due to space limitation, we omit the details for the PLSA model, which can be found in (Hofmann, 1999).

3.2 aPLSA: Annotation-based PLSA

In this section, we consider how to incorporate a large number of socially annotated images in a

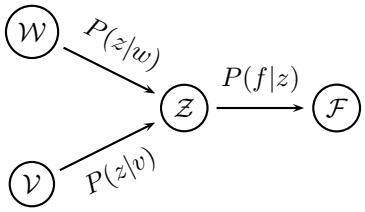


Figure 3: Graphical model representation of aPLSA model.

unified PLSA model for the purpose of utilizing the correlation between text features and image features. In the auxiliary data, each image has certain textual tags that are attached by users. The correlation between text features and image features can be formulated as follows.

$$P(f|w) = \sum_{z \in \mathcal{Z}} P(f|z)P(z|w). \tag{4}$$

It is clear that Equations (1) and (4) share a same term $P(f|z)$. So we design a new PLSA model by joining the probabilistic model in Equation (1) and the probabilistic model in Equation (4) into a unified model, as shown in Figure 3. In Figure 3, the latent variables $\mathcal{Z}$ depend not only on the correlation between image instances $\mathcal{V}$ and image features $\mathcal{F}$, but also the correlation between text features $\mathcal{W}$ and image features $\mathcal{F}$. Therefore, the auxiliary socially-annotated image data can be used to help the target image clustering performance by estimating good set of latent variables $\mathcal{Z}$.

Based on the graphical model representation in Figure 3, we derive the log-likelihood objective function, in a similar way as in (Cohn and Hofmann, 2000), as follows

$$\mathcal{L} = \sum_j \left[\lambda \sum_i \frac{A_{ij}}{\sum_{j'} A_{ij'}} \log P(f_j|v_i) + (1-\lambda) \sum_l \frac{B_{lj}}{\sum_{j'} B_{lj'}} \log P(f_j|w_l) \right], \tag{5}$$

where $A^{|\mathcal{V}|\times|\mathcal{F}|} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{F}|}$ is the image instance-feature co-occurrence matrix, and $B^{|\mathcal{W}|\times|\mathcal{F}|} \in \mathbb{R}^{|\mathcal{W}|\times|\mathcal{F}|}$ is the text-to-image feature-level co-occurrence matrix. Similar to Equation (2), $\frac{A_{ij}}{\sum_{j'} A_{ij'}}$ and $\frac{B_{lj}}{\sum_{j'} B_{lj'}}$ in Equation (5) are the normalization terms to prevent imbalanced cases.

Furthermore, λ acts as a trade-off parameter between the co-occurrence matrices A and B. In the extreme case when $\lambda = 1$, the log-likelihood objective function ignores all the biases from the text-to-image occurrence matrix B. In this case, the aPLSA model degenerates to the traditional PLSA model. Therefore, aPLSA is an extension to the PLSA model.

Now, the objective is to maximize the log-likelihood $\mathcal{L}$ of the aPLSA model in Equation (5). Then we apply the EM algorithm (Dempster et al., 1977) to estimate the conditional probabilities $P(f|z)$, $P(z|w)$ and $P(z|v)$ with respect to each dependence in Figure 3 as follows.

- E-Step: calculate the posterior probability of each latent variable z given the observation of image features f, image instances v and text features w based on the old estimate of $P(f|z)$, $P(z|w)$ and $P(z|v)$:

$$P(z_k|v_i, f_j) = \frac{P(f_j|z_k)P(z_k|v_i)}{\sum_{k'} P(f_j|z_{k'})P(z_{k'}|v_i)} \tag{6}$$

$$P(z_k|w_l, f_j) = \frac{P(f_j|z_k)P(z_k|w_l)}{\sum_{k'} P(f_j|z_{k'})P(z_{k'}|w_l)} \tag{7}$$

- M-Step: re-estimates conditional probabilities $P(z_k|v_i)$ and $P(z_k|w_l)$:

$$P(z_k|v_i) = \sum_j \frac{A_{ij}}{\sum_{j'} A_{ij'}} P(z_k|v_i, f_j) \tag{8}$$

$$P(z_k|w_l) = \sum_j \frac{B_{lj}}{\sum_{j'} B_{lj'}} P(z_k|w_l, f_j) \tag{9}$$

  and conditional probability $P(f_j|z_k)$, which is a mixture portion of posterior probability of latent variables

$$P(f_j|z_k) \propto \lambda \sum_i \frac{A_{ij}}{\sum_{j'} A_{ij'}} P(z_k|v_i, f_j) + (1-\lambda) \sum_l \frac{B_{lj}}{\sum_{j'} B_{lj'}} P(z_k|w_l, f_j) \tag{10}$$

Finally, the clustering function for a certain image v is

$$g(v) = \operatorname*{argmax}_{z \in \mathcal{Z}} P(z|v). \tag{11}$$

From the above equations, we can derive our annotation-based probabilistic latent semantic analysis (aPLSA) algorithm. As shown in Algorithm 1, aPLSA iteratively performs the E-Step and the M-Step in order to seek local optimal points based on the objective function $\mathcal{L}$ in Equation (5).

Algorithm 1 Annotation-based PLSA Algorithm (aPLSA)

Input: The $\mathcal{V}$-$\mathcal{F}$ co-occurrence matrix A and $\mathcal{W}$-$\mathcal{F}$ co-occurrence matrix B.

Output: A clustering (partition) function $g : \mathcal{V} \mapsto \mathcal{Z}$, which maps an image instance $v \in \mathcal{V}$ to a latent variable $z \in \mathcal{Z}$.

1: Initial $\mathcal{Z}$ so that $|\mathcal{Z}|$ equals the number clusters desired.
2: Initialize $P(z|v)$, $P(z|w)$, $P(f|z)$ randomly.
3: **while** the change of $\mathcal{L}$ in Eq. (5) between two sequential iterations is greater than a predefined threshold **do**
4: E-Step: Update $P(z|v, f)$ and $P(z|w, f)$ based on Eq. (6) and (7) respectively.
5: M-Step: Update $P(z|v)$, $P(z|w)$ and $P(f|z)$ based on Eq. (8), (9) and (10) respectively.
6: **end while**
7: **for all** v in $\mathcal{V}$ **do**
8: $g(v) \leftarrow \operatorname{argmax}_z P(z|v)$.
9: **end for**
10: Return g.

4 Experiments

In this section, we empirically evaluate the aPLSA algorithm together with some state-of-art baseline methods on two widely used image corpora, to demonstrate the effectiveness of our algorithm aPLSA.

4.1 Data Sets

In order to evaluate the effectiveness of our algorithm aPLSA, we conducted experiments on several data sets generated from two image corpora, Caltech-256 (Griffin et al., 2007) and the fifteen-scene (Lazebnik et al., 2006). The Caltech-256 data set has 256 image objective categories, ranging from animals to buildings, from plants to automobiles, etc. The fifteen-scene data set contains 15 scenes such as store and forest. From these two corpora, we randomly generated eleven image clustering tasks, including seven 2-way clustering tasks, two 4-way clustering task, one 5-way clustering task and one 8-way clustering task. The detailed descriptions for these clustering tasks are given in Table 1. In these tasks, bi7 and oct1 were generated from fifteen-scene data set, and the rest were from Caltech-256 data set.

DATA SET	INVOLVED CLASSES	DATA SIZE
bi1	skateboard, airplanes	102, 800
bi2	billiards, mars	278, 155
bi3	cd, greyhound	102, 94
bi4	electric-guitar, snake	122, 112
bi5	calculator, dolphin	100, 106
bi6	mushroom, teddy-bear	202, 99
bi7	MIThighway, livingroom	260, 289
quad1	calculator, diamond-ring, dolphin, microscope	100, 118, 106, 116
quad2	bonsai, comet, frog, saddle	122, 120, 115, 110
quint1	frog, kayak, bear, jesus-christ, watch	115, 102, 101, 87, 201
oct1	MIThighway, MITmountain, kitchen, MITcoast, PARoffice, MITtallbuilding, livingroom, bedroom	260, 374, 210, 360, 215, 356, 289, 216
tune1	coin, horse	123, 270
tune2	socks, spider	111, 106
tune3	galaxy, snowmobile	80, 112
tune4	dice, fern	98, 110
tune5	backpack, lightning, mandolin, swan	151, 136, 93, 114

Table 1: The descriptions of all the image clustering tasks used in our experiment. Among these data sets, bi7 and oct1 were generated from *fifteen-scene* data set, and the rest were from *Caltech-256* data set.

To empirically investigate the parameter λ and the convergence of our algorithm aPLSA, we generated five more date sets as the development sets. The detailed description of these five development sets, namely tune1 to tune5 is listed in Table 1 as well.

The auxiliary data were crawled from the Flickr (http://www.flickr.com/) web site during August 2007. Flickr is an internet community where people share photos online and express their opinions as social tags (annotations) attached to each image. From Flicker, we collected $19,959$ images and $91,719$ related annotations, among which $2,600$ words are distinct. Based on the method described in Section 3, we estimated the co-occurrence matrix B between text features and image features. This co-occurrence matrix B was used by all the clustering tasks in our experiments.

For data preprocessing, we adopted the *bag-of-features* representation of images (Li and Perona, 2005) in our experiments. Interesting points were found in the images and described via the *SIFT descriptors* (Lowe, 2004). Then, the interesting points were clustered to generate a codebook to form an image feature space. The size of codebook was set to $2,000$ in our experiments. Based on the codebook, which serves as the image feature space, each image can be represented as a corresponding feature vector to be used in the next step.

To set our evaluation criterion, we used the

Data Set	KMeans		PLSA		STC	aPLSA
	separate	combined	separate	combined		
bi1	0.645±0.064	0.548±0.031	0.544±0.074	0.537±0.033	0.586±0.139	**0.482±0.062**
bi2	0.687±0.003	0.662±0.014	0.464±0.074	0.692±0.001	0.577±0.016	**0.455±0.096**
bi3	1.294±0.060	1.300±0.015	1.085±0.073	1.126±0.036	1.103±0.108	**1.029±0.074**
bi4	1.227±0.080	1.164±0.053	0.976±0.051	1.038±0.068	1.024±0.089	**0.919±0.065**
bi5	1.450±0.058	1.417±0.045	1.426±0.025	1.405±0.040	1.411±0.043	**1.377±0.040**
bi6	1.969±0.078	1.852±0.051	1.514±0.039	1.709±0.028	1.589±0.121	**1.503±0.030**
bi7	0.686±0.006	0.683±0.004	0.643±0.058	0.632±0.037	0.651±0.012	**0.624±0.066**
quad1	0.591±0.094	0.675±0.017	0.488±0.071	0.662±0.013	0.580±0.115	**0.432±0.085**
quad2	0.648±0.036	0.646±0.045	0.614±0.062	0.626±0.026	0.591±0.087	**0.515±0.098**
quint1	0.557±0.021	0.508±0.104	0.547±0.060	0.539±0.051	0.538±0.100	**0.502±0.067**
oct1	0.659±0.031	0.680±0.012	0.340±0.147	0.691±0.002	0.411±0.089	**0.306±0.101**
average	0.947±0.029	0.922±0.017	0.786±0.009	0.878±0.006	0.824±0.036	**0.741±0.018**

Table 2: Experimental result in term of entropy for all data sets and evaluation methods.

entropy to measure the quality of our clustering results. In information theory, entropy (Shannon, 1948) is a measure of the uncertainty associated with a random variable. In our problem, entropy serves as a measure of randomness of clustering result. The entropy of g on a single latent variable z is defined to be $H(g,z) \triangleq -\sum_{c\in\mathcal{C}} P(c|z)\log_2 P(c|z)$, where $\mathcal{C}$ is the class label set of $\mathcal{V}$ and $P(c|z) = \frac{|\{v|g(v)=z\wedge t(v)=c\}|}{|\{v|g(v)=z\}|}$, in which $t(v)$ is the *true* class label of image v. Lower entropy $H(g,\mathcal{Z})$ indicates less randomness and thus better clustering result.

4.2 Empirical Analysis

We now empirically analyze the effectiveness of our aPLSA algorithm. Because, to our best of knowledge, few existing methods addressed the problem of image clustering with the help of social annotation image data, we can only compare our aPLSA with several state-of-the-art clustering algorithms that are not directly designed for our problem. The first baseline is the well-known KMeans algorithm (MacQueen, 1967). Since our algorithm is designed based on PLSA (Hofmann, 1999), we also included PLSA for clustering as a baseline method in our experiments.

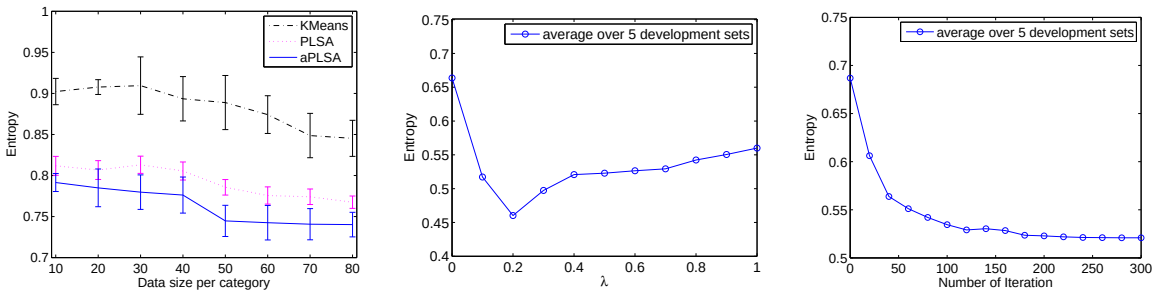
For each of the above two baselines, we have two strategies: (1) separated: the baseline method was applied on the target image data only; (2) combined: the baseline method was applied to cluster the combined data consisting of both target image data and the annotated image data. Clustering results on target image data were used for evaluation. Note that, in the combined data, all the annotations were thrown away since baseline methods evaluated in this paper do not leverage annotation information.

In addition, we compared our algorithm aPLSA to a state-of-the-art transfer clustering strategy, known as *self-taught clustering* (STC) (Dai et al., 2008b). STC makes use of auxiliary data to estimate a better feature representation to benefit the target clustering. In these experiments, the annotated image data were used as auxiliary data in STC, which does not use the annotation text.

In our experiments, the performance is in the form of the average entropy and variance of five repeats by randomly selecting 50 images from each of the categories. We selected only 50 images per category, since this paper is focused on clustering sparse data. Table 2 shows the performance with respect to all comparison methods on each of the image clustering tasks measured by the entropy criterion. From the tables, we can see that our algorithm aPLSA outperforms the baseline methods in all the data sets. We believe that is because aPLSA can effectively utilize the knowledge from the socially annotated image data. On average, aPLSA gives rise to 21.8% of entropy reduction and as compared to KMeans, 5.7% of entropy reduction as compared to PLSA, and 10.1% of entropy reduction as compared to STC.

4.2.1 Varying Data Size

We now show how the data size affects aPLSA, with two baseline methods KMeans and PLSA as reference. The experiments were conducted on different amounts of target image data, varying from 10 to 80. The corresponding experimental results in average entropy over all the 11 clustering tasks are shown in Figure 4(a). From this figure, we observe that aPLSA always yields a significant reduction in entropy as compared with two baseline methods KMeans and PLSA, regardless of the size of target image data that we used.



(a) (b) (c)

Figure 4: (a) The entropy curve as a function of different amounts of data per category. (b) The entropy curve as a function of different number of iterations. (c) The entropy curve as a function of different trade-off parameter λ.

4.2.2 Parameter Sensitivity

In aPLSA, there is a trade-off parameter λ that affects how the algorithm relies on auxiliary data. When $\lambda = 0$, the aPLSA relies only on annotated image data B. When $\lambda = 1$, aPLSA relies only on target image data A, in which case aPLSA degenerates to PLSA. Smaller λ indicates heavier reliance on the annotated image data. We have done some experiments on the development sets to investigate how different λ affect the performance of aPLSA. We set the number of images per category to 50, and tested the performance of aPLSA. The result in average entropy over all development sets is shown in Figure 4(b). In the experiments described in this paper, we set λ to 0.2, which is the best point in Figure 4(b).

4.2.3 Convergence

In our experiments, we tested the convergence property of our algorithm aPLSA as well. Figure 4(c) shows the average entropy curve given by aPLSA over all development sets. From this figure, we see that the entropy decreases very fast during the first 100 iterations and becomes stable after 150 iterations. We believe that 200 iterations is sufficient for aPLSA to converge.

5 Conclusions

In this paper, we proposed a new learning scenario called heterogeneous transfer learning and illustrated its application to image clustering. Image clustering, a vital component in organizing search results for query-based image search, was shown to be improved by transferring knowledge from unrelated images with annotations in a social Web. This is done by first learning the high-quality latent variables in the auxiliary data, and then transferring this knowledge to help improve the clustering of the target image data. We conducted experiments on two image data sets, using the Flickr data as the annotated auxiliary image data, and showed that our aPLSA algorithm can greatly outperform several state-of-the-art clustering algorithms.

In natural language processing, there are many future opportunities to apply heterogeneous transfer learning. In (Ling et al., 2008) we have shown how to classify the Chinese text using English text as the training data. We may also consider clustering, topic modeling, question answering, etc., to be done using data in different feature spaces. We can consider data in different modalities, such as video, image and audio, as the training data. Finally, we will explore the theoretical foundations and limitations of heterogeneous transfer learning as well.

Acknowledgement Qiang Yang thanks Hong Kong CERG grant 621307 for supporting the research.

References

Alina Andreevskaia and Sabine Bergler. 2008. When specialists and generalists work together: Overcoming domain dependence in sentiment tagging. In *ACL-08: HLT*, pages 290–298, Columbus, Ohio, June.

Andrew Arnold, Ramesh Nallapati, and William W. Cohen. 2007. A comparative study of methods for transductive transfer learning. In ICDM 2007 Workshop on Mining and Management of Biological Data, pages 77-82.

Andrew Arnold, Ramesh Nallapati, and William W. Cohen. 2008. Exploiting feature hierarchy for transfer learning in named entity recognition. In *ACL-08: HLT*.

Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. 2004. A probabilistic framework for semi-supervised clustering. In *ACM SIGKDD 2004*, pages 59–68.

John Blitzer, Ryan Mcdonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP 2006*, pages 120–128, Sydney, Australia.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL 2007*, pages 440–447, Prague, Czech Republic.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT 1998*, pages 92–100, New York, NY, USA. ACM.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Yee Seng Chan and Hwee Tou Ng. 2007. Domain adaptation with active learning for word sense disambiguation. In *ACL 2007*, Prague, Czech Republic.

David A. Cohn and Thomas Hofmann. 2000. The missing link - a probabilistic model of document content and hypertext connectivity. In *NIPS 2000*, pages 430–436.

Wenyuan Dai, Yuqiang Chen, Gui-Rong Xue, Qiang Yang, and Yong Yu. 2008a. Translated learning: Transfer learning across different feature spaces. In *NIPS 2008*, pages 353–360.

Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. 2008b. Self-taught clustering. In *ICML 2008*, pages 200–207. Omnipress.

Hal Daume, III. 2007. Frustratingly easy domain adaptation. In *ACL 2007*, pages 256–263, Prague, Czech Republic.

Jesse Davis and Pedro Domingos. 2008. Deep transfer via second-order markov logic. In *AAAI 2008 Workshop on Transfer Learning*, Chicago, USA.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. L, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, pages 391–407.

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

Thomas Finley and Thorsten Joachims. 2005. Supervised clustering with support vector machines. In *ICML 2005*, pages 217–224, New York, NY, USA. ACM.

G. Griffin, A. Holub, and P. Perona. 2007. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology.

Thomas Hofmann. 1999 Probabilistic latent semantic analysis. In Proc. of Uncertainty in Artificial Intelligence, UAI99. Pages 289–296

Thomas Hofmann. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*. volume 42, number 1-2, pages 177–196. Kluwer Academic Publishers.

Jing Jiang and Chengxiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In *ACL 2007*, pages 264–271, Prague, Czech Republic, June.

Leonard Kaufman and Peter J. Rousseeuw. 1990. *Finding groups in data: an introduction to cluster analysis*. John Wiley and Sons, New York.

Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR 2006*, pages 2169–2178, Washington, DC, USA.

Fei-Fei Li and Pietro Perona. 2005. A bayesian hierarchical model for learning natural scene categories. In *CVPR 2005*, pages 524–531, Washington, DC, USA.

Xiao Ling, Gui-Rong Xue, Wenyuan Dai, Yun Jiang, Qiang Yang, and Yong Yu. 2008. Can chinese web pages be classified with english data source? In *WWW 2008*, pages 969–978, New York, NY, USA. ACM.

Nicolas Loeff, Cecilia Ovesdotter Alm, and David A. Forsyth. 2006. Discriminating image senses by clustering with multimodal features. In *COLING/ACL 2006 Main conference poster sessions*, pages 547–554.

David G. Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV) 2004*, volume 60, number 2, pages 91–110.

J. B. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 1:281–297, Berkeley, CA, USA.

Kamal Nigam and Rayid Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, pages 86–93, New York, USA.

Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. 2007. Self-taught learning: transfer learning from unlabeled data. In *ICML 2007*, pages 759–766, New York, NY, USA. ACM.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *ACL 2007*.

Roi Reichart, Katrin Tomanek, Udo Hahn, and Ari Rappoport. 2008. Multi-task active learning for linguistic annotations. In *ACL-08: HLT*, pages 861–869.

C. E. Shannon. 1948. A mathematical theory of communication. *Bell system technical journal*, 27.

J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. 2005. Discovering object categories in image collections. In *ICCV 2005*.

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. 1999. In *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377.

Pengcheng Wu and Thomas G. Dietterich. 2004. Improving svm accuracy by training on auxiliary data sources. In *ICML 2004*, pages 110–117, New York, NY, USA.

Yejun Wu and Douglas W. Oard. 2008. Bilingual topic aspect classification with a few training examples. In *ACM SIGIR 2008*, pages 203–210, New York, NY, USA.

Xiaojin Zhu. 2007. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

Investigations on Word Senses and Word Usages

Katrin Erk
University of Texas at Austin
katrin.erk@mail.utexas.edu

Diana McCarthy
University of Sussex
dianam@sussex.ac.uk

Nicholas Gaylord
University of Texas at Austin
nlgaylord@mail.utexas.edu

Abstract

The vast majority of work on word senses has relied on predefined sense inventories and an annotation schema where each word instance is tagged with the best fitting sense. This paper examines the case for a graded notion of word meaning in two experiments, one which uses WordNet senses in a graded fashion, contrasted with the "winner takes all" annotation, and one which asks annotators to judge the similarity of two usages. We find that the graded responses correlate with annotations from previous datasets, but sense assignments are used in a way that weakens the case for clear cut sense boundaries. The responses from both experiments correlate with the overlap of paraphrases from the English lexical substitution task which bodes well for the use of substitutes as a proxy for word sense. This paper also provides two novel datasets which can be used for evaluating computational systems.

1 Introduction

The vast majority of work on word sense tagging has assumed that predefined word senses from a dictionary are an adequate proxy for the task, although of course there are issues with this enterprise both in terms of cognitive validity (Hanks, 2000; Kilgarriff, 1997; Kilgarriff, 2006) and adequacy for computational linguistics applications (Kilgarriff, 2006). Furthermore, given a predefined list of senses, annotation efforts and computational approaches to word sense disambiguation (WSD) have usually assumed that one best fitting sense should be selected for each usage. While there is usually some allowance made for multiple senses, this is typically not adopted by annotators or computational systems.

Research on the psychology of concepts (Murphy, 2002; Hampton, 2007) shows that categories in the human mind are not simply sets with clear-cut boundaries: Some items are perceived as more typical than others (Rosch, 1975; Rosch and Mervis, 1975), and there are borderline cases on which people disagree more often, and on whose categorization they are more likely to change their minds (Hampton, 1979; McCloskey and Glucksberg, 1978). Word meanings are certainly related to mental concepts (Murphy, 2002). This raises the question of whether there is any such thing as the one appropriate sense for a given occurrence.

In this paper we will explore using graded responses for sense tagging within a novel annotation paradigm. Modeling the annotation framework after psycholinguistic experiments, we do not train annotators to conform to sense distinctions; rather we assess individual differences by asking annotators to produce graded ratings instead of making a binary choice. We perform two annotation studies. In the first one, referred to as **WSsim** (Word Sense Similarity), annotators give graded ratings on the applicability of WordNet senses. In the second one, **Usim** (Usage Similarity), annotators rate the similarity of pairs of occurrences (usages) of a common target word. Both studies explore whether users make use of a graded scale or persist in making binary decisions even when there is the option for a graded response. The first study additionally tests to what extent the judgments on WordNet senses fall into clear-cut clusters, while the second study allows us to explore meaning similarity independently of any lexicon resource.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 10–18,
Suntec, Singapore, 2-7 August 2009.

2 Related Work

Manual word sense assignment is difficult for human annotators (Krishnamurthy and Nicholls, 2000). Reported inter-annotator agreement (ITA) for fine-grained word sense assignment tasks has ranged between 69% (Kilgarriff and Rosenzweig, 2000) for a lexical sample using the HECTOR dictionary and 78.6.% using WordNet (Landes et al., 1998) in all-words annotation. The use of more coarse-grained senses alleviates the problem: In OntoNotes (Hovy et al., 2006), an ITA of 90% is used as the criterion for the construction of coarse-grained sense distinctions. However, intriguingly, for some high-frequency lemmas such as *leave* this ITA threshold is not reached even after multiple re-partitionings of the semantic space (Chen and Palmer, 2009). Similarly, the performance of WSD systems clearly indicates that WSD is not easy unless one adopts a coarse-grained approach, and then systems tagging all words at best perform a few percentage points above the most frequent sense heuristic (Navigli et al., 2007). Good performance on coarse-grained sense distinctions may be more useful in applications than poor performance on fine-grained distinctions (Ide and Wilks, 2006) but we do not know this yet and there is some evidence to the contrary (Stokoe, 2005).

Rather than focus on the granularity of clusters, the approach we will take in this paper is to examine the phenomenon of word meaning both with and without recourse to predefined senses by focusing on the similarity of uses of a word. Human subjects show excellent agreement on judging word similarity out of context (Rubenstein and Goodenough, 1965; Miller and Charles, 1991), and human judgments have previously been used successfully to study synonymy and near-synonymy (Miller and Charles, 1991; Bybee and Eddington, 2006). We focus on polysemy rather than synonymy. Our aim will be to use WSsim to determine to what extent annotations form cohesive clusters. In principle, it should be possible to use existing sense-annotated data to explore this question: almost all sense annotation efforts have allowed annotators to assign multiple senses to a single occurrence, and the distribution of these sense labels should indicate whether annotators viewed the senses as disjoint or not. However, the percentage of markables that received multiple sense labels in existing corpora is small, and it varies massively between corpora: In the SemCor corpus (Landes et al., 1998), only 0.3% of all markables received multiple sense labels. In the SENSEVAL-3 English lexical task corpus (Mihalcea et al., 2004) (hereafter referred to as SE-3), the ratio is much higher at 8% of all markables[1]. This could mean annotators feel that there is usually a single applicable sense, or it could point to a bias towards single-sense assignment in the annotation guidelines and/or the annotation tool. The WSsim experiment that we report in this paper is designed to eliminate such bias as far as possible and we conduct it on data taken from SemCor and SE-3 so that we can compare the annotations. Although we use WordNet for the annotation, our study is not a study of WordNet per se. We choose WordNet because it is sufficiently fine-grained to examine subtle differences in usage, and because traditionally annotated datasets exist to which we can compare our results.

Predefined dictionaries and lexical resources are not the only possibilities for annotating lexical items with meaning. In cross-lingual settings, the actual translations of a word can be taken as the sense labels (Resnik and Yarowsky, 2000). Recently, McCarthy and Navigli (2007) proposed the English Lexical Substitution task (hereafter referred to as LEXSUB) under the auspices of SemEval-2007. It uses paraphrases for words in context as a way of annotating meaning. The task was proposed following a background of discussions in the WSD community as to the adequacy of predefined word senses. The LEXSUB dataset comprises open class words (nouns, verbs, adjectives and adverbs) with token instances of each word appearing in the context of one sentence taken from the English Internet Corpus (Sharoff, 2006). The methodology can only work where there are paraphrases, so the dataset only contains words with more than one meaning where at least two different meanings have near synonyms. For meanings without obvious substitutes the annotators were allowed to use multiword paraphrases or words with slightly more general meanings. This dataset has been used to evaluate automatic systems which can find substitutes appropriate for the context. To the best of our knowledge there has been no study of how the data collected relates to word sense annotations or judgments of semantic similarity. In this paper we examine these relation-

[1] This is even though both annotation efforts use balanced corpora, the Brown corpus in the case of SemCor, the British National Corpus for SE-3.

ships by re-using data from LEXSUB in both new annotation experiments and testing the results for correlation.

3 Annotation

We conducted two experiments through an online annotation interface. Three annotators participated in each experiment; all were native British English speakers. The first experiment, WSsim, collected annotator judgments about the applicability of dictionary senses using a 5-point rating scale. The second, Usim, also utilized a 5-point scale but collected judgments on the similarity in meaning between two uses of a word. [2] The scale was 1 – *completely different*, 2 – *mostly different*, 3 – *similar*, 4 – *very similar* and 5 – *identical*. In Usim, this scale rated the similarity of the two uses of the common target word; in WSsim it rated the similarity between the use of the target word and the sense description. In both experiments, the annotation interface allowed annotators to revisit and change previously supplied judgments, and a comment box was provided alongside each item.

WSsim. This experiment contained a total of 430 sentences spanning 11 lemmas (nouns, verbs and adjectives). For 8 of these lemmas, 50 sentences were included, 25 of them randomly sampled from SemCor [3] and 25 randomly sampled from SE-3.[4] The remaining 3 lemmas in the experiment each had 10 sentences taken from the LEXSUB data.

WSsim is a word sense annotation task using WordNet senses.[5] Unlike previous word sense annotation projects, we asked annotators to provide judgments on the applicability of *every* WordNet sense of the target lemma with the instruction: [6]

Your task is to rate, for each of these descriptions, how well they reflect the meaning of the boldfaced word in the sentence.

Applicability judgments were not binary, but were instead collected using the five-point scale given above which allowed annotators to indicate not only whether a given sense applied, but *to what degree*. Each annotator annotated each of the 430 items. By having multiple annotators per item and a graded, non-binary annotation scheme we allow for and measure differences between annotators, rather than training annotators to conform to a common sense distinction guideline. By asking annotators to provide ratings for each individual sense, we strive to eliminate all bias towards either single-sense or multiple-sense assignment. In traditional word sense annotation, such bias could be introduced directly through annotation guidelines or indirectly, through tools that make it easier to assign fewer senses. We focus not on finding the best fitting sense but collect judgments on the applicability of all senses.

Usim. This experiment used data from LEXSUB. For more information on LEXSUB, see McCarthy and Navigli (2007). 34 lemmas (nouns, verbs, adjectives and adverbs) were manually selected, including the 3 lemmas also used in WSsim. We selected lemmas which exhibited a range of meanings and substitutes in the LEXSUB data, with as few multiword substitutes as possible. Each lemma is the target in 10 LEXSUB sentences. For our experiment, we took every possible pairwise comparison of these 10 sentences for a lemma. We refer to each such pair of sentences as an SPAIR. The resulting dataset comprised 45 SPAIRs per lemma, adding up to 1530 comparisons per annotator overall.

In this annotation experiment, annotators saw SPAIRs with a common target word and rated the similarity in meaning between the two uses of the target word with the instruction:

Your task is to rate, for each pair of sentences, how similar in meaning the two boldfaced words are on a five-point scale.

In addition annotators had the ability to respond with "Cannot Decide", indicating that they were unable to make an effective comparison between the two contexts, for example because the meaning of one usage was unclear. This occurred in 9 paired occurrences during the course of annotation, and these items (paired occurrences) were

[2] Throughout this paper, a target word is assumed to be a word in a given PoS.

[3] The SemCor dataset was produced alongside WordNet, so it can be expected to support the WordNet sense distinctions. The same cannot be said for SE-3.

[4] Sentence fragments and sentences with 5 or fewer words were excluded from the sampling. Annotators were given the sentences, but not the original annotation from these resources.

[5] WordNet 1.7.1 was used in the annotation of both SE-3 and SemCor; we used the more current WordNet 3.0 after verifying that the lemmas included in this experiment had the same senses listed in both versions. Care was taken additionally to ensure that senses were not presented in an order that reflected their frequency of occurrence.

[6] The guidelines for both experiments are available at `http://comp.ling.utexas.edu/people/katrin_erk/graded_sense_and_usage_annotation`

excluded from further analysis.

The purpose of Usim was to collect judgments about degrees of similarity between a word's meaning in different contexts. Unlike WSsim, Usim does not rely upon any dictionary resource as a basis for the judgments.

4 Analyses

This section reports on analyses on the annotated data. In all the analyses we use Spearman's rank correlation coefficient (ρ), a nonparametric test, because the data does not seem to be normally distributed. We used two-tailed tests in all cases, rather than assume the direction of the relationship. As noted above, we have three annotators per task, and each annotator gave judgments for every sentence (WSsim) or sentence pair (Usim). Since the annotators may vary as to how they use the ordinal scale, we do not use the mean of judgments[7] but report all individual correlations. All analyses were done using the R package.[8]

4.1 WSsim analysis

In the WSsim experiment, annotators rated the applicability of each WordNet 3.0 sense for a given target word occurrence. Table 1 shows a sample annotation for the target *argument.n*. [9]

Pattern of annotation and annotator agreement. Figure 1 shows how often each of the five judgments on the scale was used, individually and summed over all annotators. (The y-axis shows raw counts of each judgment.) We can see from this figure that the extreme ratings 1 and 5 are used more often than the intermediate ones, but annotators make use of the full ordinal scale when judging the applicability of a sense. Also, the figure shows that annotator 1 used the extreme negative rating 1 much less than the other two annotators. Figure 2 shows the percentage of times each judgment was used on senses of three lemmas, *different.a*, *interest.n*, and *win.v*. In WordNet, they have 5, 7, and 4 senses, respectively. The pattern for *win.v* resembles the overall distribution of judgments, with peaks at the extreme ratings 1 and 5. The lemma *interest.n* has a single peak at rating 1, partly due to the fact that senses 5 (financial involvement) and 6 (interest group) were rarely judged to apply. For the lemma *different.a*, all judgments have been used with approximately the same frequency.

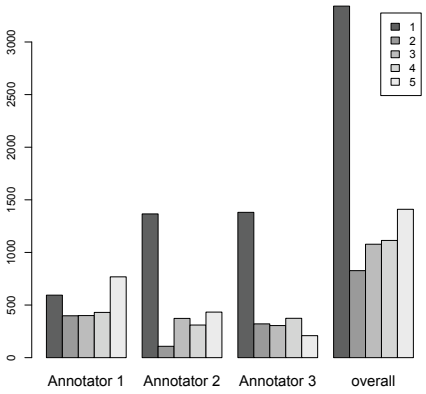


Figure 1: WSsim experiment: number of times each judgment was used, by annotator and summed over all annotators. The y-axis shows raw counts of each judgment.

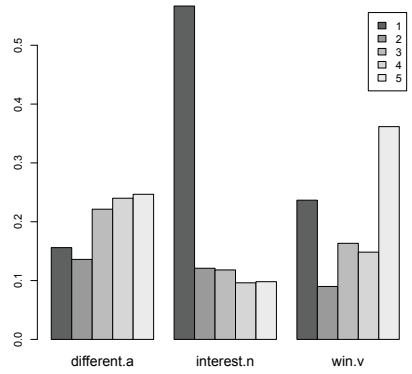


Figure 2: WSsim experiment: percentage of times each judgment was used for the lemmas *different.a*, *interest.n* and *win.v*. Judgment counts were summed over all three annotators.

We measured the level of agreement between annotators using Spearman's ρ between the judgments of every pair of annotators. The pairwise correlations were $\rho = 0.506$, $\rho = 0.466$ and $\rho = 0.540$, all highly significant with $p < 2.2\text{e-}16$.

Agreement with previous annotation in SemCor and SE-3. 200 of the items in WSsim had been previously annotated in SemCor, and 200 in SE-3. This lets us compare the annotation results across annotation efforts. Table 2 shows the percentage of items where more than one sense was assigned in the subset of WSsim from SemCor (first row), from SE-3 (second row), and

[7] We have also performed several of our calculations using the mean judgment, and they also gave highly significant results in all the cases we tested.

[8] http://www.r-project.org/

[9] We use *word*.PoS to denote a target word (lemma).

Sentence	Senses 1	2	3	4	5	6	7	Annotator
This question provoked **arguments** in America about the	1	4	4	2	1	1	3	Ann. 1
Norton Anthology of Literature by Women, some of the	4	5	4	2	1	1	4	Ann. 2
contents of which were said to have had little value as literature.	1	4	5	1	1	1	1	Ann. 3

Table 1: A sample annotation in the WSsim experiment. The senses are: 1:statement, 2:controversy, 3:debate, 4:literary argument, 5:parameter, 6:variable, 7:line of reasoning

Data	Orig.	WSsim judgment ≥ 3	≥ 4	5
WSsim/SemCor	0.0	80.2	57.5	28.3
WSsim/SE-3	24.0	78.0	58.3	27.1
All WSsim		78.8	57.4	27.7

Table 2: Percentage of items with multiple senses assigned. *Orig*: in the original SemCor/SE-3 data. WS*sim judgment*: items with judgments at or above the specified threshold. The percentages for WSsim are averaged over the three annotators.

	$p < 0.05$ pos	neg	$p < 0.01$ pos	neg
Ann. 1	30.8	11.4	23.2	5.9
Ann. 2	22.2	24.1	19.6	19.6
Ann. 3	12.7	12.0	10.0	6.0

Table 3: Percentage of sense pairs that were significantly positively (pos) or negatively (neg) correlated at $p < 0.05$ and $p < 0.01$, shown by annotator.

	$j \geq 3$	$j \geq 4$	$j = 5$
Ann. 1	71.9	49.1	8.1
Ann. 2	55.3	24.7	8.1
Ann. 3	42.8	24.0	4.9

Table 4: Percentage of sentences in which at least two uncorrelated ($p > 0.05$) or negatively correlated senses have been annotated with judgments at the specified threshold.

all of WSsim (third row). The *Orig.* column indicates how many items had multiple labels in the original annotation (SemCor or SE-3) [10]. Note that no item had more than one sense label in SemCor. The columns under WS*sim judgment* show the percentage of items (averaged over the three annotators) that had judgments at or above the specified threshold, starting from rating 3 – *similar*. Within WSsim, the percentage of multiple assignments in the three rows is fairly constant. WSsim avoids the bias to one sense by deliberately asking for judgments on the applicability of each sense rather than asking annotators to find the best one.

To compute the Spearman's correlation between the original sense labels and those given in the WSsim annotation, we converted SemCor and SE-3 labels to the format used within WSsim: Assigned senses were converted to a judgment of 5, and unassigned senses to a judgment of 1. For the WSsim/SemCor dataset, the correlation between original and WSsim annotation was $\rho = 0.234$, $\rho = 0.448$, and $\rho = 0.390$ for the three annotators, each highly significant with $p <$ 2.2e-16. For the WSsim/SE-3 dataset, the correlations were $\rho = 0.346$, $\rho = 0.449$ and $\rho = 0.338$, each of them again highly significant at $p <$ 2.2e-16.

Degree of sense grouping. Next we test to what extent the sense applicability judgments in the WSsim task could be explained by more coarse-grained, categorial sense assignments. We first test how many pairs of senses for a given lemma show similar patterns in the ratings that they receive. Table 3 shows the percentage of sense pairs that were significantly correlated for each annotator.[11] Significantly positively correlated senses can possibly be reduced to more coarse-grained senses. Would annotators have been able to designate a single appropriate sense given these more coarse-grained senses? Call two senses *groupable* if they are significantly positively correlated; in order not to overlook correlations that are relatively weak but existent, we use a cutoff of $p = 0.05$ for significant correlation. We tested how often annotators gave ratings of at least *similar*, i.e. ratings ≥ 3, to senses that were *not* groupable. Table 4 shows the percentages of items where at least two non-groupable senses received ratings at or above the specified threshold. The table shows that regardless of which annotator we look at, over 40% of all items had two or more non-groupable senses receive judgments of at least 3 (*similar*). There

[10] Overall, 0.3% of tokens in SemCor have multiple labels, and 8% of tokens in SE-3, so the multiple label assignment in our sample is not an underestimate.

[11] We exclude senses that received a uniform rating of 1 on all items. This concerned 4 senses for annotator 2 and 6 for annotator 3.

1) We study the methods and concepts that each writer uses to defend the cogency of legal, deliberative, or more generally political prudence against explicit or implicit charges that practical thinking is merely a knack or form of cleverness.

2) Eleven CIRA members have been convicted of criminal charges and others are awaiting trial.

Figure 3: An SPAIR for *charge.n*. Annotator judgments: 2,3,4

were even several items where two or more non-groupable senses each got a judgment of 5. The sentence in table 1 is a case where several non-groupable senses got ratings ≥ 3. This is most pronounced for Annotator 2, who along with sense 2 (controversy) assigned senses 1 (statement), 7 (line of reasoning), and 3 (debate), none of which are groupable with sense 2.

4.2 Usim analysis

In this experiment, ratings between 1 and 5 were given for every pairwise combination of sentences for each target lemma. An example of an SPAIR for *charge.n* is shown in figure 3. In this case the verdicts from the annotators were 2, 3 and 4.

Pattern of Annotations and Annotator Agreement Figure 4 gives a bar chart of the judgments for each annotator and summed over annotators. We can see from this figure that the annotators use the full ordinal scale when judging the similarity of a word's usages, rather than sticking to the extremes. There is variation across words, depending on the relatedness of each word's usages. Figure 5 shows the judgments for the words *bar.n, work.v* and *raw.a*. We see that *bar.n* has predominantly different usages with a peak for category 1, *work.v* has more similar judgments (category 5) compared to any other category and *raw.a* has a peak in the middle category (3). [12] There are other words, like for example *fresh.a*, where the spread is more uniform.

To gauge the level of agreement between annotators, we calculated Spearman's ρ between the judgments of every pair of annotators as in section 4.1. The pairwise correlations are all highly significant ($p < 2.2e\text{-}16$) with Spearman's $\rho =$ 0.502, 0.641 and 0.501 giving an average correlation of 0.548. We also perform leave-one-out resampling following Lapata (2006) which gave us a Spearman's correlation of 0.630.

[12] For figure 5 we sum the judgments over annotators.

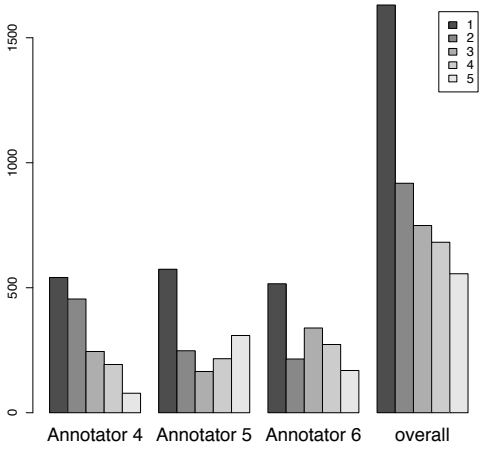


Figure 4: Usim experiment: number of times each judgment was used, by annotator and summed over all annotators

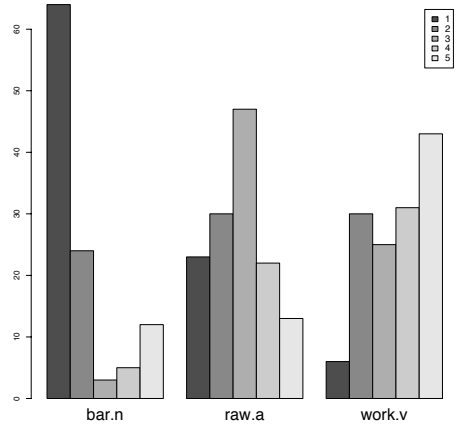


Figure 5: Usim experiment: number of times each judgment was used for *bar.n, work.v* and *raw.a*

Comparison with LEXSUB substitutions Next we look at whether the Usim judgments on sentence pairs (SPAIRs) correlate with LEXSUB substitutes. To do this we use the overlap of substitutes provided by the five LEXSUB annotators between two sentences in an SPAIR. In LEXSUB the annotators had to replace each item (a target word within the context of a sentence) with a substitute that fitted the context. Each annotator was permitted to supply up to three substitutes provided that they all fitted the context equally. There were 10 sentences per lemma. For our analyses we take every SPAIR for a given lemma and calculate the overlap (*inter*) of the substitutes provided by the annotators for the two usages under scrutiny. Let s_1 and s_2 be a pair of sentences in an SPAIR and

x_1 and x_2 be the multisets of substitutes for the respective sentences. Let $freq(w,x)$ be the frequency of a substitute w in a multiset x of substitutes for a given sentence. [13] $\text{INTER}(s_1,s_2) =$

$$\frac{\sum_{w \in x_1 \cap x_2} min(freq(w,x_1), freq(w,x_2))}{max(|x_1|,|x_2|)}$$

Using this calculation for each SPAIR we can now compute the correlation between the Usim judgments for each annotator and the INTER values, again using Spearman's. The figures are shown in the leftmost block of table 5. The average correlation for the 3 annotators was 0.488 and the p-values were all $<$ 2.2e-16. This shows a highly significant correlation of the Usim judgments and the overlap of substitutes.

We also compare the WSsim judgments against the LEXSUB substitutes, again using the INTER measure of substitute overlap. For this analysis, we only use those WSsim sentences that are originally from LEXSUB. In WSsim, the judgments for a sentence comprise judgments for each WordNet sense of that sentence. In order to compare against INTER, we need to transform these sentence-wise ratings in WSsim to a WSsim-based judgment of sentence similarity. To this end, we compute the Euclidean Distance[14] (ED) between two vectors J_1 and J_2 of judgments for two sentences s_1, s_2 for the same lemma ℓ. Each of the n indexes of the vector represent one of the n different WordNet senses for ℓ. The value at entry i of the vector J_1 is the judgment that the annotator in question (we do not average over annotators here) provided for sense i of ℓ for sentence s_1.

$$ED(J_1,J_2) = \sqrt{(\sum_{i=1}^{n}(J_1[i]-J_2[i])^2)} \qquad (1)$$

We correlate the Euclidean distances with INTER. We can only test correlation for the subset of WSsim that overlaps with the LEXSUB data: the 30 sentences for *investigator.n, function.n* and *order.v*, which together give 135 unique SPAIRs. We refer to this subset as W∩U. The results are given in the third block of table 5. Note that since we are measuring *distance* between SPAIRs for WSsim whereas INTER is a measure of *similarity*, the correlation is negative. The results are highly significant with individual p-values from $<$ 1.067e-10 to $<$ 1.551e-08 and a mean correlation of -0.495. The results in the first and third block of table 5 are not directly comparable, as the results in the first block are for all Usim data and not the subset of LEXSUB with WSsim annotations. We therefore repeated the analysis for Usim on the subset of data in WSsim and provide the correlation in the middle section of table 5. The mean correlation for Usim on this subset of the data is 0.532, which is a stronger relationship compared to WSsim, although there is more discrepancy between individual annotators, with the result for annotator 4 giving a p-value = 9.139e-05 while the other two annotators had p-values $<$ 2.2e-16.

Usim All		Usim W∩U	WSsim W∩U	
ann.	ρ	ρ	ann.	ρ
4	0.383	0.330	1	-0.520
5	0.498	0.635	2	-0.503
6	0.584	0.631	3	-0.463

Table 5: Annotator correlation with LEXSUB substitute overlap (*inter*)

The LEXSUB substitute overlaps between different usages correlate well with both Usim and WSsim judgments, with a slightly stronger relationship to Usim, perhaps due to the more complicated representation of word meaning in WSsim which uses the full set of WordNet senses.

4.3 Correlation between WSsim and Usim

As we showed in section 4.1, WSsim correlates with previous word sense annotations in SemCor and SE-3 while allowing the user a more graded response to sense tagging. As we saw in section 4.2, Usim and WSsim judgments both have a highly significant correlation with similarity of usages as measured using the overlap of substitutes from LEXSUB. Here, we look at the correlation of WSsim and Usim, considering again the subset of data that is common to both experiments. We again transform WSsim sense judgments for individual sentences to distances between SPAIRs using Euclidean Distance (ED). The Spearman's ρ range between -0.307 and -0.671, and all results are highly significant with p-values between 0.0003 and $<$ 2.2e-16. As above, the correlation is negative because ED is a distance measure between sentences in an SPAIR, whereas the judg-

[13] The frequency of a substitute in a multiset depends on the number of LEXSUB annotators that picked the substitute for this item.

[14] We use Euclidean Distance rather than a normalizing measure like Cosine because a sentence where all ratings are 5 should be very different from a sentence where all senses received a rating of 1.

ments for Usim are similarity judgments. We see that there is highly significant correlation for every pairing of annotators from the two experiments.

5 Discussion

Validity of annotation scheme. Annotator ratings show highly significant correlation on both tasks. This shows that the tasks are well-defined. In addition, there is a strong correlation between WSsim and Usim, which indicates that the potential bias introduced by the use of dictionary senses in WSsim is not too prominent. However, we note that WSsim only contained a small portion of 3 lemmas (30 sentences and 135 SPAIRs) in common with Usim, so more annotation is needed to be certain of this relationship. Given the differences between annotator 1 and the other annotators in Fig. 1, it would be interesting to collect judgments for additional annotators.

Graded judgments of use similarity and sense applicability. The annotators made use of the full spectrum of ratings, as shown in Figures 1 and 4. This may be because of a graded perception of the similarity of uses as well as senses, or because some uses and senses are very similar. Table 4 shows that for a large number of WSsim items, multiple senses that were not significantly positively correlated got high ratings. This seems to indicate that the ratings we obtained cannot simply be explained by more coarse-grained senses. It may hence be reasonable to pursue computational models of word meaning that are graded, maybe even models that do not rely on dictionary senses at all (Erk and Pado, 2008).

Comparison to previous word sense annotation. Our graded WSsim annotations do correlate with traditional "best fitting sense" annotations from SemCor and SE-3; however, if annotators perceive similarity between uses and senses as graded, traditional word sense annotation runs the risk of introducing bias into the annotation.

Comparison to lexical substitutions. There is a strong correlation between both Usim and WSsim and the overlap in paraphrases that annotators generated for LEXSUB. This is very encouraging, and especially interesting because LEXSUB annotators freely generated paraphrases rather than selecting them from a list.

6 Conclusions

We have introduced a novel annotation paradigm for word sense annotation that allows for graded judgments and for some variation between annotators. We have used this annotation paradigm in two experiments, WSsim and Usim, that shed some light on the question of whether differences between word usages are perceived as categorial or graded. Both datasets will be made publicly available. There was a high correlation between annotator judgments within and across tasks, as well as with previous word sense annotation and with paraphrases proposed in the English Lexical Substitution task. Annotators made ample use of graded judgments in a way that cannot be explained through more coarse-grained senses. These results suggest that it may make sense to evaluate WSD systems on a task of graded rather than categorial meaning characterization, either through dictionary senses or similarity between uses. In that case, it would be useful to have more extensive datasets with graded annotation, even though this annotation paradigm is more time consuming and thus more expensive than traditional word sense annotation.

As a next step, we will automatically cluster the judgments we obtained in the WSsim and Usim experiments to further explore the degree to which the annotation gives rise to sense grouping. We will also use the ratings in both experiments to evaluate automatically induced models of word meaning. The SemEval-2007 word sense induction task (Agirre and Soroa, 2007) already allows for evaluation of automatic sense induction systems, but compares output to gold-standard senses from OntoNotes. We hope that the Usim dataset will be particularly useful for evaluating methods which relate usages without necessarily producing hard clusters. Also, we will extend the current dataset using more annotators and exploring additional lexicon resources.

Acknowledgments. We acknowledge support from the UK Royal Society for a Dorothy Hodkin Fellowship to the second author. We thank Sebastian Pado for many helpful discussions, and Andrew Young for help with the interface.

References

E. Agirre and A. Soroa. 2007. SemEval-2007 task 2: Evaluating word sense induction and dis-

crimination systems. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, pages 7–12, Prague, Czech Republic.

J. Bybee and D. Eddington. 2006. A usage-based approach to Spanish verbs of 'becoming'. *Language*, 82(2):323–355.

J. Chen and M. Palmer. 2009. Improving English verb sense disambiguation performance with linguistically motivated features and clear sense distinction boundaries. *Journal of Language Resources and Evaluation, Special Issue on SemEval-2007*. in press.

K. Erk and S. Pado. 2008. A structured vector space model for word meaning in context. In *Proceedings of EMNLP-08*, Waikiki, Hawaii.

J. A. Hampton. 1979. Polymorphous concepts in semantic memory. *Journal of Verbal Learning and Verbal Behavior*, 18:441–461.

J. A. Hampton. 2007. Typicality, graded membership, and vagueness. *Cognitive Science*, 31:355–384.

P. Hanks. 2000. Do word meanings exist? *Computers and the Humanities*, 34(1-2):205–215(11).

E. H. Hovy, M. Marcus, M. Palmer, S. Pradhan, L. Ramshaw, and R. Weischedel. 2006. OntoNotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (NAACL-2006)*, pages 57–60, New York.

N. Ide and Y. Wilks. 2006. Making sense about sense. In E. Agirre and P. Edmonds, editors, *Word Sense Disambiguation, Algorithms and Applications*, pages 47–73. Springer.

A. Kilgarriff and J. Rosenzweig. 2000. Framework and results for English Senseval. *Computers and the Humanities*, 34(1-2):15–48.

A. Kilgarriff. 1997. I don't believe in word senses. *Computers and the Humanities*, 31(2):91–113.

A. Kilgarriff. 2006. Word senses. In E. Agirre and P. Edmonds, editors, *Word Sense Disambiguation, Algorithms and Applications*, pages 29–46. Springer.

R. Krishnamurthy and D. Nicholls. 2000. Peeling an onion: the lexicographers' experience of manual sense-tagging. *Computers and the Humanities*, 34(1-2).

S. Landes, C. Leacock, and R. Tengi. 1998. Building semantic concordances. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, MA.

M. Lapata. 2006. Automatic evaluation of information ordering. *Computational Linguistics*, 32(4):471–484.

D. McCarthy and R. Navigli. 2007. SemEval-2007 task 10: English lexical substitution task. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, pages 48–53, Prague, Czech Republic.

M. McCloskey and S. Glucksberg. 1978. Natural categories: Well defined or fuzzy sets? *Memory & Cognition*, 6:462–472.

R. Mihalcea, T. Chklovski, and A. Kilgarriff. 2004. The Senseval-3 English lexical sample task. In *3rd International Workshop on Semantic Evaluations (SensEval-3) at ACL-2004*, Barcelona, Spain.

G. Miller and W. Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

G. L. Murphy. 2002. *The Big Book of Concepts*. MIT Press.

R. Navigli, K. C. Litkowski, and O. Hargraves. 2007. SemEval-2007 task 7: Coarse-grained English all-words task. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, pages 30–35, Prague, Czech Republic.

P. Resnik and D. Yarowsky. 2000. Distinguishing systems and distinguishing senses: New evaluation methods for word sense disambiguation. *Natural Language Engineering*, 5(3):113–133.

E. Rosch and C. B. Mervis. 1975. Family resemblance: Studies in the internal structure of categories. *Cognitive Psychology*, 7:573–605.

E. Rosch. 1975. Cognitive representations of semantic categories. *Journal of Experimental Psychology: General*, 104:192–233.

H. Rubenstein and J. Goodenough. 1965. Contextual correlates of synonymy. *Computational Linguistics*, 8:627–633.

S. Sharoff. 2006. Open-source corpora: Using the net to fish for linguistic data. *International Journal of Corpus Linguistics*, 11(4):435–462.

C. Stokoe. 2005. Differentiating homonymy and polysemy in information retrieval. In *Proceedings of HLT/EMNLP-05*, pages 403–410, Vancouver, B.C., Canada.

A Comparative Study on Generalization of Semantic Roles in FrameNet

Yuichiroh Matsubayashi[†] **Naoaki Okazaki**[†] **Jun'ichi Tsujii**[†‡*]
[†]Department of Computer Science, University of Tokyo, Japan
[‡]School of Computer Science, University of Manchester, UK
[*]National Centre for Text Mining, UK
{y-matsu,okazaki,tsujii}@is.s.u-tokyo.ac.jp

Abstract

A number of studies have presented machine-learning approaches to semantic role labeling with availability of corpora such as FrameNet and PropBank. These corpora define the semantic roles of predicates for each frame independently. Thus, it is crucial for the machine-learning approach to generalize semantic roles across different frames, and to increase the size of training instances. This paper explores several criteria for generalizing semantic roles in FrameNet: role hierarchy, human-understandable descriptors of roles, semantic types of filler phrases, and mappings from FrameNet roles to thematic roles of VerbNet. We also propose feature functions that naturally combine and weight these criteria, based on the training data. The experimental result of the role classification shows 19.16% and 7.42% improvements in error reduction rate and macro-averaged F1 score, respectively. We also provide in-depth analyses of the proposed criteria.

1 Introduction

Semantic Role Labeling (SRL) is a task of analyzing predicate-argument structures in texts. More specifically, SRL identifies predicates and their arguments with appropriate semantic roles. Resolving surface divergence of texts (e.g., voice of verbs and nominalizations) into unified semantic representations, SRL has attracted much attention from researchers into various NLP applications including question answering (Narayanan and Harabagiu, 2004; Shen and Lapata, 2007; Moschitti et al., 2007), and information extraction (Surdeanu et al., 2003).

buy.v	PropBank	FrameNet
Frame	buy.01	Commerce_buy
Roles	ARG0: buyer	Buyer
	ARG1: thing bought	Goods
	ARG2: seller	Seller
	ARG3: paid	Money
	ARG4: benefactive	Recipient
	...	...

Figure 1: A comparison of frames for *buy.v* defined in PropBank and FrameNet

In recent years, with the wide availability of corpora such as PropBank (Palmer et al., 2005) and FrameNet (Baker et al., 1998), a number of studies have presented statistical approaches to SRL (Màrquez et al., 2008). Figure 1 shows an example of the frame definitions for a verb *buy* in PropBank and FrameNet. These corpora define a large number of frames and define the semantic roles for each frame independently. This fact is problematic in terms of the performance of the machine-learning approach, because these definitions produce many roles that have few training instances.

PropBank defines a frame for each sense of predicates (e.g., *buy.01*), and semantic roles are defined in a frame-specific manner (e.g., *buyer* and *seller* for *buy.01*). In addition, these roles are associated with tags such as *ARG0-5* and *AM-**, which are commonly used in different frames. Most SRL studies on PropBank have used these tags in order to gather a sufficient amount of training data, and to generalize semantic-role classifiers across different frames. However, Yi et al. (2007) reported that tags *ARG2–ARG5* were inconsistent and not that suitable as training instances. Some recent studies have addressed alternative approaches to generalizing semantic roles across different frames (Gordon and Swanson, 2007; Zapi-

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 19–27,
Suntec, Singapore, 2-7 August 2009.

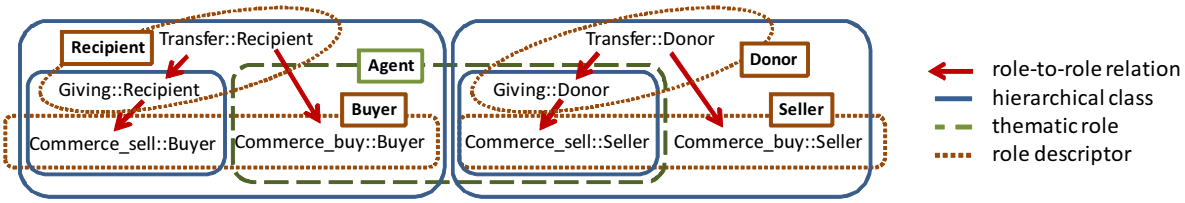


Figure 2: An example of role groupings using different criteria.

rain et al., 2008).

FrameNet designs semantic roles as frame specific, but also defines hierarchical relations of semantic roles among frames. Figure 2 illustrates an excerpt of the role hierarchy in FrameNet; this figure indicates that the *Buyer* role for the *Commerce_buy* frame (Commerce_buy::Buyer hereafter) and the Commerce_sell::Buyer role are inherited from the Transfer::Recipient role. Although the role hierarchy was expected to generalize semantic roles, no positive results for role classification have been reported (Baldewein et al., 2004). Therefore, the generalization of semantic roles across different frames has been brought up as a critical issue for FrameNet (Gildea and Jurafsky, 2002; Shi and Mihalcea, 2005; Giuglea and Moschitti, 2006)

In this paper, we explore several criteria for generalizing semantic roles in FrameNet. In addition to the FrameNet hierarchy, we use various pieces of information: human-understandable descriptors of roles, semantic types of filler phrases, and mappings from FrameNet roles to the thematic roles of VerbNet. We also propose feature functions that naturally combines these criteria in a machine-learning framework. Using the proposed method, the experimental result of the role classification shows 19.16% and 7.42% improvements in error reduction rate and macro-averaged F1, respectively. We provide in-depth analyses with respect to these criteria, and state our conclusions.

2 Related Work

Moschitti et al. (2005) first classified roles by using four coarse-grained classes (Core Roles, Adjuncts, Continuation Arguments and Co-referring Arguments), and built a classifier for each coarse-grained class to tag PropBank *ARG* tags. Even though the initial classifiers could perform rough estimations of semantic roles, this step was not able to solve the ambiguity problem in PropBank *ARG2-5*. When training a classifier for a semantic role, Baldewein et al. (2004) re-used the training instances of other roles that were similar to the target role. As similarity measures, they used the FrameNet hierarchy, peripheral roles of FrameNet, and clusters constructed by a EM-based method. Gordon and Swanson (2007) proposed a generalization method for the PropBank roles based on syntactic similarity in frames.

Many previous studies assumed that thematic roles bridged semantic roles in different frames. Gildea and Jurafsky (2002) showed that classification accuracy was improved by manually replacing FrameNet roles into 18 thematic roles. Shi and Mihalcea (2005) and Giuglea and Moschitti (2006) employed VerbNet thematic roles as the target of mappings from the roles defined by the different semantic corpora. Using the thematic roles as alternatives of *ARG* tags, Loper et al. (2007) and Yi et al. (2007) demonstrated that the classification accuracy of PropBank roles was improved for *ARG2* roles, but that it was diminished for *ARG1*. Yi et al. (2007) also described that *ARG2–5* were mapped to a variety of thematic roles. Zapirain et al. (2008) evaluated PropBank ARG tags and VerbNet thematic roles in a state-of-the-art SRL system, and concluded that PropBank *ARG* tags achieved a more robust generalization of the roles than did VerbNet thematic roles.

3 Role Classification

SRL is a complex task wherein several problems are intertwined: *frame-evoking word identification*, *frame disambiguation* (selecting a correct frame from candidates for the evoking word), *role-phrase identification* (identifying phrases that fill semantic roles), and *role classification* (assigning correct roles to the phrases). In this paper, we focus on role classification, in which the role generalization is particularly critical to the machine learning approach.

In the role classification task, we are given a sentence, a frame evoking word, a frame, and

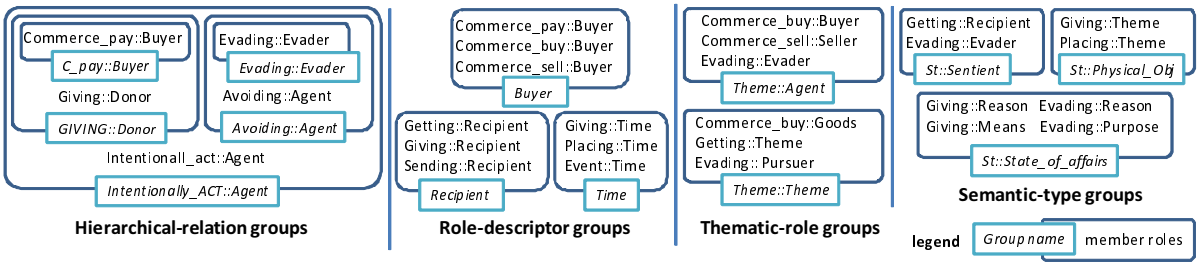


Figure 4: Examples for each type of role group.

INPUT:
frame = Commerce_sell
candidate roles = {Seller, Buyer, Goods, Reason, Time, ... , Place}
sentence = Can't [you] [**sell** $_{\text{Commerce\_sell}}$] [the factory] [to some other company] ?

OUTPUT:
sentence = Can't [you $_{\text{Seller}}$] [**sell** $_{\text{Commerce\_sell}}$] [the factory $_{\text{Goods}}$] [to some other company $_{\text{Buyer}}$] ?

Figure 3: An example of input and output of role classification.

phrases that take semantic roles. We are interested in choosing the correct role from the candidate roles for each phrase in the frame. Figure 3 shows a concrete example of input and output; the semantic roles for the phrases are chosen from the candidate roles: Seller, Buyer, Goods, Reason, ... , and Place.

4 Design of Role Groups

We formalize the generalization of semantic roles as the act of grouping several roles into a class. We define a *role group* as a set of role labels grouped by a criterion. Figure 4 shows examples of role groups; a group *Giving::Donor* (in the hierarchical-relation groups) contains the roles Giving::Donor and Commerce_pay::Buyer. The remainder of this section describes the grouping criteria in detail.

4.1 Hierarchical relations among roles

FrameNet defines hierarchical relations among frames (frame-to-frame relations). Each relation is assigned one of the seven types of directional relationships (*Inheritance*, *Using*, *Perspective_on*, *Causative_of*, *Inchoative_of*, *Subframe*, and *Precedes*). Some roles in two related frames are also connected with role-to-role relations. We assume that this hierarchy is a promising resource for generalizing the semantic roles; the idea is that the role at a node in the hierarchy inherits the characteristics of the roles of its ancestor nodes. For example, Commerce_sell::Seller in Figure 2 inherits the property of Giving::Donor.

For *Inheritance*, *Using*, *Perspective_on*, and *Subframe* relations, we assume that descendant roles in these relations have the same or specialized properties of their ancestors. Hence, for each role y_i, we define the following two role groups,

$$H^{\text{child}}_{y_i} = \{y | y = y_i \vee y \text{ is a child of } y_i\},$$
$$H^{\text{desc}}_{y_i} = \{y | y = y_i \vee y \text{ is a descendant of } y_i\}.$$

The hierarchical-relation groups in Figure 4 are the illustrations of $H^{\text{desc}}_{y_i}$.

For the relation types *Inchoative_of* and *Causative_of*, we define role groups in the opposite direction of the hierarchy,

$$H^{\text{parent}}_{y_i} = \{y | y = y_i \vee y \text{ is a parent of } y_i\},$$
$$H^{\text{ance}}_{y_i} = \{y | y = y_i \vee y \text{ is an ancestor of } y_i\}.$$

This is because lower roles of *Inchoative_of* and *Causative_of* relations represent more neutral stances or consequential states; for example, Killing::Victim is a parent of Death::Protagonist in the *Causative_of* relation.

Finally, the *Precedes* relation describes the sequence of states and events, but does not specify the direction of semantic inclusion relations. Therefore, we simply try $H^{\text{child}}_{y_i}$, $H^{\text{desc}}_{y_i}$, $H^{\text{parent}}_{y_i}$, and $H^{\text{ance}}_{y_i}$ for this relation type.

4.2 Human-understandable role descriptor

FrameNet defines each role as frame-specific; in other words, the same identifier does not appear in different frames. However, in FrameNet, human experts assign a human-understandable name to each role in a rather systematic manner. Some names are shared by the roles in different frames, whose identifiers are different. Therefore, we examine the semantic

commonality of these names; we construct an equivalence class of the roles sharing the same name. We call these human-understandable names *role descriptors*. In Figure 4, the role-descriptor group *Buyer* collects the roles Commerce_pay::Buyer, Commerce_buy::Buyer, and Commerce_sell::Buyer.

This criterion may be effective in collecting similar roles since the descriptors have been annotated by intuition of human experts. As illustrated in Figure 2, the role descriptors group the semantic roles which are similar to the roles that the FrameNet hierarchy connects as sister or parent-child relations. However, role-descriptor groups cannot express the relations between the roles as inclusions since they are equivalence classes. For example, the roles Commerce_sell::Buyer and Commerce_buy::Buyer are included in the role descriptor group *Buyer* in Figure 2; however, it is difficult to merge Giving::Recipient and Commerce_sell::Buyer because the Commerce_sell::Buyer has the extra property that one gives something of value in exchange and a human assigns different descriptors to them. We expect that the most effective weighting of these two criteria will be determined from the training data.

4.3 Semantic type of phrases

We consider that the selectional restriction is helpful in detecting the semantic roles. FrameNet provides information concerning the semantic types of role phrases (fillers); phrases that play specific roles in a sentence should fulfill the semantic constraint from this information. For instance, FrameNet specifies the constraint that Self_motion::Area should be filled by phrases whose semantic type is *Location*. Since these types suggest a coarse-grained categorization of semantic roles, we construct role groups that contain roles whose semantic types are identical.

4.4 Thematic roles of VerbNet

VerbNet thematic roles are 23 frame-independent semantic categories for arguments of verbs, such as *Agent*, *Patient*, *Theme* and *Source*. These categories have been used as consistent labels across verbs. We use a partial mapping between FrameNet roles and VerbNet thematic roles provided by SemLink. [1] Each group is constructed as a set $T_{t_i} = \{y|\text{SemLink maps } y \text{ into the thematic role } t_i\}$.

SemLink currently maps 1,726 FrameNet roles into VerbNet thematic roles, which are 37.61% of roles appearing at least once in the FrameNet corpus. This may diminish the effect of thematic-role groups than its potential.

5 Role classification method

5.1 Traditional approach

We are given a frame-evoking word e, a frame f and a role phrase x detected by a human or some automatic process in a sentence s. Let Y_f be the set of semantic roles that FrameNet defines as being possible role assignments for the frame f, and let $\mathbf{x} = \{x_1, \ldots, x_n\}$ be observed features for x from s, e and f. The task of semantic role classification can be formalized as the problem of choosing the most suitable role $\tilde{y}$ from Y_f. Suppose we have a model $P(y|f, \mathbf{x})$ which yields the conditional probability of the semantic role y for given f and $\mathbf{x}$. Then we can choose $\tilde{y}$ as follows:

$$\tilde{y} = \underset{y \in Y_f}{\operatorname{argmax}} P(y|f, \mathbf{x}). \quad (1)$$

A traditional way to incorporate role groups into this formalization is to overwrite each role y in the training and test data with its role group $m(y)$ according to the memberships of the group. For example, semantic roles Commerce_sell::Seller and Giving::Donor can be replaced by their thematic-role group *Theme::Agent* in this approach. We determine the most suitable role group $\tilde{c}$ as follows:

$$\tilde{c} = \underset{c \in \{m(y)|y \in Y_f\}}{\operatorname{argmax}} P_m(c|f, \mathbf{x}). \quad (2)$$

Here, $P_m(c|f, \mathbf{x})$ presents the probability of the role group c for f and $\mathbf{x}$. The role $\tilde{y}$ is determined uniquely iff a single role $y \in Y_f$ is associated with $\tilde{c}$. Some previous studies have employed this idea to remedy the data sparseness problem in the training data (Gildea and Jurafsky, 2002). However, we cannot apply this approach when multiple roles in Y_f are contained in the same class. For example, we can construct a semantic-type group *St::State_of_affairs* in which Giving::Reason and Giving::Means are included, as illustrated in Figure 4. If $\tilde{c}$ = *St::State_of_affairs*, we cannot disambiguate which original role is correct. In addition, it may be more effective to use various

[1] http://verbs.colorado.edu/semlink/

groupings of roles together in the model. For instance, the model could predict the correct role Commerce_sell::Seller for the phrase "you" in Figure 3 more confidently, if it could infer its thematic-role group as *Theme::Agent* and its parent group *Giving::Donor* correctly. Although the ensemble of various groupings seems promising, we need an additional procedure to prioritize the groupings for the case where the models for multiple role groupings disagree; for example, it is unsatisfactory if two models assign the groups *Giving::Theme* and *Theme::Agent* to the same phrase.

5.2 Role groups as feature functions

We thus propose another approach that incorporates group information as feature functions. We model the conditional probability $P(y|f, \mathbf{x})$ by using the maximum entropy framework,

$$p(y|f, \mathbf{x}) = \frac{\exp(\sum_i \lambda_i g_i(\mathbf{x}, y))}{\sum_{y \in Y_f} \exp(\sum_i \lambda_i g_i(\mathbf{x}, y))}. \quad (3)$$

Here, $G = \{g_i\}$ denotes a set of n feature functions, and $\Lambda = \{\lambda_i\}$ denotes a weight vector for the feature functions.

In general, feature functions for the maximum entropy model are designed as indicator functions for possible pairs of x_j and y. For example, the event where the head word of x is "you" ($x_1 = 1$) and x plays the role Commerce_sell::Seller in a sentence is expressed by the indicator function,

$$g_1^{\textit{role}}(\mathbf{x}, y) = \begin{cases} 1 & (x_1 = 1 \wedge \\ & y = \text{Commerce\_sell::Seller}) \\ 0 & (\text{otherwise}) \end{cases}. \quad (4)$$

We call this kind of feature function an *x-role*.

In order to incorporate role groups into the model, we also include all feature functions for possible pairs of x_j and role groups. Equation 5 is an example of a feature function for instances where the head word of x is "you" and y is in the role group *Theme::Agent*,

$$g_2^{\textit{theme}}(\mathbf{x}, y) = \begin{cases} 1 & (x_1 = 1 \wedge \\ & y \in \textit{Theme::Agent}) \\ 0 & (\text{otherwise}) \end{cases}. \quad (5)$$

Thus, this feature function fires for the roles wherever the head word "you" plays *Agent* (e.g., Commerce_sell::Seller, Commerce_buy::Buyer and Giving::Donor). We call this kind of feature function an *x-group* function.

In this way, we obtain *x-group* functions for all grouping methods, e.g., $g_k^{\textit{theme}}$, $g_k^{\textit{hierarchy}}$. The role-group features will receive more training instances by collecting instances for fine-grained roles. Thus, semantic roles with few training instances are expected to receive additional clues from other training instances via role-group features. Another advantage of this approach is that the usefulness of the different role groups is determined by the training processes in terms of weights of feature functions. Thus, we do not need to assume that we have found the best criterion for grouping roles; we can allow a training process to choose the criterion. We will discuss the contributions of different groupings in the experiments.

5.3 Comparison with related work

Baldewein et al. (2004) suggested an approach that uses role descriptors and hierarchical relations as criteria for generalizing semantic roles in FrameNet. They created a classifier for each frame, additionally using training instances for the role *A* to train the classifier for the role *B*, if the roles *A* and *B* were judged as similar by a criterion. This approach performs similarly to the overwriting approach, and it may obscure the differences among roles. Therefore, they only re-used the descriptors as a similarity measure for the roles whose *coreness* was *peripheral*. [2]

In contrast, we use all kinds of role descriptors to construct groups. Since we use the feature functions for both the original roles and their groups, appropriate units for classification are determined automatically in the training process.

6 Experiment and Discussion

We used the training set of the Semeval-2007 Shared task (Baker et al., 2007) in order to ascertain the contributions of role groups. This dataset consists of the corpus of FrameNet release 1.3 (containing roughly 150,000 annotations), and an additional full-text annotation dataset. We randomly extracted 10% of the dataset for testing, and used the remainder (90%) for training.

Performance was measured by micro- and macro-averaged F1 (Chang and Zheng, 2008) with respect to a variety of roles. The micro average biases each F1 score by the frequencies of the roles,

[2] In FrameNet, each role is assigned one of four different types of *coreness* (*core*, *core-unexpressed*, *peripheral*, *extra-thematic*) It represents the conceptual necessity of the roles in the frame to which it belongs.

and the average is equal to the classification accuracy when we calculate it with all of the roles in the test set. In contrast, the macro average does not bias the scores, thus the roles having a small number of instances affect the average more than the micro average.

6.1 Experimental settings

We constructed a baseline classifier that uses only the *x-role* features. The feature design is similar to that of the previous studies (Màrquez et al., 2008). The characteristics of *x* are: **frame, frame evoking word, head word, content word** (Surdeanu et al., 2003), **first/last word, head word of left/right sister, phrase type, position, voice, syntactic path (directed/undirected/partial), governing category** (Gildea and Jurafsky, 2002), **WordNet supersense in the phrase**, combination features of **frame evoking word & headword**, combination features of **frame evoking word & phrase type**, and combination features of **voice & phrase type**. We also used **PoS tags** and **stem forms** as extra features of any word-features.

We employed Charniak and Johnson's reranking parser (Charniak and Johnson, 2005) to analyze syntactic trees. As an alternative for the traditional named-entity features, we used WordNet supersenses: 41 coarse-grained semantic categories of words such as person, plant, state, event, time, location. We used Ciaramita and Altun's Super Sense Tagger (Ciaramita and Altun, 2006) to tag the supersenses. The baseline system achieved 89.00% with respect to the micro-averaged F1.

The *x-group* features were instantiated similarly to the *x-role* features; the *x-group* features combined the characteristics of *x* with the role groups presented in this paper. The total number of features generated for all *x-roles* and *x-groups* was 74,873,602. The optimal weights Λ of the features were obtained by the maximum a posterior (MAP) estimation. We maximized an L_2-regularized log-likelihood of the training set using the Limited-memory BFGS (L-BFGS) method (Nocedal, 1980).

6.2 Effect of role groups

Table 1 shows the micro and macro averages of F1 scores. Each role group type improved the micro average by 0.5 to 1.7 points. The best result was obtained by using all types of groups together. The result indicates that different kinds of group complement each other with respect to semantic role generalization. Baldewein et al. (2004) reported that hierarchical relations did not perform well for their method and experimental setting; however, we found that significant improvements could also be achieved with hierarchical relations. We also tried a traditional label-replacing approach with role descriptors (in the third row of Table 1). The comparison between the second and third rows indicates that mixing the original fine-grained roles and the role groups does result in a more accurate classification.

Feature	Micro	Macro	−Err.
Baseline	89.00	68.50	0.00
role descriptor	90.78	**76.58**	16.17
role descriptor (replace)	90.23	76.19	11.23
hierarchical relation	90.25	72.41	11.40
semantic type	90.36	74.51	12.38
VN thematic role	89.50	69.21	4.52
All	**91.10**	75.92	19.16

Table 1: The accuracy and error reduction rate of role classification for each type of role group.

Feature	#instances	Pre.	Rec.	Micro
baseline	≤ 10	63.89	38.00	47.66
	≤ 20	69.01	51.26	58.83
	≤ 50	75.84	65.85	70.50
+ all groups	≤ 10	72.57	55.85	63.12
	≤ 20	76.30	65.41	70.43
	≤ 50	80.86	74.59	77.60

Table 2: The effect of role groups on the roles with few instances.

By using all types of groups together, the model reduced 19.16 % of the classification errors from the baseline. Moreover, the macro-averaged F1 scores clearly showed improvements resulting from using role groups. In order to determine the reason for the improvements, we measured the precision, recall, and F1-scores with respect to roles for which the number of training instances was at most 10, 20, and 50. In Table 2, we show that the micro-averaged F1 score for roles having 10 instances or less was improved (by 15.46 points) when all role groups were used. This result suggests the reason for the effect of role groups; by bridging similar semantic roles, they supply roles having a small number of instances with the information from other roles.

6.3 Analyses of role descriptors

In Table 1, the largest improvement was obtained by the use of role descriptors. We analyze the effect of role descriptors in detail in Tables 3 and 4. Table 3 shows the micro-averaged F1 scores of all

Coreness	#roles	#instances/#role	#groups	#instances/#group	#roles/#group
Core	1902	122.06	655	354.4	2.9
Peripheral	1924	25.24	250	194.3	**7.7**
Extra-thematic	763	13.90	171	62.02	4.5

Table 4: The analysis of the numbers of roles, instances, and role-descriptor groups, for each type of coreness.

Coreness	Micro
Baseline	89.00
Core	89.51
Peripheral	**90.12**
Extra-thematic	89.09
All	90.77

Table 3: The effect of employing role-descriptor groups of each type of coreness.

semantic roles when we use role-descriptor groups constructed from each type of coreness (core[3], peripheral, and extra-thematic) individually. The *peripheral* type generated the largest improvements.

Table 4 shows the number of roles associated with each type of coreness (#roles), the number of instances for the original roles (#instances/#role), the number of groups for each type of coreness (#groups), the number of instances for each group (#instances/#group), and the number of roles per each group (#roles/#group). In the *peripheral* type, the role descriptors subdivided 1,924 distinct roles into 250 groups, each of which contained 7.7 roles on average. The *peripheral* type included semantic roles such as *place*, *time*, *reason*, *duration*. These semantic roles appear in many frames, because they have general meanings that can be shared by different frames. Moreover, the semantic roles of *peripheral* type originally occurred in only a small number (25.24) of training instances on average. Thus, we infer that the *peripheral* type generated the largest improvement because semantic roles in this type acquired the greatest benefit from the generalization.

6.4 Hierarchical relations and relation types

We analyzed the contributions of the FrameNet hierarchy for each type of role-to-role relations and for different depths of grouping. Table 5 shows the micro-averaged F1 scores obtained from various relation types and depths. The *Inheritance* and *Using* relations resulted in a slightly better accuracy than the other types. We did not observe any real differences among the remaining five relation types, possibly because there were few semantic roles associated with these types. We obtained better results by using not only groups for parent roles, but also groups for all ancestors. The best result was obtained by using all relations in the hierarchy.

[3]We include *Core-unexpressed* in *core*, because it has a property of *core* inside one frame.

No.	Relation Type	Micro
-	baseline	89.00
1	+ Inheritance (children)	89.52
2	+ Inheritance (descendants)	**89.70**
3	+ Using (children)	89.35
4	+ Using (descendants)	**89.37**
5	+ Perspective on (children)	89.01
6	+ Perspective on (descendants)	89.01
7	+ Subframe (children)	89.04
8	+ Subframe (descendants)	89.05
9	+ Causative of (parents)	89.03
10	+ Causative of (ancestors)	89.03
11	+ Inchoative of (parents)	89.02
12	+ Inchoative of (ancestors)	89.02
13	+ Precedes (children)	89.01
14	+ Precedes (descendants)	89.03
15	+ Precedes (parents)	89.00
16	+ Precedes (ancestors)	89.00
18	+ all relations (2,4,6,8,10,12,14)	**90.25**

Table 5: Comparison of the accuracy with different types of hierarchical relations.

6.5 Analyses of different grouping criteria

Table 6 reports the precision, recall, and micro-averaged F1 scores of semantic roles with respect to each coreness type.[4] In general, semantic roles of the *core* coreness were easily identified by all of the grouping criteria; even the baseline system obtained an F1 score of 91.93. For identifying semantic roles of the *peripheral* and *extra-thematic* types of coreness, the simplest solution, the descriptor criterion, outperformed other criteria.

In Table 7, we categorize feature functions whose weights are in the top 1000 in terms of greatest absolute value. The behaviors of the role groups can be distinguished by the following two characteristics. Groups of role descriptors and semantic types have large weight values for the first word and supersense features, which capture the characteristics of adjunctive phrases. The original roles and hierarchical-relation groups have strong

[4]The figures of role descriptors in Tables 4 and 6 differ. In Table 4, we measured the performance when we used one or all types of coreness for training. In contrast, in Table 6, we used all types of coreness for training, but computed the performance of semantic roles for each coreness separately.

Feature	Type	Pre.	Rec.	Micro
baseline	c	91.07	92.83	91.93
	p	81.05	76.03	78.46
	e	78.17	66.51	71.87
+ descriptor group	c	92.50	93.41	92.95
	p	84.32	82.72	83.51
	e	80.91	69.59	74.82
+ hierarchical relation class	c	92.10	93.28	92.68
	p	82.23	79.84	81.01
	e	77.94	65.58	71.23
+ semantic type group	c	92.23	93.31	92.77
	p	83.66	81.76	82.70
	e	80.29	67.26	73.20
+ VN thematic role group	c	91.57	93.06	92.31
	p	80.66	76.95	78.76
	e	78.12	66.60	71.90
+ all group	c	92.66	93.61	93.13
	p	84.13	82.51	83.31
	e	80.77	68.56	74.17

Table 6: The precision and recall of each type of coreness with role groups. Type represents the type of coreness; c denotes core, p denotes peripheral, and e denotes extra-thematic.

features of x	class type				
	or	hr	rl	st	vn
frame	0	4	0	1	0
evoking word	3	4	7	3	0
ew & hw stem	9	34	20	8	0
ew & phrase type	11	7	11	3	1
head word	13	19	8	3	1
hw stem	11	17	8	8	1
content word	7	19	12	3	0
cw stem	11	26	13	5	0
cw PoS	4	5	14	15	2
directed path	19	27	24	6	7
undirected path	21	35	17	2	6
partial path	15	18	16	13	5
last word	15	18	12	3	2
first word	11	23	53	26	10
supersense	7	7	35	25	4
position	4	6	30	9	5
others	27	29	33	19	6
total	188	298	313	152	50

Table 7: The analysis of the top 1000 feature functions. Each number denotes the number of feature functions categorized in the corresponding cell. Notations for the columns are as follows. 'or': original role, 'hr': hierarchical relation, 'rd': role descriptor, 'st': semantic type, and 'vn': VerbNet thematic role.

associations with lexical and structural characteristics such as the syntactic path, content word, and head word. Table 7 suggests that role-descriptor groups and semantic-type groups are effective for *peripheral* or adjunctive roles, and hierarchical relation groups are effective for *core* roles.

7 Conclusion

We have described different criteria for generalizing semantic roles in FrameNet. They were: role hierarchy, human-understandable descriptors of roles, semantic types of filler phrases, and mappings from FrameNet roles to thematic roles of VerbNet. We also proposed a feature design that combines and weights these criteria using the training data. The experimental result of the role classification task showed a 19.16% of the error reduction and a 7.42% improvement in the macro-averaged F1 score. In particular, the method we have presented was able to classify roles having few instances. We confirmed that modeling the role generalization at feature level was better than the conventional approach that replaces semantic role labels.

Each criterion presented in this paper improved the accuracy of classification. The most successful criterion was the use of human-understandable role descriptors. Unfortunately, the FrameNet hierarchy did not outperform the role descriptors, contrary to our expectations. A future direction of this study would be to analyze the weakness of the FrameNet hierarchy in order to discuss possible improvement of the usage and annotations of the hierarchy.

Since we used the latest release of FrameNet in order to use a greater number of hierarchical role-to-role relations, we could not make a direct comparison of performance with that of existing systems; however we may say that the 89.00% F1 micro-average of our baseline system is roughly comparable to the 88.93% value of Bejan and Hathaway (2007) for SemEval-2007 (Baker et al., 2007). [5] In addition, the methodology presented in this paper applies generally to any SRL resources; we are planning to determine several grouping criteria from existing linguistic resources and to apply the methodology to the PropBank corpus.

Acknowledgments

The authors thank Sebastian Riedel for his useful comments on our work. This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan).

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of Coling-ACL 1998*, pages 86–90.

Collin Baker, Michael Ellsworth, and Katrin Erk. 2007. Semeval-2007 task 19: Frame semantic struc-

[5] There were two participants that performed whole SRL in SemEval-2007. Bejan and Hathaway (2007) evaluated role classification accuracy separately for the training data.

ture extraction. In *Proceedings of SemEval-2007*, pages 99–104.

Ulrike Baldewein, Katrin Erk, Sebastian Padó, and Detlef Prescher. 2004. Semantic role labeling with similarity based generalization using EM-based clustering. In *Proceedings of Senseval-3*, pages 64–68.

Cosmin Adrian Bejan and Chris Hathaway. 2007. UTD-SRL: A Pipeline Architecture for Extracting Frame Semantic Structures. In *Proceedings of SemEval-2007*, pages 460–463. Association for Computational Linguistics.

X. Chang and Q. Zheng. 2008. Knowledge Element Extraction for Knowledge-Based Learning Resources Organization. *Lecture Notes in Computer Science*, 4823:102–113.

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

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of EMNLP-2006*, pages 594–602.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Ana-Maria Giuglea and Alessandro Moschitti. 2006. Semantic role labeling via FrameNet, VerbNet and PropBank. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL*, pages 929–936.

Andrew Gordon and Reid Swanson. 2007. Generalizing semantic role annotations across syntactically similar verbs. In *Proceedings of ACL-2007*, pages 192–199.

Edward Loper, Szu-ting Yi, and Martha Palmer. 2007. Combining lexical resources: Mapping between propbank and verbnet. In *Proceedings of the 7th International Workshop on Computational Semantics*, pages 118–128.

Lluís Màrquez, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. 2008. Semantic role labeling: an introduction to the special issue. *Computational linguistics*, 34(2):145–159.

Alessandro Moschitti, Ana-Maria Giuglea, Bonaventura Coppola, and Roberto Basili. 2005. Hierarchical semantic role labeling. In *Proceedings of CoNLL-2005*, pages 201–204.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question answer classification. In *Proceedings of ACL-07*, pages 776–783.

Srini Narayanan and Sanda Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of Coling-2004*, pages 693–701.

Jorge Nocedal. 1980. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of EMNLP-CoNLL 2007*, pages 12–21.

Lei Shi and Rada Mihalcea. 2005. Putting Pieces Together: Combining FrameNet, VerbNet and WordNet for Robust Semantic Parsing. In *Proceedings of CICLing-2005*, pages 100–111.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL-2003*, pages 8–15.

Szu-ting Yi, Edward Loper, and Martha Palmer. 2007. Can semantic roles generalize across genres? In *Proceedings of HLT-NAACL 2007*, pages 548–555.

Beñat Zapirain, Eneko Agirre, and Lluís Màrquez. 2008. Robustness and generalization of role sets: PropBank vs. VerbNet. In *Proceedings of ACL-08: HLT*, pages 550–558.

Unsupervised Argument Identification for Semantic Role Labeling

Omri Abend[1] **Roi Reichart**[2] **Ari Rappoport**[1]

[1]Institute of Computer Science , [2]ICNC
Hebrew University of Jerusalem
{omria01|roiri|arir}@cs.huji.ac.il

Abstract

The task of Semantic Role Labeling (SRL) is often divided into two sub-tasks: verb argument identification, and argument classification. Current SRL algorithms show lower results on the identification sub-task. Moreover, most SRL algorithms are supervised, relying on large amounts of manually created data. In this paper we present an unsupervised algorithm for identifying verb arguments, where the only type of annotation required is POS tagging. The algorithm makes use of a fully unsupervised syntactic parser, using its output in order to detect clauses and gather candidate argument collocation statistics. We evaluate our algorithm on PropBank10, achieving a precision of 56%, as opposed to 47% of a strong baseline. We also obtain an 8% increase in precision for a Spanish corpus. This is the first paper that tackles unsupervised verb argument identification without using manually encoded rules or extensive lexical or syntactic resources.

1 Introduction

Semantic Role Labeling (SRL) is a major NLP task, providing a shallow sentence-level semantic analysis. SRL aims at identifying the relations between the predicates (usually, verbs) in the sentence and their associated arguments.

The SRL task is often viewed as consisting of two parts: argument identification (ARGID) and argument classification. The former aims at identifying the arguments of a given predicate present in the sentence, while the latter determines the type of relation that holds between the identified arguments and their corresponding predicates. The division into two sub-tasks is justified by the fact that they are best addressed using different feature sets (Pradhan et al., 2005). Performance in the ARGID stage is a serious bottleneck for general SRL performance, since only about 81% of the arguments are identified, while about 95% of the identified arguments are labeled correctly (Màrquez et al., 2008).

SRL is a complex task, which is reflected by the algorithms used to address it. A standard SRL algorithm requires thousands to dozens of thousands sentences annotated with POS tags, syntactic annotation and SRL annotation. Current algorithms show impressive results but only for languages and domains where plenty of annotated data is available, e.g., English newspaper texts (see Section 2). Results are markedly lower when testing is on a domain wider than the training one, even in English (see the WSJ-Brown results in (Pradhan et al., 2008)).

Only a small number of works that do not require manually labeled SRL training data have been done (Swier and Stevenson, 2004; Swier and Stevenson, 2005; Grenager and Manning, 2006). These papers have replaced this data with the VerbNet (Kipper et al., 2000) lexical resource or a set of manually written rules and supervised parsers.

A potential answer to the SRL training data bottleneck are unsupervised SRL models that require little to no manual effort for their training. Their output can be used either by itself, or as training material for modern supervised SRL algorithms.

In this paper we present an algorithm for unsupervised argument identification. The only type of annotation required by our algorithm is POS tag-

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 28–36,
Suntec, Singapore, 2-7 August 2009.

ging, which needs relatively little manual effort.

The algorithm consists of two stages. As preprocessing, we use a fully unsupervised parser to parse each sentence. Initially, the set of possible arguments for a given verb consists of all the constituents in the parse tree that do not contain that predicate. The first stage of the algorithm attempts to detect the minimal clause in the sentence that contains the predicate in question. Using this information, it further reduces the possible arguments only to those contained in the minimal clause, and further prunes them according to their position in the parse tree. In the second stage we use pointwise mutual information to estimate the collocation strength between the arguments and the predicate, and use it to filter out instances of weakly collocating predicate argument pairs.

We use two measures to evaluate the performance of our algorithm, precision and F-score. Precision reflects the algorithm's applicability for creating training data to be used by supervised SRL models, while the standard SRL F-score measures the model's performance when used by itself. The first stage of our algorithm is shown to outperform a strong baseline both in terms of F-score and of precision. The second stage is shown to increase precision while maintaining a reasonable recall.

We evaluated our model on sections 2-21 of Propbank. As is customary in unsupervised parsing work (e.g. (Seginer, 2007)), we bounded sentence length by 10 (excluding punctuation). Our first stage obtained a precision of 52.8%, which is more than 6% improvement over the baseline. Our second stage improved precision to nearly 56%, a 9.3% improvement over the baseline. In addition, we carried out experiments on Spanish (on sentences of length bounded by 15, excluding punctuation), achieving an increase of over 7.5% in precision over the baseline. Our algorithm increases F–score as well, showing an 1.8% improvement over the baseline in English and a 2.2% improvement in Spanish.

Section 2 reviews related work. In Section 3 we detail our algorithm. Sections 4 and 5 describe the experimental setup and results.

2 Related Work

The advance of machine learning based approaches in this field owes to the usage of large scale annotated corpora. English is the most studied language, using the FrameNet (FN) (Baker et al., 1998) and PropBank (PB) (Palmer et al., 2005) resources. PB is a corpus well suited for evaluation, since it annotates every non-auxiliary verb in a real corpus (the WSJ sections of the Penn Treebank). PB is a standard corpus for SRL evaluation and was used in the CoNLL SRL shared tasks of 2004 (Carreras and Màrquez, 2004) and 2005 (Carreras and Màrquez, 2005).

Most work on SRL has been supervised, requiring dozens of thousands of SRL annotated training sentences. In addition, most models assume that a syntactic representation of the sentence is given, commonly in the form of a parse tree, a dependency structure or a shallow parse. Obtaining these is quite costly in terms of required human annotation.

The first work to tackle SRL as an independent task is (Gildea and Jurafsky, 2002), which presented a supervised model trained and evaluated on FrameNet. The CoNLL shared tasks of 2004 and 2005 were devoted to SRL, and studied the influence of different syntactic annotations and domain changes on SRL results. *Computational Linguistics* has recently published a special issue on the task (Màrquez et al., 2008), which presents state-of-the-art results and surveys the latest achievements and challenges in the field.

Most approaches to the task use a multi-level approach, separating the task to an ARGID and an argument classification sub-tasks. They then use the unlabeled argument structure (without the semantic roles) as training data for the ARGID stage and the entire data (perhaps with other features) for the classification stage. Better performance is achieved on the classification, where state-of-the-art supervised approaches achieve about 81% F-score on the in-domain identification task, of which about 95% are later labeled correctly (Màrquez et al., 2008).

There have been several exceptions to the standard architecture described in the last paragraph. One suggestion poses the problem of SRL as a sequential tagging of words, training an SVM classifier to determine for each word whether it is inside, outside or in the beginning of an argument (Hacioglu and Ward, 2003). Other works have integrated argument classification and identification into one step (Collobert and Weston, 2007), while others went further and combined the former two along with parsing into a single model (Musillo

and Merlo, 2006).

Work on less supervised methods has been scarce. Swier and Stevenson (2004) and Swier and Stevenson (2005) presented the first model that does not use an SRL annotated corpus. However, they utilize the extensive verb lexicon VerbNet, which lists the possible argument structures allowable for each verb, and supervised syntactic tools. Using VerbNet along with the output of a rule-based chunker (in 2004) and a supervised syntactic parser (in 2005), they spot instances in the corpus that are very similar to the syntactic patterns listed in VerbNet. They then use these as seed for a bootstrapping algorithm, which consequently identifies the verb arguments in the corpus and assigns their semantic roles.

Another less supervised work is that of (Grenager and Manning, 2006), which presents a Bayesian network model for the argument structure of a sentence. They use EM to learn the model's parameters from unannotated data, and use this model to tag a test corpus. However, ARGID was not the task of that work, which dealt solely with argument classification. ARGID was performed by manually-created rules, requiring a supervised or manual syntactic annotation of the corpus to be annotated.

The three works above are relevant but incomparable to our work, due to the extensive amount of supervision (namely, VerbNet and a rule-based or supervised syntactic system) they used, both in detecting the syntactic structure and in detecting the arguments.

Work has been carried out in a few other languages besides English. Chinese has been studied in (Xue, 2008). Experiments on Catalan and Spanish were done in SemEval 2007 (Màrquez et al., 2007) with two participating systems. Attempts to compile corpora for German (Burdchardt et al., 2006) and Arabic (Diab et al., 2008) are also underway. The small number of languages for which extensive SRL annotated data exists reflects the considerable human effort required for such endeavors.

Some SRL works have tried to use unannotated data to improve the performance of a base supervised model. Methods used include bootstrapping approaches (Gildea and Jurafsky, 2002; Kate and Mooney, 2007), where large unannotated corpora were tagged with SRL annotation, later to be used to retrain the SRL model. Another approach used similarity measures either between verbs (Gordon and Swanson, 2007) or between nouns (Gildea and Jurafsky, 2002) to overcome lexical sparsity. These measures were estimated using statistics gathered from corpora augmenting the model's training data, and were then utilized to generalize across similar verbs or similar arguments.

Attempts to substitute full constituency parsing by other sources of syntactic information have been carried out in the SRL community. Suggestions include posing SRL as a sequence labeling problem (Màrquez et al., 2005) or as an edge tagging problem in a dependency representation (Hacioglu, 2004). Punyakanok et al. (2008) provide a detailed comparison between the impact of using shallow vs. full constituency syntactic information in an English SRL system. Their results clearly demonstrate the advantage of using full annotation.

The identification of arguments has also been carried out in the context of automatic subcategorization frame acquisition. Notable examples include (Manning, 1993; Briscoe and Carroll, 1997; Korhonen, 2002) who all used statistical hypothesis testing to filter a parser's output for arguments, with the goal of compiling verb subcategorization lexicons. However, these works differ from ours as they attempt to characterize the behavior of a verb type, by collecting statistics from various instances of that verb, and not to determine which are the arguments of specific verb instances.

The algorithm presented in this paper performs unsupervised clause detection as an intermediate step towards argument identification. Supervised clause detection was also tackled as a separate task, notably in the CoNLL 2001 shared task (Tjong Kim Sang and Dèjean, 2001). Clause information has been applied to accelerating a syntactic parser (Glaysher and Moldovan, 2006).

3 Algorithm

In this section we describe our algorithm. It consists of two stages, each of which reduces the set of argument candidates, which a-priori contains all consecutive sequences of words that do not contain the predicate in question.

3.1 Algorithm overview

As pre-processing, we use an unsupervised parser that generates an unlabeled parse tree for each sen-

tence (Seginer, 2007). This parser is unique in that it is able to induce a bracketing (unlabeled parsing) from raw text (without even using POS tags) achieving state-of-the-art results. Since our algorithm uses millions to tens of millions sentences, we must use very fast tools. The parser's high speed (thousands of words per second) enables us to process these large amounts of data.

The only type of supervised annotation we use is POS tagging. We use the taggers MXPOST (Ratnaparkhi, 1996) for English and TreeTagger (Schmid, 1994) for Spanish, to obtain POS tags for our model.

The first stage of our algorithm uses linguistically motivated considerations to reduce the set of possible arguments. It does so by confining the set of argument candidates only to those constituents which obey the following two restrictions. First, they should be contained in the minimal clause containing the predicate. Second, they should be k-th degree cousins of the predicate in the parse tree. We propose a novel algorithm for clause detection and use its output to determine which of the constituents obey these two restrictions.

The second stage of the algorithm uses pointwise mutual information to rule out constituents that appear to be weakly collocating with the predicate in question. Since a predicate greatly restricts the type of arguments with which it may appear (this is often referred to as "selectional restrictions"), we expect it to have certain characteristic arguments with which it is likely to collocate.

3.2 Clause detection stage

The main idea behind this stage is the observation that most of the arguments of a predicate are contained within the minimal clause that contains the predicate. We tested this on our development data – section 24 of the WSJ PTB, where we saw that 86% of the arguments that are also constituents (in the gold standard parse) were indeed contained in that minimal clause (as defined by the tree label types in the gold standard parse that denote a clause, e.g., S, SBAR). Since we are not provided with clause annotation (or any label), we attempted to detect them in an unsupervised manner. Our algorithm attempts to find sub-trees within the parse tree, whose structure resembles the structure of a full sentence. This approximates the notion of a clause.

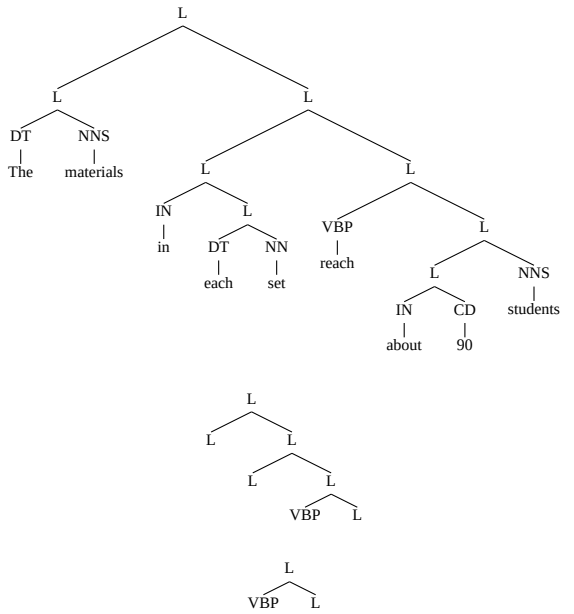


Figure 1: An example of an unlabeled POS tagged parse tree. The middle tree is the ST of 'reach' with the root as the encoded ancestor. The bottom one is the ST with its parent as the encoded ancestor.

Statistics gathering. In order to detect which of the verb's ancestors is the minimal clause, we score each of the ancestors and select the one that maximizes the score. We represent each ancestor using its *Spinal Tree* (ST). The ST of a given verb's ancestor is obtained by replacing all the constituents that do not contain the verb by a leaf having a label. This effectively encodes all the k-th degree cousins of the verb (for every k). The leaf labels are either the word's POS in case the constituent is a leaf, or the generic label "L" denoting a non-leaf. See Figure 1 for an example.

In this stage we collect statistics of the occurrences of STs in a large corpus. For every ST in the corpus, we count the number of times it occurs in a form we consider to be a clause (positive examples), and the number of times it appears in other forms (negative examples).

Positive examples are divided into two main types. First, when the ST encodes the root ancestor (as in the middle tree of Figure 1); second, when the ancestor complies to a clause lexico-syntactic pattern. In many languages there is a small set of lexico-syntactic patterns that mark a clause, e.g. the English 'that', the German 'dass' and the Spanish 'que'. The patterns which were used in our experiments are shown in Figure 2.

For each verb instance, we traverse over its an-

English
TO + VB. The constituent starts with "to" followed by a verb in infinitive form.
WP. The constituent is preceded by a Wh-pronoun.
That. The constituent is preceded by a "that" marked by an "IN" POS tag indicating that it is a subordinating conjunction.
Spanish
CQUE. The constituent is preceded by a word with the POS "CQUE" which denotes the word "que" as a conjunction.
INT. The constituent is preceded by a word with the POS "INT" which denotes an interrogative pronoun.
CSUB. The constituent is preceded by a word with one of the POSs "CSUBF", "CSUBI" or "CSUBX", which denote a subordinating conjunction.

Figure 2: The set of lexico-syntactic patterns that mark clauses which were used by our model.

cestors from top to bottom. For each of them we update the following counters: $sentence(ST)$ for the root ancestor's ST, $pattern_i(ST)$ for the ones complying to the i-th lexico-syntactic pattern and $negative(ST)$ for the other ancestors[1].

Clause detection. At test time, when detecting the minimal clause of a verb instance, we use the statistics collected in the previous stage. Denote the ancestors of the verb with $A_1 \dots A_m$. For each of them, we calculate $clause(ST_{A_j})$ and $total(ST_{A_j})$. $clause(ST_{A_j})$ is the sum of $sentence(ST_{A_j})$ and $pattern_i(ST_{A_j})$ if this ancestor complies to the i-th pattern (if there is no such pattern, $clause(ST_{A_j})$ is equal to $sentence(ST_{A_j})$). $total(ST_{A_j})$ is the sum of $clause(ST_{A_j})$ and $negative(ST_{A_j})$.

The selected ancestor is given by:

$$(1)\ A_{max} = argmax_{A_j} \frac{clause(ST_{A_j})}{total(ST_{A_j})}$$

An ST whose $total(ST)$ is less than a small threshold[2] is not considered a candidate to be the minimal clause, since its statistics may be unreliable. In case of a tie, we choose the lowest constituent that obtained the maximal score. If there is only one verb in the sentence[3] or if $clause(ST_{A_j}) = 0$ for every $1 \leq j \leq m$, we choose the top level constituent by default to be the minimal clause containing the verb. Otherwise, the minimal clause is defined to be the yield of the selected ancestor.

Argument identification. For each predicate in the corpus, its argument candidates are now defined to be the constituents contained in the minimal clause containing the predicate. However, these constituents may be (and are) nested within each other, violating a major restriction on SRL arguments. Hence we now prune our set, by keeping only the siblings of all of the verb's ancestors, as is common in supervised SRL (Xue and Palmer, 2004).

3.3 Using collocations

We use the following observation to filter out some superfluous argument candidates: since the arguments of a predicate many times bear a semantic connection with that predicate, they consequently tend to collocate with it.

We collect collocation statistics from a large corpus, which we annotate with parse trees and POS tags. We mark arguments using the argument detection algorithm described in the previous two sections, and extract all (predicate, argument) pairs appearing in the corpus. Recall that for each sentence, the arguments are a subset of the constituents in the parse tree.

We use two representations of an argument: one is the POS tag sequence of the terminals contained in the argument, the other is its head word[4]. The predicate is represented as the conjunction of its lemma with its POS tag.

Denote the number of times a predicate x appeared with an argument y by n_{xy}. Denote the total number of (predicate, argument) pairs by N. Using these notations, we define the following quantities: $n_x = \Sigma_y n_{xy}$, $n_y = \Sigma_x n_{xy}$, $p(x) = \frac{n_x}{N}$, $p(y) = \frac{n_y}{N}$ and $p(x,y) = \frac{n_{xy}}{N}$. The pointwise mutual information of x and y is then given by:

[1] If while traversing the tree, we encounter an ancestor whose first word is preceded by a coordinating conjunction (marked by the POS tag "CC"), we refrain from performing any additional counter updates. Structures containing coordinating conjunctions tend not to obey our lexico-syntactic rules.

[2] We used 4 per million sentences, derived from development data.

[3] In this case, every argument in the sentence must be related to that verb.

[4] Since we do not have syntactic labels, we use an approximate notion. For English we use the Bikel parser default head word rules (Bikel, 2004). For Spanish, we use the leftmost word.

(2) $PMI(x,y) = log\frac{p(x,y)}{p(x)\cdot p(y)} = log\frac{n_{xy}}{(n_x \cdot n_y)/N}$

PMI effectively measures the ratio between the number of times x and y appeared together and the number of times they were expected to appear, had they been independent.

At test time, when an (x,y) pair is observed, we check if $PMI(x,y)$, computed on the large corpus, is lower than a threshold α for either of x's representations. If this holds, for at least one representation, we prune all instances of that (x,y) pair. The parameter α may be selected differently for each of the argument representations.

In order to avoid using unreliable statistics, we apply this for a given pair only if $\frac{n_x \cdot n_y}{N} > r$, for some parameter r. That is, we consider $PMI(x,y)$ to be reliable, only if the denominator in equation (2) is sufficiently large.

4 Experimental Setup

Corpora. We used the PropBank corpus for development and for evaluation on English. Section 24 was used for the development of our model, and sections 2 to 21 were used as our test data. The free parameters of the collocation extraction phase were tuned on the development data. Following the unsupervised parsing literature, multiple brackets and brackets covering a single word are omitted. We exclude punctuation according to the scheme of (Klein, 2005). As is customary in unsupervised parsing (e.g. (Seginer, 2007)), we bounded the lengths of the sentences in the corpus to be at most 10 (excluding punctuation). This results in 207 sentences in the development data, containing a total of 132 different verbs and 173 verb instances (of the non-auxiliary verbs in the SRL task, see 'evaluation' below) having 403 arguments. The test data has 6007 sentences containing 1008 different verbs and 5130 verb instances (as above) having 12436 arguments.

Our algorithm requires large amounts of data to gather argument structure and collocation patterns. For the statistics gathering phase of the clause detection algorithm, we used 4.5M sentences of the NANC (Graff, 1995) corpus, bounding their length in the same manner. In order to extract collocations, we used 2M sentences from the British National Corpus (Burnard, 2000) and about 29M sentences from the Dmoz corpus (Gabrilovich and Markovitch, 2005). Dmoz is a web corpus obtained by crawling and cleaning the URLs in the Open Directory Project (dmoz.org). All of the above corpora were parsed using Seginer's parser and POS-tagged by MXPOST (Ratnaparkhi, 1996).

For our experiments on Spanish, we used 3.3M sentences of length at most 15 (excluding punctuation) extracted from the Spanish Wikipedia. Here we chose to bound the length by 15 due to the smaller size of the available test corpus. The same data was used both for the first and the second stages. Our development and test data were taken from the training data released for the SemEval 2007 task on semantic annotation of Spanish (Màrquez et al., 2007). This data consisted of 1048 sentences of length up to 15, from which 200 were randomly selected as our development data and 848 as our test data. The development data included 313 verb instances while the test data included 1279. All corpora were parsed using the Seginer parser and tagged by the "TreeTagger" (Schmid, 1994).

Baselines. Since this is the first paper, to our knowledge, which addresses the problem of unsupervised argument identification, we do not have any previous results to compare to. We instead compare to a baseline which marks all k-th degree cousins of the predicate (for every k) as arguments (this is the second pruning we use in the clause detection stage). We name this baseline the ALL COUSINS baseline. We note that a random baseline would score very poorly since any sequence of terminals which does not contain the predicate is a possible candidate. Therefore, beating this random baseline is trivial.

Evaluation. Evaluation is carried out using standard SRL evaluation software[5]. The algorithm is provided with a list of predicates, whose arguments it needs to annotate. For the task addressed in this paper, non-consecutive parts of arguments are treated as full arguments. A match is considered each time an argument in the gold standard data matches a marked argument in our model's output. An unmatched argument is an argument which appears in the gold standard data, and fails to appear in our model's output, and an excessive argument is an argument which appears in our model's output but does not appear in the gold standard. Precision and recall are defined accordingly. We report an F-score as well (the harmonic mean of precision and recall). We do not attempt

[5] http://www.lsi.upc.edu/∼srlconll/soft.html#software.

to identify multi-word verbs, and therefore do not report the model's performance in identifying verb boundaries.

Since our model detects clauses as an intermediate product, we provide a separate evaluation of this task for the English corpus. We show results on our development data. We use the standard parsing F-score evaluation measure. As a gold standard in this evaluation, we mark for each of the verbs in our development data the minimal clause containing it. A minimal clause is the lowest ancestor of the verb in the parse tree that has a syntactic label of a clause according to the gold standard parse of the PTB. A verb is any terminal marked by one of the POS tags of type verb according to the gold standard POS tags of the PTB.

5 Results

Our results are shown in Table 1. The left section presents results on English and the right section presents results on Spanish. The top line lists results of the clause detection stage alone. The next two lines list results of the full algorithm (clause detection + collocations) in two different settings of the collocation stage. The bottom line presents the performance of the ALL COUSINS baseline.

In the "Collocation Maximum Precision" setting the parameters of the collocation stage (α and r) were generally tuned such that maximal precision is achieved while preserving a minimal recall level (40% for English, 20% for Spanish on the development data). In the "Collocation Maximum F-score" the collocation parameters were generally tuned such that the maximum possible F-score for the collocation algorithm is achieved.

The best or close to best F-score is achieved when using the clause detection algorithm alone (59.14% for English, 23.34% for Spanish). Note that for both English and Spanish F-score improvements are achieved via a precision improvement that is more significant than the recall degradation. F-score maximization would be the aim of a system that uses the output of our unsupervised ARGID by itself.

The "Collocation Maximum Precision" achieves the best precision level (55.97% for English, 21.8% for Spanish) but at the expense of the largest recall loss. Still, it maintains a reasonable level of recall. The "Collocation Maximum F-score" is an example of a model that provides a precision improvement (over both the baseline and the clause detection stage) with a relatively small recall degradation. In the Spanish experiments its F-score (23.87%) is even a bit higher than that of the clause detection stage (23.34%).

The full two–stage algorithm (clause detection + collocations) should thus be used when we intend to use the model's output as training data for supervised SRL engines or supervised ARGID algorithms.

In our algorithm, the initial set of potential arguments consists of constituents in the Seginer parser's parse tree. Consequently the fraction of arguments that are also constituents (81.87% for English and 51.83% for Spanish) poses an upper bound on our algorithm's recall. Note that the recall of the ALL COUSINS baseline is 74.27% (45.75%) for English (Spanish). This score emphasizes the baseline's strength, and justifies the restriction that the arguments should be k-th cousins of the predicate. The difference between these bounds for the two languages provides a partial explanation for the corresponding gap in the algorithm's performance.

Figure 3 shows the precision of the collocation model (on development data) as a function of the amount of data it was given. We can see that the algorithm reaches saturation at about 5M sentences. It achieves this precision while maintaining a reasonable recall (an average recall of 43.1% after saturation). The parameters of the collocation model were separately tuned for each corpus size, and the graph displays the maximum which was obtained for each of the corpus sizes.

To better understand our model's performance, we performed experiments on the English corpus to test how well its first stage detects clauses. Clause detection is used by our algorithm as a step towards argument identification, but it can be of potential benefit for other purposes as well (see Section 2). The results are 23.88% recall and 40% precision. As in the ARGID task, a random selection of arguments would have yielded an extremely poor result.

6 Conclusion

In this work we presented the first algorithm for argument identification that uses neither supervised syntactic annotation nor SRL tagged data. We have experimented on two languages: English and Spanish. The straightforward adaptability of un-

	English (Test Data)			Spanish (Test Data)		
	Precision	Recall	F1	Precision	Recall	F1
Clause Detection	52.84	67.14	**59.14**	18.00	33.19	23.34
Collocation Maximum F–score	54.11	63.53	58.44	20.22	29.13	**23.87**
Collocation Maximum Precision	**55.97**	40.02	46.67	**21.80**	18.47	20.00
ALL COUSINS baseline	46.71	74.27	57.35	14.16	45.75	21.62

Table 1: Precision, Recall and F1 score for the different stages of our algorithm. Results are given for English (PTB, sentences length bounded by 10, left part of the table) and Spanish (SemEval 2007 Spanish SRL task, right part of the table). The results of the collocation (second) stage are given in two configurations, Collocation Maximum F-score and Collocation Maximum Precision (see text). The upper bounds on Recall, obtained by taking all arguments output by our unsupervised parser, are 81.87% for English and 51.83% for Spanish.

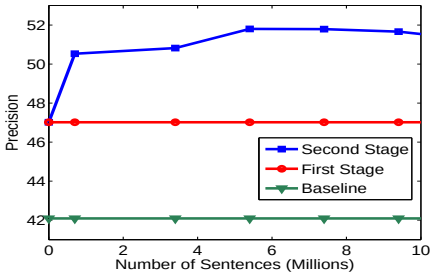


Figure 3: The performance of the second stage on English (squares) vs. corpus size. The precision of the baseline (triangles) and of the first stage (circles) is displayed for reference. The graph indicates the maximum precision obtained for each corpus size. The graph reaches saturation at about 5M sentences. The average recall of the sampled points from there on is 43.1%. Experiments were performed on the English development data.

supervised models to different languages is one of their most appealing characteristics. The recent availability of unsupervised syntactic parsers has offered an opportunity to conduct research on SRL, without reliance on supervised syntactic annotation. This work is the first to address the application of unsupervised parses to an SRL related task.

Our model displayed an increase in precision of 9% in English and 8% in Spanish over a strong baseline. Precision is of particular interest in this context, as instances tagged by high quality annotation could be later used as training data for supervised SRL algorithms. In terms of F–score, our model showed an increase of 1.8% in English and of 2.2% in Spanish over the baseline.

Although the quality of unsupervised parses is currently low (compared to that of supervised approaches), using great amounts of data in identifying recurring structures may reduce noise and in addition address sparsity. The techniques presented in this paper are based on this observation, using around 35M sentences in total for English and 3.3M sentences for Spanish.

As this is the first work which addressed unsupervised ARGID, many questions remain to be explored. Interesting issues to address include assessing the utility of the proposed methods when supervised parses are given, comparing our model to systems with no access to unsupervised parses and conducting evaluation using more relaxed measures.

Unsupervised methods for syntactic tasks have matured substantially in the last few years. Notable examples are (Clark, 2003) for unsupervised POS tagging and (Smith and Eisner, 2006) for unsupervised dependency parsing. Adapting our algorithm to use the output of these models, either to reduce the little supervision our algorithm requires (POS tagging) or to provide complementary syntactic information, is an interesting challenge for future work.

References

Collin F. Baker, Charles J. Fillmore and John B. Lowe, 1998. *The Berkeley FrameNet Project.* ACL-COLING '98.

Daniel M. Bikel, 2004. *Intricacies of Collins' Parsing Model.* Computational Linguistics, 30(4):479–511.

Ted Briscoe, John Carroll, 1997. *Automatic Extraction of Subcategorization from Corpora.* Applied NLP 1997.

Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Pad and Manfred Pinkal, 2006 *The SALSA Corpus: a German Corpus Resource for Lexical Semantics.* LREC '06.

Lou Burnard, 2000. *User Reference Guide for the British National Corpus.* Technical report, Oxford University.

Xavier Carreras and Lluìs Màrquez, 2004. *Introduction to the CoNLL–2004 Shared Task: Semantic Role Labeling.* CoNLL '04.

Xavier Carreras and Lluìs Màrquez, 2005. *Introduction to the CoNLL–2005 Shared Task: Semantic Role Labeling.* CoNLL '05.

Alexander Clark, 2003. *Combining Distributional and Morphological Information for Part of Speech Induction.* EACL '03.

Ronan Collobert and Jason Weston, 2007. *Fast Semantic Extraction Using a Novel Neural Network Architecture.* ACL '07.

Mona Diab, Aous Mansouri, Martha Palmer, Olga Babko-Malaya, Wajdi Zaghouani, Ann Bies and Mohammed Maamouri, 2008. *A pilot Arabic PropBank.* LREC '08.

Evgeniy Gabrilovich and Shaul Markovitch, 2005. *Feature Generation for Text Categorization using World Knowledge.* IJCAI '05.

Daniel Gildea and Daniel Jurafsky, 2002. *Automatic Labeling of Semantic Roles.* Computational Linguistics, 28(3):245–288.

Elliot Glaysher and Dan Moldovan, 2006. *Speeding Up Full Syntactic Parsing by Leveraging Partial Parsing Decisions.* COLING/ACL '06 poster session.

Andrew Gordon and Reid Swanson, 2007. *Generalizing Semantic Role Annotations across Syntactically Similar Verbs.* ACL '07.

David Graff, 1995. *North American News Text Corpus.* Linguistic Data Consortium. LDC95T21.

Trond Grenager and Christopher D. Manning, 2006. *Unsupervised Discovery of a Statistical Verb Lexicon.* EMNLP '06.

Kadri Hacioglu, 2004. *Semantic Role Labeling using Dependency Trees.* COLING '04.

Kadri Hacioglu and Wayne Ward, 2003. *Target Word Detection and Semantic Role Chunking using Support Vector Machines.* HLT-NAACL '03.

Rohit J. Kate and Raymond J. Mooney, 2007. *Semi-Supervised Learning for Semantic Parsing using Support Vector Machines.* HLT–NAACL '07.

Karin Kipper, Hoa Trang Dang and Martha Palmer, 2000. *Class-Based Construction of a Verb Lexicon.* AAAI '00.

Dan Klein, 2005. *The Unsupervised Learning of Natural Language Structure.* Ph.D. thesis, Stanford University.

Anna Korhonen, 2002. *Subcategorization Acquisition.* Ph.D. thesis, University of Cambridge.

Christopher D. Manning, 1993. *Automatic Acquisition of a Large Subcategorization Dictionary.* ACL '93.

Lluìs Màrquez, Xavier Carreras, Kenneth C. Litkowski and Suzanne Stevenson, 2008. *Semantic Role Labeling: An introdution to the Special Issue.* Computational Linguistics, 34(2):145–159

Lluìs Màrquez, Jesus Gimènez Pere Comas and Neus Català, 2005. *Semantic Role Labeling as Sequential Tagging.* CoNLL '05.

Lluìs Màrquez, Lluis Villarejo, M. A. Martì and Mariona Taulè, 2007. *SemEval–2007 Task 09: Multilevel Semantic Annotation of Catalan and Spanish.* The 4th international workshop on Semantic Evaluations (SemEval '07).

Gabriele Musillo and Paula Merlo, 2006. *Accurate Parsing of the proposition bank.* HLT-NAACL '06.

Martha Palmer, Daniel Gildea and Paul Kingsbury, 2005. *The Proposition Bank: A Corpus Annotated with Semantic Roles.* Computational Linguistics, 31(1):71–106.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James H. Martin and Daniel Jurafsky, 2005. *Support Vector Learning for Semantic Argument Classification.* Machine Learning, 60(1):11–39.

Sameer Pradhan, Wayne Ward, James H. Martin, 2008. *Towards Robust Semantic Role Labeling.* Computational Linguistics, 34(2):289–310.

Adwait Ratnaparkhi, 1996. *Maximum Entropy Part-Of-Speech Tagger.* EMNLP '96.

Helmut Schmid, 1994. *Probabilistic Part-of-Speech Tagging Using Decision Trees* International Conference on New Methods in Language Processing.

Yoav Seginer, 2007. *Fast Unsupervised Incremental Parsing.* ACL '07.

Noah A. Smith and Jason Eisner, 2006. *Annealing Structural Bias in Multilingual Weighted Grammar Induction.* ACL '06.

Robert S. Swier and Suzanne Stevenson, 2004. *Unsupervised Semantic Role Labeling.* EMNLP '04.

Robert S. Swier and Suzanne Stevenson, 2005. *Exploiting a Verb Lexicon in Automatic Semantic Role Labelling.* EMNLP '05.

Erik F. Tjong Kim Sang and Hervé Déjean, 2001. *Introduction to the CoNLL-2001 Shared Task: Clause Identification.* CoNLL '01.

Nianwen Xue and Martha Palmer, 2004. *Calibrating Features for Semantic Role Labeling.* EMNLP '04.

Nianwen Xue, 2008. *Labeling Chinese Predicates with Semantic Roles.* Computational Linguistics, 34(2):225–255.

Brutus: A Semantic Role Labeling System Incorporating CCG, CFG, and Dependency Features

Stephen A. Boxwell, Dennis Mehay, and Chris Brew
Department of Linguistics
The Ohio State University
{boxwe11,mehay,cbrew}@1ing.ohio-state.edu

Abstract

We describe a semantic role labeling system that makes primary use of CCG-based features. Most previously developed systems are CFG-based and make extensive use of a treepath feature, which suffers from data sparsity due to its use of explicit tree configurations. CCG affords ways to augment treepath-based features to overcome these data sparsity issues. By adding features over CCG word-word dependencies and lexicalized verbal subcategorization frames ("supertags"), we can obtain an F-score that is substantially better than a previous CCG-based SRL system and competitive with the current state of the art. A manual error analysis reveals that parser errors account for many of the errors of our system. This analysis also suggests that simultaneous incremental parsing and semantic role labeling may lead to performance gains in both tasks.

1 Introduction

Semantic Role Labeling (SRL) is the process of assigning semantic roles to strings of words in a sentence according to their relationship to the semantic predicates expressed in the sentence. The task is difficult because the relationship between syntactic relations like "subject" and "object" do not always correspond to semantic relations like "agent" and "patient". An effective semantic role labeling system must recognize the differences between different configurations:

(a) [The man]$_{Arg0}$ opened [the door]$_{Arg1}$ [for him]$_{Arg3}$ [today]$_{ArgM-TMP}$.

(b) [The door]$_{Arg1}$ opened.

(c) [The door]$_{Arg1}$ was opened by [a man]$_{Arg0}$.

We use Propbank (Palmer et al., 2005), a corpus of newswire text annotated with verb predicate semantic role information that is widely used in the SRL literature (Màrquez et al., 2008). Rather than describe semantic roles in terms of "agent" or "patient", Propbank defines semantic roles on a verb-by-verb basis. For example, the verb *open* encodes the OPENER as Arg0, the OPENEE as Arg1, and the beneficiary of the OPENING action as Arg3. Propbank also defines a set of adjunct roles, denoted by the letter M instead of a number. For example, ArgM-TMP denotes a temporal role, like "today". By using verb-specific roles, Propbank avoids specific claims about parallels between the roles of different verbs.

We follow the approach in (Punyakanok et al., 2008) in framing the SRL problem as a two-stage pipeline: identification followed by labeling. During identification, every word in the sentence is labeled either as bearing some (as yet undetermined) semantic role or not . This is done for each verb. Next, during labeling, the precise verb-specific roles for each word are determined. In contrast to the approach in (Punyakanok et al., 2008), which tags constituents directly, we tag headwords and then associate them with a constituent, as in a previous CCG-based approach (Gildea and Hockenmaier, 2003). Another difference is our choice of parsers. Brutus uses the CCG parser of (Clark and Curran, 2007, henceforth the C&C parser), Charniak's parser (Charniak, 2001) for additional CFG-based features, and MALT parser (Nivre et al., 2007) for dependency features, while (Punyakanok et al., 2008) use results from an ensemble of parses from Charniak's Parser and a Collins parser (Collins, 2003; Bikel, 2004). Finally, the system described in (Punyakanok et al., 2008) uses a joint inference model to resolve discrepancies between multiple automatic parses. We do not employ a similar strategy due to the differing notions of constituency represented in our parsers (CCG having a much more fluid notion of constituency and the MALT parser using a different approach entirely).

For the identification and labeling steps, we train a maximum entropy classifier (Berger et al., 1996) over sections 02-21 of a version of the CCGbank corpus (Hockenmaier and Steedman, 2007) that has been augmented by projecting the Propbank semantic annotations (Boxwell and White, 2008). We evaluate our SRL system's argument predictions at the word string level, making our results directly comparable for each argument labeling.[1]

In the following, we briefly introduce the CCG grammatical formalism and motivate its use in SRL (Sections 2–3). Our main contribution is to demonstrate that CCG — arguably a more expressive and lin-

[1]This is guaranteed by our string-to-string mapping from the original Propbank to the CCGbank.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 37–45,
Suntec, Singapore, 2-7 August 2009.

guistically appealing syntactic framework than vanilla CFGs — is a viable basis for the SRL task. This is supported by our experimental results, the setup and details of which we give in Sections 4–10. In particular, using CCG enables us to map semantic roles directly onto verbal categories, an innovation of our approach that leads to performance gains (Section 7). We conclude with an error analysis (Section 11), which motivates our discussion of future research for computational semantics with CCG (Section 12).

2 Combinatory Categorial Grammar

Combinatory Categorial Grammar (Steedman, 2000) is a grammatical framework that describes syntactic structure in terms of the combinatory potential of the lexical (word-level) items. Rather than using standard part-of-speech tags and grammatical rules, CCG encodes much of the combinatory potential of each word by assigning a syntactically informative category. For example, the verb *loves* has the category (s\np)/np, which could be read "the kind of word that would be a sentence if it could combine with a noun phrase on the right and a noun phrase on the left". Further, CCG has the advantage of a transparent interface between the way the words combine and their dependencies with other words. Word-word dependencies in the CCGbank are encoded using predicate-argument (PARG) relations. PARG relations are defined by the functor word, the argument word, the category of the functor word and which argument slot of the functor category is being filled. For example, in the sentence *John loves Mary* (figure 1), there are two slots on the verbal category to be filled by NP arguments. The first argument (the subject) fills slot 1. This can be encoded as <loves,john,(s\np)/np,1>, indicating the head of the functor, the head of the argument, the functor category and the argument slot. The second argument (the direct object) fills slot 2. This can be encoded as <loves,mary,(s\np)/np,2>. One of the potential advantages to using CCGbank-style PARG relations is that they uniformly encode both local and long-range dependencies — e.g., the noun phrase *the Mary that John loves* expresses the same set of two dependencies. We will show this to be a valuable tool for semantic role prediction.

3 Potential Advantages to using CCG

There are many potential advantages to using the CCG formalism in SRL. One is the uniformity with which CCG can express equivalence classes of local and long-range (including unbounded) dependencies. CFG-based approaches often rely on examining potentially long sequences of categories (or *treepaths*) between the verb and the target word. Because there are a number of different treepaths that correspond to a single relation (figure 2), this approach can suffer from data sparsity. CCG, however, can encode all treepath-distinct expressions of a single grammatical relation into a single predicate-argument relationship (figure 3). This feature has been shown (Gildea and Hockenmaier, 2003) to be an effective substitute for treepath-based features. But while predicate-argument-based features are very effective, they are still vulnerable both to parser errors and to cases where the semantics of a sentence do not correspond directly to syntactic dependencies. To counteract this, we use both kinds of features with the expectation that the treepath feature will provide low-level detail to compensate for missed, incorrect or syntactically impossible dependencies.

Another advantage of a CCG-based approach (and lexicalist approaches in general) is the ability to encode verb-specific argument mappings. An argument mapping is a link between the CCG category and the semantic roles that are likely to go with each of its arguments. The projection of argument mappings onto CCG verbal categories is explored in (Boxwell and White, 2008). We describe this feature in more detail in section 7.

4 Identification and Labeling Models

As in previous approaches to SRL, Brutus uses a two-stage pipeline of maximum entropy classifiers. In addition, we train an argument mapping classifier (described in more detail below) whose predictions are used as features for the labeling model. The same features are extracted for both treebank and automatic parses. Automatic parses were generated using the C&C CCG parser (Clark and Curran, 2007) with its derivation output format converted to resemble that of the CCGbank. This involved following the derivational bracketings of the C&C parser's output and reconstructing the backpointers to the lexical heads using an in-house implementation of the basic CCG combinatory operations. All classifiers were trained to 500 iterations of L-BFGS training — a quasi-Newton method from the numerical optimization literature (Liu and Nocedal, 1989) — using Zhang Le's maxent toolkit.[2] To prevent overfitting we used Gaussian priors with global variances of 1 and 5 for the identifier and labeler, respectively.[3] The Gaussian priors were determined empirically by testing on the development set.

Both the identifier and the labeler use the following features:

(1) **Words**. Words drawn from a 3 word window around the target word,[4] with each word associated with a binary indicator feature.

(2) **Part of Speech**. Part of Speech tags drawn from a 3 word window around the target word,

[2] Available for download at `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`.

[3] Gaussian priors achieve a smoothing effect (to prevent overfitting) by penalizing very large feature weights.

[4] The size of the window was determined experimentally on the development set – we use the same window sizes throughout.

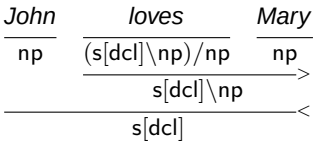


Figure 1: This sentence has two dependencies: <loves,mary,(s\np)/np,2> and <loves,john,(s\np)/np,1>

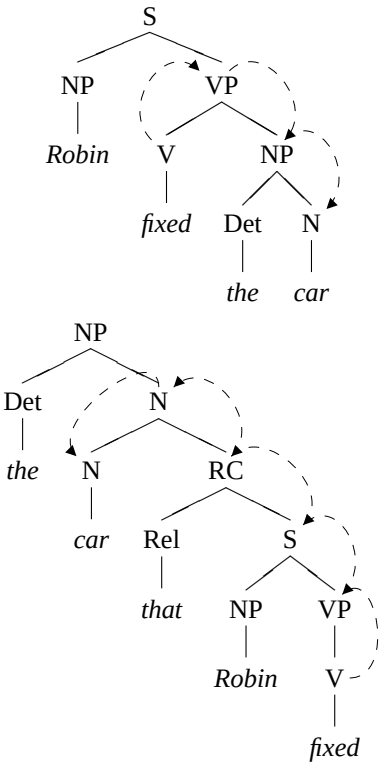


Figure 2: The semantic relation (Arg1) between 'car' and 'fixed' in both phrases is the same, but the treepaths — traced with arrows above — are different: (V>VP<NP<N and V>VP>S>RC>N<N, respectively).

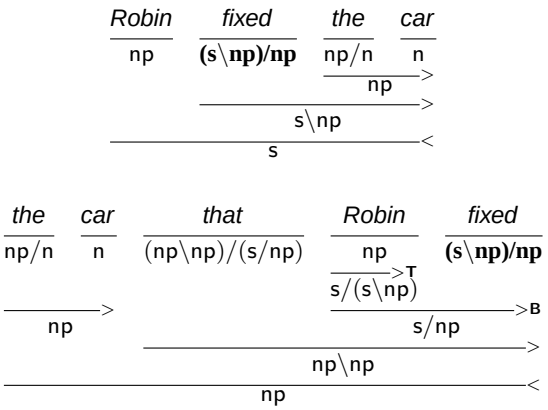


Figure 3: CCG word-word dependencies are passed up through subordinate clauses, encoding the relation between *car* and *fixed* the same in both cases: (s\np)/np.2.→ (Gildea and Hockenmaier, 2003)

with each associated with a binary indicator feature.

(3) **CCG Categories**. CCG categories drawn from a 3 word window around the target word, with each associated with a binary indicator feature.

(4) **Predicate**. The lemma of the predicate we are tagging. E.g. *fix* is the lemma of *fixed*.

(5) **Result Category Detail**. The grammatical feature on the category of the predicate (indicating declarative, passive, progressive, etc). This can be read off the verb category: declarative for *eats*: *(s[dcl]\np)/np* or progressive for *running*: *s[ng]\np*.

(6) **Before/After**. A binary indicator variable indicating whether the target word is before or after the verb.

(7) **Treepath**. The sequence of CCG categories representing the path through the derivation from the predicate to the target word. For the relationship between *fixed* and *car* in the first sentence of figure 3, the treepath is (s[dcl]\np)/np>s[dcl]\np<np<n, with > and < indicating movement up and down the tree, respectively.

(8) **Short Treepath**. Similar to the above treepath feature, except the path stops at the highest node under the least common subsumer that is headed by the target word (this is the *constituent* that the role would be marked on if we identified this terminal as a role-bearing word). Again, for the relationship between *fixed* and *car* in the first sentence of figure 3, the short treepath is (s[dcl]\np)/np>s[dcl]\np<np.

(9) **NP Modified**. A binary indicator feature indicating whether the target word is modified by an NP modifier.[5]

[5]This is easily read off of the CCG PARG relationships.

(10) **Subcategorization**. A sequence of the categories that the verb combines with in the CCG derivation tree. For the first sentence in figure 3, the correct subcategorization would be *np,np*. Notice that this is not necessarily a restatement of the verbal category – in the second sentence of figure 3, the correct subcategorization is *s/(s\np),(np\np)/(s[dcl]/np),np*.

(11) **PARG feature**. We follow a previous CCG-based approach (Gildea and Hockenmaier, 2003) in using a feature to describe the PARG relationship between the two words, if one exists. If there is a dependency in the PARG structure between the two words, then this feature is defined as the conjunction of (1) the category of the functor, (2) the argument slot that is being filled in the functor category, and (3) an indication as to whether the functor (→) or the argument (←) is the lexical head. For example, to indicate the relationship between *car* and *fixed* in both sentences of figure 3, the feature is (s\np)/np.2.→.

The labeler uses all of the previous features, plus the following:

(12) **Headship**. A binary indicator feature as to whether the functor or the argument is the lexical head of the dependency between the two words, if one exists.

(13) **Predicate and Before/After**. The conjunction of two earlier features: the predicate lemma and the Before/After feature.

(14) **Rel Clause**. Whether the path from predicate to target word passes through a relative clause (e.g., marked by the word *'that'* or any other word with a relativizer category).

(15) **PP features**. When the target word is a preposition, we define binary indicator features for the word, POS, and CCG category of the head of the topmost NP in the prepositional phrase headed by a preposition (a.k.a. the '*lexical head*' of the PP). So, if *on* heads the phrase *'on the third Friday'*, then we extract features relating to *Friday* for the preposition *on*. This is null when the target word is not a preposition.

(16) **Argument Mappings**. If there is a PARG relation between the predicate and the target word, the argument mapping is the most likely predicted role to go with that argument. These mappings are predicted using a separate classifier that is trained primarily on lexical information of the verb, its immediate string-level context, and its observed arguments in the training data. This feature is null when there is no PARG relation between the predicate and the target word. The Argument Mapping feature can be viewed as a simple prediction about some of the non-modifier semantic roles that a verb is likely to express. We use this information as a feature and not a hard constraint to allow other features to overrule the recommendation made by the argument mapping classifier. The features used in the argument mapping classifier are described in detail in section 7.

5 CFG based Features

In addition to CCG-based features, features can be drawn from a traditional CFG-style approach when they are available. Our motivation for this is twofold. First, others (Punyakanok et al., 2008, e.g.), have found that different parsers have different error patterns, and so using multiple parsers can yield complementary sources of correct information. Second, we noticed that, although the CCG-based system performed well on head word labeling, performance dropped when projecting these labels to the constituent level (see sections 8 and 9 for more). This may have to do with the fact that CCG is not centered around a constituency-based analysis, as well as with inconsistencies between CCG and Penn Treebank-style bracketings (the latter being what was annotated in the original Propbank).

Penn Treebank-derived features are used in the identifier, labeler, and argument mapping classifiers. For automatic parses, we use Charniak's parser (Charniak, 2001). For gold-standard parses, we remove functional tag and trace information from the Penn Treebank parses before we extract features over them, so as to simulate the conditions of an automatic parse. The Penn Treebank features are as follows:

(17) **CFG Treepath**. A sequence of traditional CFG-style categories representing the path from the verb to the target word.

(18) **CFG Short Treepath**. Analogous to the CCG-based short treepath feature.

(19) **CFG Subcategorization**. Analogous to the CCG-based subcategorization feature.

(20) **CFG Least Common Subsumer**. The category of the root of the smallest tree that dominates both the verb and the target word.

6 Dependency Parser Features

Finally, several features can be extracted from a dependency representation of the same sentence. Automatic dependency relations were produced by the MALT parser. We incorporate MALT into our collection of parses because it provides detailed information on the exact syntactic relations between word pairs (subject, object, adverb, etc) that is not found in other automatic parsers. The features used from the dependency parses are listed below:

(21) **DEP-Exists** A binary indicator feature showing whether or not there is a dependency between the target word and the predicate.

(22) **DEP-Type** If there is a dependency between the target word and the predicate, what type of dependency it is (SUBJ, OBJ, etc).

7 Argument Mapping Model

An innovation in our approach is to use a separate classifier to predict an argument mapping feature. An argument mapping is a mapping from the syntactic arguments of a verbal category to the semantic arguments that should correspond to them (Boxwell and White, 2008). In order to generate examples of the argument mapping for training purposes, it is necessary to employ the PARG relations for a given sentence to identify the headwords of each of the verbal arguments. That is, we use the PARG relations to identify the headwords of each of the constituents that are arguments of the verb. Next, the appropriate semantic role that corresponds to that headword (given by Propbank) is identified. This is done by climbing the CCG derivation tree towards the root until we find a semantic role corresponding to the verb in question — i.e., by finding the point where the constituent headed by the verbal category combines with the constituent headed by the argument in question. These semantic roles are then marked on the corresponding syntactic argument of the verb.

As an example, consider the sentence *The boy loves a girl.* (figure 4). By examining the arguments that the verbal category combines with in the treebank, we can identify the corresponding semantic role for each argument that is marked on the verbal category. We then use these tags to train the Argument Mapping model, which will predict likely argument mappings for verbal categories based on their local surroundings and the headwords of their arguments, similar to the supertagging approaches used to label the informative syntactic categories of the verbs (Bangalore and Joshi, 1999; Clark, 2002), except tagging "one level above" the syntax.

The Argument Mapping Predictor uses the following features:

(23) **Predicate**. The lemma of the predicate, as before.

(24) **Words**. Words drawn from a 5 word window around the target word, with each word associated with a binary indicator feature, as before.

(25) **Parts of Speech**. Part of Speech tags drawn from a 5 word window around the target word, with each tag associated with a binary indicator feature, as before.

(26) **CCG Categories**. CCG categories drawn from a 5 word window around the target word, with each category associated with a binary indicator feature, as before.

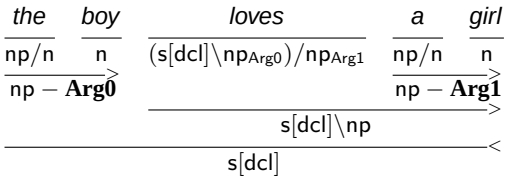


Figure 4: By looking at the constituents that the verb combines with, we can identify the semantic roles corresponding to the arguments marked on the verbal category.

(27) **Argument Data**. The word, POS, and CCG category, and treepath of the headwords of each of the verbal arguments (i.e., PARG dependents), each encoded as a separate binary indicator feature.

(28) **Number of arguments**. The number of arguments marked on the verb.

(29) **Words of Arguments**. The head words of each of the verb's arguments.

(30) **Subcategorization**. The CCG categories that combine with this verb. This includes syntactic adjuncts as well as arguments.

(31) **CFG-Sisters**. The POS categories of the sisters of this predicate in the CFG representation.

(32) **DEP-dependencies**. The individual dependency types of each of the dependencies relating to the verb (SBJ, OBJ, ADV, etc) taken from the dependency parse. We also incorporate a single feature representing the entire set of dependency types associated with this verb into a single feature, representing the set of dependencies as a whole.

Given these features with gold standard parses, our argument mapping model can predict entire argument mappings with an accuracy rate of 87.96% on the test set, and 87.70% on the development set. We found the features generated by this model to be very useful for semantic role prediction, as they enable us to make decisions about entire sets of semantic roles associated with individual lemmas, rather than choosing them independently of each other.

8 Enabling Cross-System Comparison

The Brutus system is designed to label headwords of semantic roles, rather than entire constituents. However, because most SRL systems are designed to label constituents rather than headwords, it is necessary to project the roles up the derivation to the correct constituent in order to make a meaningful comparison of the system's performance. This introduces the potential for further error, so we report results on the accuracy of headwords as well as the correct string of words. We deterministically move the role to the highest constituent in the derivation that is headed by the

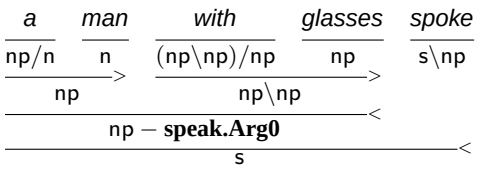


Figure 5: The role is moved towards the root until the original node is no longer the head of the marked constituent.

	P	R	F
G&H (treebank)	67.5%	60.0%	63.5%
Brutus (treebank)	88.18%	85.00%	86.56%
G&H (automatic)	55.7%	49.5%	52.4%
Brutus (automatic)	76.06%	70.15%	72.99%

Table 1: Accuracy of semantic role prediction using only CCG based features.

originally tagged terminal. In most cases, this corresponds to the node immediately dominated by the lowest common subsuming node of the the target word and the verb (figure 5). In some cases, the highest constituent that is headed by the target word is not immediately dominated by the lowest common subsuming node (figure 6).

9 Results

Using a version of Brutus incorporating only the CCG-based features described above, we achieve better results than a previous CCG based system (Gildea and Hockenmaier, 2003, henceforth G&H). This could be due to a number of factors, including the fact that our system employs a different CCG parser, uses a more complete mapping of the Propbank onto the CCGbank, uses a different machine learning approach,[6] and has a richer feature set. The results for constituent tagging accuracy are shown in table 1.

As expected, by incorporating Penn Treebank-based features and dependency features, we obtain better results than with the CCG-only system. The results for gold standard parses are comparable to the winning system of the CoNLL 2005 shared task on semantic role labeling (Punyakanok et al., 2008). Other systems (Toutanova et al., 2008; Surdeanu et al., 2007; Johansson and Nugues, 2008) have also achieved comparable results – we compare our system to (Punyakanok et al., 2008) due to the similarities in our approaches. The performance of the full system is shown in table 2.

Table 3 shows the ability of the system to predict the correct headwords of semantic roles. This is a necessary condition for correctness of the full constituent, but not a sufficient one. In parser evaluation, Carroll, Minnen, and Briscoe (Carroll et al., 2003) have argued

[6] G&H use a generative model with a back-off lattice, whereas we use a maximum entropy classifier.

	P	R	F
P. et al (treebank)	86.22%	87.40%	86.81%
Brutus (treebank)	88.29%	86.39%	87.33%
P. et al (automatic)	77.09%	75.51%	76.29%
Brutus (automatic)	76.73%	70.45%	73.45%

Table 2: Accuracy of semantic role prediction using CCG, CFG, and MALT based features.

	P	R	F
Headword (treebank)	88.94%	86.98%	87.95%
Boundary (treebank)	88.29%	86.39%	87.33%
Headword (automatic)	82.36%	75.97%	79.04%
Boundary (automatic)	76.33%	70.59%	73.35%

Table 3: Accuracy of the system for labeling semantic roles on both constituent boundaries and headwords. Headwords are easier to predict than boundaries, reflecting CCG's focus on word-word relations rather than constituency.

for dependencies as a more appropriate means of evaluation, reflecting the focus on headwords from constituent boundaries. We argue that, especially in the heavily lexicalized CCG framework, headword evaluation is more appropriate, reflecting the emphasis on headword combinatorics in the CCG formalism.

10 The Contribution of the New Features

Two features which are less frequently used in SRL research play a major role in the Brutus system: The PARG feature (Gildea and Hockenmaier, 2003) and the argument mapping feature. Removing them has a strong effect on accuracy when labeling treebank parses, as shown in our feature ablation results in table 4. We do not report results including the Argument Mapping feature but not the PARG feature, because some predicate-argument relation information is assumed in generating the Argument Mapping feature.

	P	R	F
+PARG +AM	88.77%	86.15%	87.44%
+PARG -AM	88.42%	85.78%	87.08%
-PARG -AM	87.92%	84.65%	86.26%

Table 4: The effects of removing key features from the system on gold standard parses.

The same is true for automatic parses, as shown in table 5.

11 Error Analysis

Many of the errors made by the Brutus system can be traced directly to erroneous parses, either in the automatic or treebank parse. In some cases, PP attachment

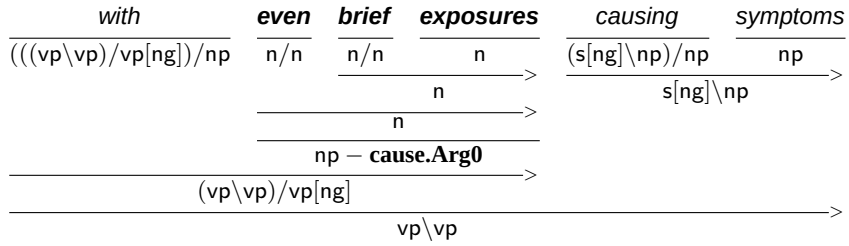


Figure 6: In this case, *with* is the head of *with even brief exposures*, so the role is correctly marked on *even brief exposures* (based on wsj_0003.2).

	P	R	F
+PARG +AM	74.14%	62.09%	67.58%
+PARG -AM	70.02%	64.68%	67.25%
-PARG -AM	73.90%	61.15%	66.93%

Table 5: The effects of removing key features from the system on automatic parses.

ambiguities cause a role to be marked too high in the derivation. In the sentence *the company stopped using asbestos in 1956* (figure 7), the correct Arg1 of *stopped* is *using asbestos*. However, because *in 1956* is erroneously modifying the verb *using* rather than the verb *stopped* in the treebank parse, the system trusts the syntactic analysis and places Arg1 of *stopped* on *using asbestos in 1956*. This particular problem is caused by an annotation error in the original Penn Treebank that was carried through in the conversion to CCGbank.

Another common error deals with genitive constructions. Consider the phrase *a form of asbestos used to make filters*. By CCG combinatorics, the relative clause could either attach to *asbestos* or to *a form of asbestos*. The gold standard CCG parse attaches the relative clause to *a form of asbestos* (figure 8). Propbank agrees with this analysis, assigning Arg1 of *use* to the constituent *a form of asbestos*. The automatic parser, however, attaches the relative clause low – to *asbestos* (figure 9). When the system is given the automatically generated parse, it incorrectly assigns the semantic role to *asbestos*. In cases where the parser attaches the relative clause correctly, the system is much more likely to assign the role correctly.

Problems with relative clause attachment to genitives are not limited to automatic parses – errors in gold-standard treebank parses cause similar problems when Treebank parses disagree with Propbank annotator intuitions. In the phrase *a group of workers exposed to asbestos* (figure 10), the gold standard CCG parse attaches the relative clause to *workers*. Propbank, however, annotates *a group of workers* as Arg1 of *exposed*, rather than following the parse and assigning the role only to *workers*. The system again follows the parse and incorrectly assigns the role to *workers* instead of *a group of workers*. Interestingly, the C&C parser opts for high attachment in this instance, resulting in the correct prediction of *a group of workers* as Arg1 of *exposed* in the automatic parse.

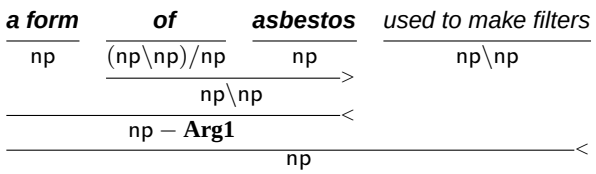


Figure 8: CCGbank gold-standard parse of a relative clause attachment. The system correctly identifies *a form of asbestos* as Arg1 of *used*. (wsj_0003.1)

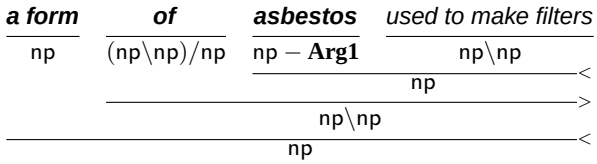


Figure 9: Automatic parse of the noun phrase in figure 8. Incorrect relative clause attachment causes the misidentification of *asbestos* as a semantic role bearing unit. (wsj_0003.1)

12 Future Work

As described in the error analysis section, a large number of errors in the system are attributable to errors in the CCG derivation, either in the gold standard or in automatically generated parses. Potential future work may focus on developing an improved CCG parser using the revised (syntactic) adjunct-argument distinctions (guided by the Propbank annotation) described in (Boxwell and White, 2008). This resource, together with the reasonable accuracy ($\approx$ 90%) with which argument mappings can be predicted, suggests the possibility of an integrated, simultaneous syntactic-semantic parsing process, similar to that of (Musillo and Merlo, 2006; Merlo and Musillo, 2008). We expect this would improve the reliability and accuracy of both the syntactic and semantic analysis components.

13 Acknowledgments

This research was funded by NSF grant IIS-0347799. We are deeply indebted to Julia Hockenmaier for the

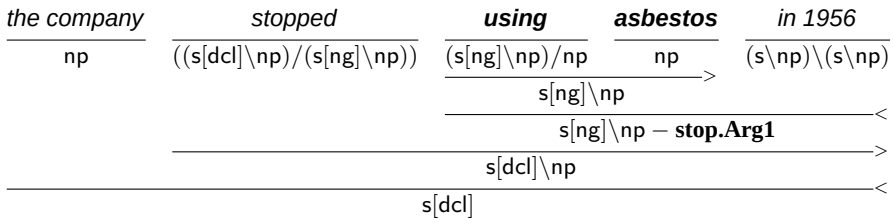


Figure 7: An example of how incorrect PP attachment can cause an incorrect labeling. Stop.Arg1 should cover *using asbestos* rather than *using asbestos in 1956*. This sentence is based on wsj_0003.3, with the structure simplified for clarity.

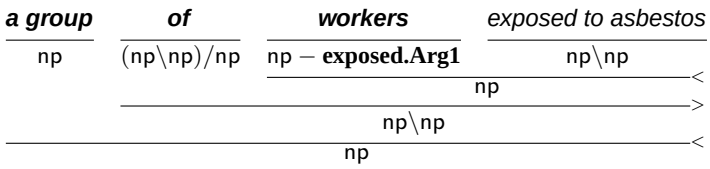


Figure 10: Propbank annotates *a group of workers* as Arg1 of *exposed*, while CCGbank attaches the relative clause low. The system incorrectly labels *workers* as a role bearing unit. (Gold standard – wsj_0003.1)

use of her PARG generation tool.

References

Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Adam L. Berger, S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

D.M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511.

Stephen A. Boxwell and Michael White. 2008. Projecting propbank roles onto the ccgbank. In *Proceedings of the Sixth International Language Resources and Evaluation Conference (LREC-08)*, Marrakech, Morocco.

J. Carroll, G. Minnen, and T. Briscoe. 2003. Parser evaluation. *Treebanks: Building and Using Parsed Corpora*, pages 299–316.

E. Charniak. 2001. Immediate-head parsing for language models. In *Proc. ACL-01*, volume 39, pages 116–123.

Stephen Clark and James R. Curran. 2007. Wide-coverage Efficient Statistical Parsing with CCG and Log-linear Models. *Computational Linguistics*, 33(4):493–552.

Stephen Clark. 2002. Supertagging for combinatory categorial grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+6)*, pages 19–24, Venice, Italy.

M. Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Daniel Gildea and Julia Hockenmaier. 2003. Identifying semantic roles using Combinatory Categorial Grammar. In *Proc. EMNLP-03*.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

R. Johansson and P. Nugues. 2008. Dependency-based syntactic–semantic analysis with PropBank and NomBank. *Proceedings of CoNLL–2008*.

D C Liu and Jorge Nocedal. 1989. On the limited memory method for large scale optimization. *Mathematical Programming B*, 45(3).

Lluís Màrquez, Xavier Carreras, Kenneth C. Litowski, and Suzanne Stevenson. 2008. Semantic Role Labeling: An Introduction to the Special Issue. *Computational Linguistics*, 34(2):145–159.

Paola Merlo and Gabrile Musillo. 2008. Semantic parsing for high-precision semantic role labelling. In *Proceedings of CONLL-08*, Manchester, UK.

Gabriele Musillo and Paola Merlo. 2006. Robust parsing of the proposition bank. In *Proceedings of the EACL 2006 Workshop ROMAND*, Trento.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

Vasin Punyakanok, Dan Roth, and Wen tau Yih. 2008. The Importance of Syntactic Parsing and Inference in Semantic Role Labeling. *Computational Linguistics*, 34(2):257–287.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

M. Surdeanu, L. Màrquez, X. Carreras, and P. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, 29:105–151.

K. Toutanova, A. Haghighi, and C.D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191.

Exploiting Heterogeneous Treebanks for Parsing

Zheng-Yu Niu, Haifeng Wang, Hua Wu
Toshiba (China) Research and Development Center
5/F., Tower W2, Oriental Plaza, Beijing, 100738, China
{niuzhengyu,wanghaifeng,wuhua}@rdc.toshiba.com.cn

Abstract

We address the issue of using heterogeneous treebanks for parsing by breaking it down into two sub-problems, converting grammar formalisms of the treebanks to the same one, and parsing on these homogeneous treebanks. First we propose to employ an iteratively trained target grammar parser to perform grammar formalism conversion, eliminating predefined heuristic rules as required in previous methods. Then we provide two strategies to refine conversion results, and adopt a corpus weighting technique for parsing on homogeneous treebanks. Results on the Penn Treebank show that our conversion method achieves 42% error reduction over the previous best result. Evaluation on the Penn Chinese Treebank indicates that a converted dependency treebank helps constituency parsing and the use of unlabeled data by self-training further increases parsing f-score to 85.2%, resulting in 6% error reduction over the previous best result.

1 Introduction

The last few decades have seen the emergence of multiple treebanks annotated with different grammar formalisms, motivated by the diversity of languages and linguistic theories, which is crucial to the success of statistical parsing (Abeille et al., 2000; Brants et al., 1999; Bohmova et al., 2003; Han et al., 2002; Kurohashi and Nagao, 1998; Marcus et al., 1993; Moreno et al., 2003; Xue et al., 2005). Availability of multiple treebanks creates a scenario where we have a treebank annotated with one grammar formalism, and another treebank annotated with another grammar formalism that we are interested in. We call the first a source treebank, and the second a target treebank. We thus encounter a problem of how to use these heterogeneous treebanks for target grammar parsing. Here heterogeneous treebanks refer to two or more treebanks with different grammar formalisms, e.g., one treebank annotated with dependency structure (DS) and the other annotated with phrase structure (PS).

It is important to acquire additional labeled data for the target grammar parsing through exploitation of existing source treebanks since there is often a shortage of labeled data. However, to our knowledge, there is no previous study on this issue.

Recently there have been some works on using multiple treebanks for domain adaptation of parsers, where these treebanks have the same grammar formalism (McClosky et al., 2006b; Roark and Bacchiani, 2003). Other related works focus on converting one grammar formalism of a treebank to another and then conducting studies on the converted treebank (Collins et al., 1999; Forst, 2003; Wang et al., 1994; Watkinson and Manandhar, 2001). These works were done either on multiple treebanks with the same grammar formalism or on only one converted treebank. We see that their scenarios are different from ours as we work with multiple heterogeneous treebanks.

For the use of heterogeneous treebanks[1], we propose a two-step solution: (1) converting the grammar formalism of the source treebank to the target one, (2) refining converted trees and using them as additional training data to build a target grammar parser.

For grammar formalism conversion, we choose the DS to PS direction for the convenience of the comparison with existing works (Xia and Palmer, 2001; Xia et al., 2008). Specifically, we assume that the source grammar formalism is dependency

[1] Here we assume the existence of two treebanks.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 46–54,
Suntec, Singapore, 2-7 August 2009.

grammar, and the target grammar formalism is phrase structure grammar.

Previous methods for DS to PS conversion (Collins et al., 1999; Covington, 1994; Xia and Palmer, 2001; Xia et al., 2008) often rely on predefined heuristic rules to eliminate converison ambiguity, e.g., minimal projection for dependents, lowest attachment position for dependents, and the selection of conversion rules that add fewer number of nodes to the converted tree. In addition, the validity of these heuristic rules often depends on their target grammars. To eliminate the heuristic rules as required in previous methods, we propose to use an existing target grammar parser (trained on the target treebank) to generate N-best parses for each sentence in the source treebank as conversion candidates, and then select the parse consistent with the structure of the source tree as the converted tree. Furthermore, we attempt to use converted trees as additional training data to retrain the parser for better conversion candidates. The procedure of tree conversion and parser retraining will be run iteratively until a stopping condition is satisfied.

Since some converted trees might be imperfect from the perspective of the target grammar, we provide two strategies to refine conversion results: (1) pruning low-quality trees from the converted treebank, (2) interpolating the scores from the source grammar and the target grammar to select better converted trees. Finally we adopt a corpus weighting technique to get an optimal combination of the converted treebank and the existing target treebank for parser training.

We have evaluated our conversion algorithm on a dependency structure treebank (produced from the Penn Treebank) for comparison with previous work (Xia et al., 2008). We also have investigated our two-step solution on two existing treebanks, the Penn Chinese Treebank (CTB) (Xue et al., 2005) and the Chinese Dependency Treebank (CDT)[2] (Liu et al., 2006). Evaluation on WSJ data demonstrates that it is feasible to use a parser for grammar formalism conversion and the conversion benefits from converted trees used for parser retraining. Our conversion method achieves 93.8% f-score on dependency trees produced from WSJ section 22, resulting in 42% error reduction over the previous best result for DS to PS conversion. Results on CTB show that score interpolation is more effective than instance pruning for the use of converted treebanks for parsing and converted CDT helps parsing on CTB. When coupled with self-training technique, a reranking parser with CTB and converted CDT as labeled data achieves 85.2% f-score on CTB test set, an absolute 1.0% improvement (6% error reduction) over the previous best result for Chinese parsing.

The rest of this paper is organized as follows. In Section 2, we first describe a parser based method for DS to PS conversion, and then we discuss possible strategies to refine conversion results, and finally we adopt the corpus weighting technique for parsing on homogeneous treebanks. Section 3 provides experimental results of grammar formalism conversion on a dependency treebank produced from the Penn Treebank. In Section 4, we evaluate our two-step solution on two existing heterogeneous Chinese treebanks. Section 5 reviews related work and Section 6 concludes this work.

2 Our Two-Step Solution

2.1 Grammar Formalism Conversion

Previous DS to PS conversion methods built a converted tree by iteratively attaching nodes and edges to the tree with the help of conversion rules and heuristic rules, based on current head-dependent pair from a source dependency tree and the structure of the built tree (Collins et al., 1999; Covington, 1994; Xia and Palmer, 2001; Xia et al., 2008). Some observations can be made on these methods: (1) for each head-dependent pair, only one locally optimal conversion was kept during tree-building process, at the risk of pruning globally optimal conversions, (2) heuristic rules are required to deal with the problem that one head-dependent pair might have multiple conversion candidates, and these heuristic rules are usually hand-crafted to reflect the structural preference in their target grammars. To overcome these limitations, we propose to employ a parser to generate N-best parses as conversion candidates and then use the structural information of source trees to select the best parse as a converted tree.

We formulate our conversion method as follows.

Let C_{DS} be a source treebank annotated with DS and C_{PS} be a target treebank annotated with PS. Our goal is to convert the grammar formalism of C_{DS} to that of C_{PS}.

We first train a constituency parser on C_{PS}

[2] Available at http://ir.hit.edu.cn/.

Input: C_{PS}, C_{DS}, Q, and a constituency parser **Output**: Converted trees C_{PS}^{DS}

1. Initialize:
— Set $C_{PS}^{DS,0}$ as null, $DevScore$=0, q=0;
— Split C_{PS} into training set $C_{PS,train}$ and development set $C_{PS,dev}$;
— Train the parser on $C_{PS,train}$ and denote it by P_{q-1};

2. Repeat:
— Use P_{q-1} to generate N-best PS parses for each sentence in C_{DS}, and convert PS to DS for each parse;
— **For** each sentence in C_{DS} **Do**
$\diamond$ $\hat{t}$=$argmax_t Score(x_{i,t})$, and select the $\hat{t}$-th parse as a converted tree for this sentence;
— Let $C_{PS}^{DS,q}$ represent these converted trees, and let C_{train}=$C_{PS,train} \bigcup C_{PS}^{DS,q}$;
— Train the parser on C_{train}, and denote the updated parser by P_q;
— Let $DevScore_q$ be the f-score of P_q on $C_{PS,dev}$;
— **If** $DevScore_q > DevScore$ **Then** $DevScore$=$DevScore_q$, and C_{PS}^{DS}=$C_{PS}^{DS,q}$;
— **Else break**;
— q++;
Until $q > Q$

Table 1: Our algorithm for DS to PS conversion.

(90% trees in C_{PS} as training set $C_{PS,train}$, and other trees as development set $C_{PS,dev}$) and then let the parser generate N-best parses for each sentence in C_{DS}.

Let n be the number of sentences (or trees) in C_{DS} and n_i be the number of N-best parses generated by the parser for the i-th ($1 \leq i \leq n$) sentence in C_{DS}. Let $x_{i,t}$ be the t-th ($1 \leq t \leq n_i$) parse for the i-th sentence. Let y_i be the tree of the i-th ($1 \leq i \leq n$) sentence in C_{DS}.

To evaluate the quality of $x_{i,t}$ as a conversion candidate for y_i, we convert $x_{i,t}$ to a dependency tree (denoted as $x_{i,t}^{DS}$) and then use unlabeled dependency f-score to measure the similarity between $x_{i,t}^{DS}$ and y_i. Let $Score(x_{i,t})$ denote the unlabeled dependency f-score of $x_{i,t}^{DS}$ against y_i. Then we determine the converted tree for y_i by maximizing $Score(x_{i,t})$ over the N-best parses.

The conversion from PS to DS works as follows:

Step 1. Use a head percolation table to find the head of each constituent in $x_{i,t}$.

Step 2. Make the head of each non-head child depend on the head of the head child for each constituent.

Unlabeled dependency f-score is a harmonic mean of unlabeled dependency precision and unlabeled dependency recall. Precision measures how many head-dependent word pairs found in $x_{i,t}^{DS}$ are correct and recall is the percentage of head-dependent word pairs defined in the gold-standard tree that are found in $x_{i,t}^{DS}$. Here we do not take dependency tags into consideration for evaluation since they cannot be obtained without more sophisticated rules.

To improve the quality of N-best parses, we attempt to use the converted trees as additional training data to retrain the parser. The procedure of tree conversion and parser retraining can be run iteratively until a termination condition is satisfied. Here we use the parser's f-score on $C_{PS,dev}$ as a termination criterion. If the update of training data hurts the performance on $C_{PS,dev}$, then we stop the iteration.

Table 1 shows this DS to PS conversion algorithm. Q is an upper limit of the number of loops, and $Q \geq 0$.

2.2 Target Grammar Parsing

Through grammar formalism conversion, we have successfully turned the problem of using heterogeneous treebanks for parsing into the problem of parsing on homogeneous treebanks. Before using converted source treebank for parsing, we present two strategies to refine conversion results.

Instance Pruning For some sentences in C_{DS}, the parser might fail to generate high quality N-best parses, resulting in inferior converted trees. To clean the converted treebank, we can remove the converted trees with low unlabeled dependency f-scores (defined in Section 2.1) before using the converted treebank for parser training

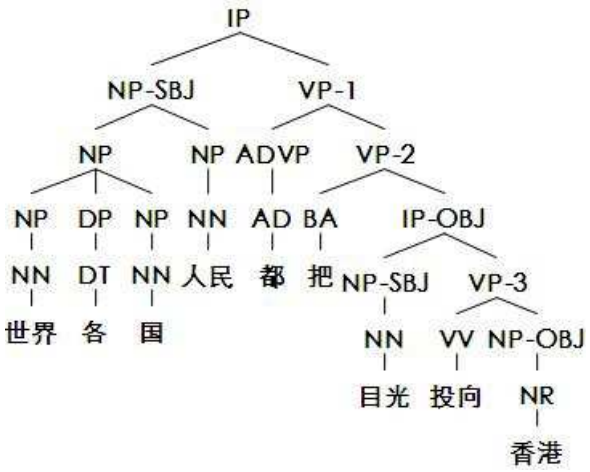


Figure 1: A parse tree in CTB for a sentence of “世界<world> 各<every> 国<country> 人民<people> 都<all> 把<with> 目光<eyes> 投 向<cast> 香 港<Hong Kong>” with “People from all over the world are casting their eyes on Hong Kong” as its English translation.

because these trees are “misleading” training instances. The number of removed trees will be determined by cross validation on development set.

Score Interpolation Unlabeled dependency f-scores used in Section 2.1 measure the quality of converted trees from the perspective of the source grammar only. In extreme cases, the top best parses in the N-best list are good conversion candidates but we might select a parse ranked quite low in the N-best list since there might be conflicts of syntactic structure definition between the source grammar and the target grammar.

Figure 1 shows an example for illustration of a conflict between the grammar of CDT and that of CTB. According to Chinese head percolation tables used in the PS to DS conversion tool “Penn2Malt” [3] and Charniak's parser[4], the head of VP-2 is the word “把” (a preposition, with “BA” as its POS tag in CTB), and the head of IP-OBJ is 投向”. Therefore the word “投向” depends on the word “把”. But according to the annotation scheme in CDT (Liu et al., 2006), the word “把” is a dependent of the word “投向”. The conflicts between the two grammars may lead to the problem that the selected parses based on the information of the source grammar might not be preferred from the perspective of the target grammar.

Therefore we modified the selection metric in Section 2.1 by interpolating two scores, the probability of a conversion candidate from the parser and its unlabeled dependency f-score, shown as follows:

$$\widehat{Score}(x_{i,t}) = \lambda \times Prob(x_{i,t}) + (1-\lambda) \times Score(x_{i,t}). \quad (1)$$

The intuition behind this equation is that converted trees should be preferred from the perspective of both the source grammar and the target grammar. Here $0 \leq \lambda \leq 1$. $Prob(x_{i,t})$ is a probability produced by the parser for $x_{i,t}$ ($0 \leq Prob(x_{i,t}) \leq 1$). The value of λ will be tuned by cross validation on development set.

After grammar formalism conversion, the problem now we face has been limited to how to build parsing models on multiple homogeneous treebank. A possible solution is to simply concatenate the two treebanks as training data. However this method may lead to a problem that if the size of C_{PS} is significantly less than that of converted C_{DS}, converted C_{DS} may weaken the effect C_{PS} might have. One possible solution is to reduce the weight of examples from converted C_{DS} in parser training. Corpus weighting is exactly such an approach, with the weight tuned on development set, that will be used for parsing on homogeneous treebanks in this paper.

3 Experiments of Grammar Formalism Conversion

3.1 Evaluation on WSJ section 22

Xia et al. (2008) used WSJ section 19 from the Penn Treebank to extract DS to PS conversion rules and then produced dependency trees from WSJ section 22 for evaluation of their DS to PS conversion algorithm. They showed that their conversion algorithm outperformed existing methods on the WSJ data. For comparison with their work, we conducted experiments in the same setting as theirs: using WSJ section 19 (1844 sentences) as C_{PS}, producing dependency trees from WSJ section 22 (1700 sentences) as C_{DS}[5], and using labeled bracketing f-scores from the tool “EVALB” on WSJ section 22 for performance evaluation.

[3] Available at http://w3.msi.vxu.se/∼nivre/.

[4] Available at http://www.cs.brown.edu/∼ec/.

[5] We used the tool “Penn2Malt” to produce dependency structures from the Penn Treebank, which was also used for PS to DS conversion in our conversion algorithm.

Models	DevScore (%)	All the sentences LR (%)	LP (%)	F (%)
The best result of Xia et al. (2008)	-	90.7	88.1	89.4
Q-0-method	86.8	92.2	92.8	92.5
Q-10-method	88.0	93.4	94.1	93.8

Table 2: Comparison with the work of Xia et al. (2008) on WSJ section 22.

Models	DevScore (%)	All the sentences LR (%)	LP (%)	F (%)
Q-0-method	91.0	91.6	92.5	92.1
Q-10-method	91.6	93.1	94.1	93.6

Table 3: Results of our algorithm on WSJ section 2∼18 and 20∼22.

We employed Charniak's maximum entropy inspired parser (Charniak, 2000) to generate N-best (N=200) parses. Xia et al. (2008) used POS tag information, dependency structures and dependency tags in test set for conversion. Similarly, we used POS tag information in the test set to restrict search space of the parser for generation of better N-best parses.

We evaluated two variants of our DS to PS conversion algorithm:

Q-0-method: We set the value of Q as 0 for a baseline method.

Q-10-method: We set the value of Q as 10 to see whether it is helpful for conversion to retrain the parser on converted trees.

Table 2 shows the results of our conversion algorithm on WSJ section 22. In the experiment of Q-10-method, DevScore reached the highest value of 88.0% when q was 1. Then we used $C_{PS}^{DS,1}$ as the conversion result. Finally Q-10-method achieved an f-score of 93.8% on WSJ section 22, an absolute 4.4% improvement (42% error reduction) over the best result of Xia et al. (2008). Moreover, Q-10-method outperformed Q-0-method on the same test set. These results indicate that it is feasible to use a parser for DS to PS conversion and the conversion benefits from the use of converted trees for parser retraining.

3.2 Evaluation on WSJ section 2∼18 and 20∼22

In this experiment we evaluated our conversion algorithm on a larger test set, WSJ section 2∼18 and 20∼22 (totally 39688 sentences). Here we also used WSJ section 19 as C_{PS}. Other settings for this experiment are as same as that in Section 3.1, except that here we used a larger test set.

Table 3 provides the f-scores of our method with Q equal to 0 or 10 on WSJ section 2∼18 and 20∼22.

With Q-10-method, DevScore reached the highest value of 91.6% when q was 1. Finally Q-10-method achieved an f-score of 93.6% on WSJ section 2∼18 and 20∼22, better than that of Q-0-method and comparable with that of Q-10-method in Section 3.1. It confirms our previous finding that the conversion benefits from the use of converted trees for parser retraining.

Training data	All the sentences LR (%)	LP (%)	F (%)
$1 \times CTB + CDT^{PS}$	84.7	85.1	84.9
$2 \times CTB + CDT^{PS}$	85.1	85.6	85.3
$5 \times CTB + CDT^{PS}$	85.0	85.5	85.3
$10 \times CTB + CDT^{PS}$	85.3	85.8	85.6
$20 \times CTB + CDT^{PS}$	85.1	85.3	85.2
$50 \times CTB + CDT^{PS}$	84.9	85.3	85.1

Table 4: Results of the generative parser on the development set, when trained with various weighting of CTB training set and CDT^{PS}.

4 Experiments of Parsing

We investigated our two-step solution on two existing treebanks, CDT and CTB, and we used CDT as the source treebank and CTB as the target treebank.

CDT consists of 60k Chinese sentences, annotated with POS tag information and dependency structure information (including 28 POS tags, and 24 dependency tags) (Liu et al., 2006). We did not use POS tag information as inputs to the parser in our conversion method due to the difficulty of conversion from CDT POS tags to CTB POS tags.

We used a standard split of CTB for performance evaluation, articles 1-270 and 400-1151 as training set, articles 301-325 as development set, and articles 271-300 as test set.

We used Charniak's maximum entropy inspired parser and their reranker (Charniak and Johnson, 2005) for target grammar parsing, called a generative parser (GP) and a reranking parser (RP) respectively. We reported ParseVal measures from the EVALB tool.

Models	Training data	All the sentences		
		LR (%)	LP (%)	F (%)
GP	CTB	79.9	82.2	81.0
RP	CTB	82.0	84.6	83.3
GP	$10 \times CTB + CDT^{PS}$	80.4	82.7	81.5
RP	$10 \times CTB + CDT^{PS}$	82.8	84.7	83.8

Table 5: Results of the generative parser (GP) and the reranking parser (RP) on the test set, when trained on only CTB training set or an optimal combination of CTB training set and CDT^{PS}.

4.1 Results of a Baseline Method to Use CDT

We used our conversion algorithm[6] to convert the grammar formalism of CDT to that of CTB. Let CDT^{PS} denote the converted CDT by our method. The average unlabeled dependency f-score of trees in CDT^{PS} was 74.4%, and their average index in 200-best list was 48.

We tried the corpus weighting method when combining CDT^{PS} with CTB training set (abbreviated as CTB for simplicity) as training data, by gradually increasing the weight (including 1, 2, 5, 10, 20, 50) of CTB to optimize parsing performance on the development set. Table 4 presents the results of the generative parser with various weights of CTB on the development set. Considering the performance on the development set, we decided to give CTB a relative weight of 10.

Finally we evaluated two parsing models, the generative parser and the reranking parser, on the test set, with results shown in Table 5. When trained on CTB only, the generative parser and the reranking parser achieved f-scores of 81.0% and 83.3%. The use of CDT^{PS} as additional training data increased f-scores of the two models to 81.5% and 83.8%.

4.2 Results of Two Strategies for a Better Use of CDT

4.2.1 Instance Pruning

We used unlabeled dependency f-score of each converted tree as the criterion to rank trees in CDT^{PS} and then kept only the top M trees with high f-scores as training data for parsing, resulting in a corpus CDT$^{PS}_{M}$. M varied from 100%×|CDT^{PS}| to 10%×|CDT^{PS}| with 10%×|CDT^{PS}| as the interval. |CDT^{PS}| is the number of trees in CDT^{PS}. Then we tuned the value of M by optimizing the parser's performance on the development set with 10×CTB+CDT$^{PS}_{M}$ as training data. Finally the optimal value of M was 100%×|CDT|. It indicates that even removing very few converted trees hurts the parsing performance. A possible reason is that most of non-perfect parses can provide useful syntactic structure information for building parsing models.

Models	Training data	All the sentences		
		LR (%)	LP (%)	F (%)
GP	$CTB + CDT^{PS}_{\lambda}$	81.4	82.8	82.1
RP	$CTB + CDT^{PS}_{\lambda}$	83.0	85.4	84.2

Table 6: Results of the generative parser and the reranking parser on the test set, when trained on an optimal combination of CTB training set and converted CDT.

4.2.2 Score Interpolation

We used $\widehat{Score}(x_{i,t})$[7] to replace $Score(x_{i,t})$ in our conversion algorithm and then ran the updated algorithm on CDT. Let CDT$^{PS}_{\lambda}$ denote the converted CDT by this updated conversion algorithm. The values of λ (varying from 0.0 to 1.0 with 0.1 as the interval) and the CTB weight (including 1, 2, 5, 10, 20, 50) were simultaneously tuned on the development set[8]. Finally we decided that the optimal value of λ was 0.4 and the optimal weight of CTB was 1, which brought the best performance on the development set (an f-score of 86.1%). In comparison with the results in Section 4.1, the average index of converted trees in 200-best list increased to 2, and their average unlabeled dependency f-score dropped to 65.4%. It indicates that structures of converted trees become more consistent with the target grammar, as indicated by the increase of average index of converted trees, further away from the source grammar.

Table 6 provides f-scores of the generative parser and the reranker on the test set, when trained on CTB and CDT$^{PS}_{\lambda}$. We see that the performance of the reranking parser increased to

[6]The setting for our conversion algorithm in this experiment was as same as that in Section 3.1. In addition, we used CTB training set as $C_{PS,train}$, and CTB development set as $C_{PS,dev}$.

[7]Before calculating $\widehat{Score}(x_{i,t})$, we normalized the values of $Prob(x_{i,t})$ for each N-best list by (1) $Prob(x_{i,t})$=$Prob(x_{i,t})$-Min($Prob(x_{i,*})$), (2)$Prob(x_{i,t})$=$Prob(x_{i,t})$/Max($Prob(x_{i,*})$), resulting in that their maximum value was 1 and their minimum value was 0.

[8]Due to space constraint, we do not show f-scores of the parser with different values of λ and the CTB weight.

Models	Training data	All the sentences		
		LR (%)	LP (%)	F (%)
Self-trained GP	$10\times T$+$10\times D$+P	83.0	84.5	83.7
Updated RP	CTB+CDT^{PS}_{λ}	84.3	86.1	85.2

Table 7: Results of the self-trained generative parser and updated reranking parser on the test set. 10×T+10×D+P stands for 10×CTB+10×CDT$^{PS}_{\lambda}$+PDC.

84.2% f-score, better than the result of the reranking parser with CTB and CDT^{PS} as training data (shown in Table 5). It indicates that the use of probability information from the parser for tree conversion helps target grammar parsing.

4.3 Using Unlabeled Data for Parsing

Recent studies on parsing indicate that the use of unlabeled data by self-training can help parsing on the WSJ data, even when labeled data is relatively large (McClosky et al., 2006a; Reichart and Rappoport, 2007). It motivates us to employ self-training technique for Chinese parsing. We used the POS tagged People Daily corpus[9] (Jan. 1998∼Jun. 1998, and Jan. 2000∼Dec. 2000) (PDC) as unlabeled data for parsing. First we removed the sentences with less than 3 words or more than 40 words from PDC to ease parsing, resulting in 820k sentences. Then we ran the reranking parser in Section 4.2.2 on PDC and used the parses on PDC as additional training data for the generative parser. Here we tried the corpus weighting technique for an optimal combination of CTB, CDT$^{PS}_{\lambda}$ and parsed PDC, and chose the relative weight of both CTB and CDT$^{PS}_{\lambda}$ as 10 by cross validation on the development set. Finally we retrained the generative parser on CTB, CDT$^{PS}_{\lambda}$ and parsed PDC. Furthermore, we used this self-trained generative parser as a base parser to retrain the reranker on CTB and CDT$^{PS}_{\lambda}$.

Table 7 shows the performance of self-trained generative parser and updated reranker on the test set, with CTB and CDT$^{PS}_{\lambda}$ as labeled data. We see that the use of unlabeled data by self-training further increased the reranking parser's performance from 84.2% to 85.2%. Our results on Chinese data confirm previous findings on English data shown in (McClosky et al., 2006a; Reichart and Rappoport, 2007).

[9] Available at http://icl.pku.edu.cn/.

4.4 Comparison with Previous Studies for Chinese Parsing

Table 8 and 9 present the results of previous studies on CTB. All the works in Table 8 used CTB articles 1-270 as labeled data. In Table 9, Petrov and Klein (2007) trained their model on CTB articles 1-270 and 400-1151, and Burkett and Klein (2008) used the same CTB articles and parse trees of their English translation (from the English Chinese Translation Treebank) as training data. Comparing our result in Table 6 with that of Petrov and Klein (2007), we see that CDT$^{PS}_{\lambda}$ helps parsing on CTB, which brought 0.9% f-score improvement. Moreover, the use of unlabeled data further boosted the parsing performance to 85.2%, an absolute 1.0% improvement over the previous best result presented in Burkett and Klein (2008).

5 Related Work

Recently there have been some studies addressing how to use treebanks with same grammar formalism for domain adaptation of parsers. Roark and Bachiani (2003) presented count merging and model interpolation techniques for domain adaptation of parsers. They showed that their system with count merging achieved a higher performance when in-domain data was weighted more heavily than out-of-domain data. McClosky et al. (2006b) used self-training and corpus weighting to adapt their parser trained on WSJ corpus to Brown corpus. Their results indicated that both unlabeled in-domain data and labeled out-of-domain data can help domain adaptation. In comparison with these works, we conduct our study in a different setting where we work with multiple heterogeneous treebanks.

Grammar formalism conversion makes it possible to reuse existing source treebanks for the study of target grammar parsing. Wang et al. (1994) employed a parser to help conversion of a treebank from a simple phrase structure to a more informative phrase structure and then used this converted treebank to train their parser. Collins et al. (1999) performed statistical constituency parsing of Czech on a treebank that was converted from the Prague Dependency Treebank under the guidance of conversion rules and heuristic rules, e.g., one level of projection for any category, minimal projection for any dependents, and fixed position of attachment. Xia and Palmer (2001) adopted better heuristic rules to build converted trees, which

Models	≤ 40 words			All the sentences		
	LR (%)	LP (%)	F (%)	LR (%)	LP (%)	F (%)
Bikel & Chiang (2000)	76.8	77.8	77.3	-	-	-
Chiang & Bikel (2002)	78.8	81.1	79.9	-	-	-
Levy & Manning (2003)	79.2	78.4	78.8	-	-	-
Bikel's thesis (2004)	78.0	81.2	79.6	-	-	-
Xiong et. al. (2005)	78.7	80.1	79.4	-	-	-
Chen et. al. (2005)	81.0	81.7	81.2	76.3	79.2	77.7
Wang et. al. (2006)	79.2	81.1	80.1	76.2	78.0	77.1

Table 8: Results of previous studies on CTB with CTB articles 1-270 as labeled data.

Models	≤ 40 words			All the sentences		
	LR (%)	LP (%)	F (%)	LR (%)	LP (%)	F (%)
Petrov & Klein (2007)	85.7	86.9	86.3	81.9	84.8	83.3
Burkett & Klein (2008)	-	-	-	-	-	84.2

Table 9: Results of previous studies on CTB with more labeled data.

reflected the structural preference in their target grammar. For acquisition of better conversion rules, Xia et al. (2008) proposed to automatically extract conversion rules from a target treebank. Moreover, they presented two strategies to solve the problem that there might be multiple conversion rules matching the same input dependency tree pattern: (1) choosing the most frequent rules, (2) preferring rules that add fewer number of nodes and attach the subtree lower.

In comparison with the works of Wang et al. (1994) and Collins et al. (1999), we went further by combining the converted treebank with the existing target treebank for parsing. In comparison with previous conversion methods (Collins et al., 1999; Covington, 1994; Xia and Palmer, 2001; Xia et al., 2008) in which for each head-dependent pair, only one locally optimal conversion was kept during tree-building process, we employed a parser to generate globally optimal syntactic structures, eliminating heuristic rules for conversion. In addition, we used converted trees to retrain the parser for better conversion candidates, while Wang et al. (1994) did not exploit the use of converted trees for parser retraining.

6 Conclusion

We have proposed a two-step solution to deal with the issue of using heterogeneous treebanks for parsing. First we present a parser based method to convert grammar formalisms of the treebanks to the same one, without applying predefined heuristic rules, thus turning the original problem into the problem of parsing on homogeneous treebanks. Then we present two strategies, instance pruning and score interpolation, to refine conversion results. Finally we adopt the corpus weighting technique to combine the converted source treebank with the existing target treebank for parser training.

The study on the WSJ data shows the benefits of our parser based approach for grammar formalism conversion. Moreover, experimental results on the Penn Chinese Treebank indicate that a converted dependency treebank helps constituency parsing, and it is better to exploit probability information produced by the parser through score interpolation than to prune low quality trees for the use of the converted treebank.

Future work includes further investigation of our conversion method for other pairs of grammar formalisms, e.g., from the grammar formalism of the Penn Treebank to more deep linguistic formalism like CCG, HPSG, or LFG.

References

Anne Abeille, Lionel Clement and Francois Toussenel. 2000. Building a Treebank for French. *In Proceedings of LREC 2000*, pages 87-94.

Daniel Bikel and David Chiang. 2000. Two Statistical Parsing Models Applied to the Chinese Treebank. *In Proceedings of the Second SIGHAN workshop*, pages 1-6.

Daniel Bikel. 2004. On the Parameter Space of Generative Lexicalized Statistical Parsing Models. *Ph.D. thesis*, University of Pennsylvania.

Alena Bohmova, Jan Hajic, Eva Hajicova and Barbora Vidova-Hladka. 2003. The Prague Dependency Treebank: A Three-Level Annotation Scenario. *Treebanks:*

Building and Using Annotated Corpora. Kluwer Academic Publishers, pages 103-127.

Thorsten Brants, Wojciech Skut and Hans Uszkoreit. 1999. Syntactic Annotation of a German Newspaper Corpus. *In Proceedings of the ATALA Treebank Workshop*, pages 69-76.

David Burkett and Dan Klein. 2008. Two Languages are Better than One (for Syntactic Parsing). *In Proceedings of EMNLP 2008*, pages 877-886.

Eugene Charniak. 2000. A Maximum Entropy Inspired Parser. *In Proceedings of NAACL 2000*, pages 132-139.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-Fine N-Best Parsing and MaxEnt Discriminative Reranking. *In Proceedings of ACL 2005*, pages 173-180.

Ying Chen, Hongling Sun and Dan Jurafsky. 2005. A Corrigendum to Sun and Jurafsky (2004) Shallow Semantic Parsing of Chinese. *University of Colorado at Boulder CSLR Tech Report TR-CSLR-2005-01*.

David Chiang and Daniel M. Bikel. 2002. Recovering Latent Information in Treebanks. *In Proceedings of COLING 2002*, pages 1-7.

Micheal Collins, Lance Ramshaw, Jan Hajic and Christoph Tillmann. 1999. A Statistical Parser for Czech. *In Proceedings of ACL 1999*, pages 505-512.

Micheal Covington. 1994. GB Theory as Dependency Grammar. *Research Report AI-1992-03*.

Martin Forst. 2003. Treebank Conversion - Establishing a Testsuite for a Broad-Coverage LFG from the TIGER Treebank. *In Proceedings of LINC at EACL 2003*, pages 25-32.

Chunghye Han, Narae Han, Eonsuk Ko and Martha Palmer. 2002. Development and Evaluation of a Korean Treebank and its Application to NLP. *In Proceedings of LREC 2002*, pages 1635-1642.

Sadao Kurohashi and Makato Nagao. 1998. Building a Japanese Parsed Corpus While Improving the Parsing System. *In Proceedings of LREC 1998*, pages 719-724.

Roger Levy and Christopher Manning. 2003. Is It Harder to Parse Chinese, or the Chinese Treebank? *In Proceedings of ACL 2003*, pages 439-446.

Ting Liu, Jinshan Ma and Sheng Li. 2006. Building a Dependency Treebank for Improving Chinese Parser. *Journal of Chinese Language and Computing*, 16(4):207-224.

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

David McClosky, Eugene Charniak and Mark Johnson. 2006a. Effective Self-Training for Parsing. *In Proceedings of NAACL 2006*, pages 152-159.

David McClosky, Eugene Charniak and Mark Johnson. 2006b. Reranking and Self-Training for Parser Adaptation. *In Proceedings of COLING/ACL 2006*, pages 337-344.

Antonio Moreno, Susana Lopez, Fernando Sanchez and Ralph Grishman. 2003. Developing a Syntactic Annotation Scheme and Tools for a Spanish Treebank. *Treebanks: Building and Using Annotated Corpora. Kluwer Academic Publishers*, pages 149-163.

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

Roi Reichart and Ari Rappoport. 2007. Self-Training for Enhancement and Domain Adaptation of Statistical Parsers Trained on Small Datasets. *In Proceedings of ACL 2007*, pages 616-623.

Brian Roark and Michiel Bacchiani. 2003. Supervised and Unsupervised PCFG Adaptation to Novel Domains. *In Proceedings of HLT/NAACL 2003*, pages 126-133.

Jong-Nae Wang, Jing-Shin Chang and Keh-Yih Su. 1994. An Automatic Treebank Conversion Algorithm for Corpus Sharing. *In Proceedings of ACL 1994*, pages 248-254.

Mengqiu Wang, Kenji Sagae and Teruko Mitamura. 2006. A Fast, Accurate Deterministic Parser for Chinese. *In Proceedings of COLING/ACL 2006*, pages 425-432.

Stephen Watkinson and Suresh Manandhar. 2001. Translating Treebank Annotation for Evaluation. *In Proceedings of ACL Workshop on Evaluation Methodologies for Language and Dialogue Systems*, pages 1-8.

Fei Xia and Martha Palmer. 2001. Converting Dependency Structures to Phrase Structures. *In Proceedings of HLT 2001*, pages 1-5.

Fei Xia, Rajesh Bhatt, Owen Rambow, Martha Palmer and Dipti Misra. Sharma. 2008. Towards a Multi-Representational Treebank. *In Proceedings of the 7th International Workshop on Treebanks and Linguistic Theories*, pages 159-170.

Deyi Xiong, Shuanglong Li, Qun Liu, Shouxun Lin and Yueliang Qian. 2005. Parsing the Penn Chinese Treebank with Semantic Knowledge. *In Proceedings of IJCNLP 2005*, pages 70-81.

Nianwen Xue, Fei Xia, Fu-Dong Chiou and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase Structure Annotation of a Large Corpus. *Natural Language Engineering*, 11(2):207-238.

Cross Language Dependency Parsing using a Bilingual Lexicon*

Hai Zhao(赵海)†‡, Yan Song(宋彦)†, Chunyu Kit†, Guodong Zhou‡
†Department of Chinese, Translation and Linguistics
City University of Hong Kong
83 Tat Chee Avenue, Kowloon, Hong Kong, China
‡School of Computer Science and Technology
Soochow University, Suzhou, China 215006
{haizhao,yansong,ctckit}@cityu.edu.hk, gdzhou@suda.edu.cn

Abstract

This paper proposes an approach to enhance dependency parsing in a language by using a translated treebank from another language. A simple statistical machine translation method, word-by-word decoding, where not a parallel corpus but a bilingual lexicon is necessary, is adopted for the treebank translation. Using an ensemble method, the key information extracted from word pairs with dependency relations in the translated text is effectively integrated into the parser for the target language. The proposed method is evaluated in English and Chinese treebanks. It is shown that a translated English treebank helps a Chinese parser obtain a state-of-the-art result.

1 Introduction

Although supervised learning methods bring state-of-the-art outcome for dependency parser inferring (McDonald et al., 2005; Hall et al., 2007), a large enough data set is often required for specific parsing accuracy according to this type of methods. However, to annotate syntactic structure, either phrase- or dependency-based, is a costly job. Until now, the largest treebanks[1] in various languages for syntax learning are with around one million words (or some other similar units). Limited data stand in the way of further performance enhancement. This is the case for each individual language at least. But, this is not the case as we observe all treebanks in different languages as a whole. For example, of ten treebanks for CoNLL-2007 shared task, none includes more than 500K tokens, while the sum of tokens from all treebanks is about two million (Nivre et al., 2007).

As different human languages or treebanks should share something common, this makes it possible to let dependency parsing in multiple languages be beneficial with each other. In this paper, we study how to improve dependency parsing by using (automatically) translated texts attached with transformed dependency information. As a case study, we consider how to enhance a Chinese dependency parser by using a translated English treebank. What our method relies on is not the close relation of the chosen language pair but the similarity of two treebanks, this is the most different from the previous work.

Two main obstacles are supposed to confront in a cross-language dependency parsing task. The first is the cost of translation. Machine translation has been shown one of the most expensive language processing tasks, as a great deal of time and space is required to perform this task. In addition, a standard statistical machine translation method based on a parallel corpus will not work effectively if it is not able to find a parallel corpus that right covers source and target treebanks. However, dependency parsing focuses on the relations of word pairs, this allows us to use a dictionary-based translation without assuming a parallel corpus available, and the training stage of translation may be ignored and the decoding will be quite fast in this case. The second difficulty is that the outputs of translation are hardly qualified for the parsing purpose. The most challenge in this aspect is morphological preprocessing. We regard that the morphological issue should be handled aiming at the specific language, our solution here is to use character-level features for a target language like Chinese.

The rest of the paper is organized as follows. The next section presents some related existing work. Section 3 describes the procedure on tree-

The study is partially supported by City University of Hong Kong through the Strategic Research Grant 7002037 and 7002388. The first author is sponsored by a research fellowship from CTL, City University of Hong Kong.

[1] It is a tradition to call an annotated syntactic corpus as treebank in parsing community.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 55–63,
Suntec, Singapore, 2-7 August 2009.

bank translation and dependency transformation. Section 4 describes a dependency parser for Chinese as a baseline. Section 5 describes how a parser can be strengthened from the translated treebank. The experimental results are reported in Section 6. Section 7 looks into a few issues concerning the conditions that the proposed approach is suitable for. Section 8 concludes the paper.

2 The Related Work

As this work is about exploiting extra resources to enhance an existing parser, it is related to domain adaption for parsing that has been draw some interests in recent years. Typical domain adaptation tasks often assume annotated data in new domain absent or insufficient and a large scale unlabeled data available. As unlabeled data are concerned, semi-supervised or unsupervised methods will be naturally adopted. In previous works, two basic types of methods can be identified to enhance an existing parser from additional resources. The first is usually focus on exploiting automatic generated labeled data from the unlabeled data (Steedman et al., 2003; McClosky et al., 2006; Reichart and Rappoport, 2007; Sagae and Tsujii, 2007; Chen et al., 2008), the second is on combining supervised and unsupervised methods, and only unlabeled data are considered (Smith and Eisner, 2006; Wang and Schuurmans, 2008; Koo et al., 2008).

Our purpose in this study is to obtain a further performance enhancement by exploiting treebanks in other languages. This is similar to the above first type of methods, some assistant data should be automatically generated for the subsequent processing. The differences are what type of data are concerned with and how they are produced. In our method, a machine translation method is applied to tackle golden-standard treebank, while all the previous works focus on the unlabeled data.

Although cross-language technique has been used in other natural language processing tasks, it is basically new for syntactic parsing as few works were concerned with this issue. The reason is straightforward, syntactic structure is too complicated to be properly translated and the cost of translation cannot be afforded in many cases. However, we empirically find this difficulty may be dramatically alleviated as dependencies rather than phrases are used for syntactic structure representation. Even the translation outputs are not so good as the expected, a dependency parser for the target language can effectively make use of them by only considering the most related information extracted from the translated text.

The basic idea to support this work is to make use of the semantic connection between different languages. In this sense, it is related to the work of (Merlo et al., 2002) and (Burkett and Klein, 2008). The former showed that complementary information about English verbs can be extracted from their translations in a second language (Chinese) and the use of multilingual features improves classification performance of the English verbs. The latter iteratively trained a model to maximize the marginal likelihood of tree pairs, with alignments treated as latent variables, and then jointly parsing bilingual sentences in a translation pair. The proposed parser using features from monolingual and mutual constraints helped its log-linear model to achieve better performance for both monolingual parsers and machine translation system. In this work, cross-language features will be also adopted as the latter work. However, although it is not essentially different, we only focus on dependency parsing itself, while the parsing scheme in (Burkett and Klein, 2008) based on a constituent representation.

Among of existing works that we are aware of, we regard that the most similar one to ours is (Zeman and Resnik, 2008), who adapted a parser to a new language that is much poorer in linguistic resources than the source language. However, there are two main differences between their work and ours. The first is that they considered a pair of sufficiently related languages, Danish and Swedish, and made full use of the similar characteristics of two languages. Here we consider two quite different languages, English and Chinese. As fewer language properties are concerned, our approach holds the more possibility to be extended to other language pairs than theirs. The second is that a parallel corpus is required for their work and a strict statistical machine translation procedure was performed, while our approach holds a merit of simplicity as only a bilingual lexicon is required.

3 Treebank Translation and Dependency Transformation

3.1 Data

As a case study, this work will be conducted between the source language, English, and the target language, Chinese, namely, we will investigate

how a translated English treebank enhances a Chinese dependency parser.

For English data, the Penn Treebank (PTB) 3 is used. The constituency structures is converted to dependency trees by using the same rules as (Yamada and Matsumoto, 2003) and the standard training/development/test split is used. However, only training corpus (sections 2-21) is used for this study. For Chinese data, the Chinese Treebank (CTB) version 4.0 is used in our experiments. The same rules for conversion and the same data split is adopted as (Wang et al., 2007): files 1-270 and 400-931 as training, 271-300 as testing and files 301-325 as development. We use the gold standard segmentation and part-of-speech (POS) tags in both treebanks.

As a bilingual lexicon is required for our task and none of existing lexicons are suitable for translating PTB, two lexicons, LDC Chinese-English Translation Lexicon Version 2.0 (LDC2002L27), and an English to Chinese lexicon in StarDict[2], are conflated, with some necessary manual extensions, to cover 99% words appearing in the PTB (the most part of the untranslated words are named entities.). This lexicon includes 123K entries.

3.2 Translation

A word-by-word statistical machine translation strategy is adopted to translate words attached with the respective dependency information from the source language to the target one. In detail, a word-based decoding is used, which adopts a log-linear framework as in (Och and Ney, 2002) with only two features, translation model and language model,

$$P(c|e) = \frac{\exp[\sum_{i=1}^{2} \lambda_i h_i(c,e)]}{\sum_c \exp[\sum_{i=1}^{2} \lambda_i h_i(c,e)]}$$

Where

$$h_1(c,e) = \log(p_\gamma(c|e))$$

is the translation model, which is converted from the bilingual lexicon, and

$$h_2(c,e) = \log(p_\theta(c))$$

is the language model, a word trigram model trained from the CTB. In our experiment, we set two weights $\lambda_1 = \lambda_2 = 1$.

[2]StarDict is an open source dictionary software, available at http://stardict.sourceforge.net/.

The conversion process of the source treebank is completed by three steps as the following:
1. Bind POS tag and dependency relation of a word with itself;
2. Translate the PTB text into Chinese word by word. Since we use a lexicon rather than a parallel corpus to estimate the translation probabilities, we simply assign uniform probabilities to all translation options. Thus the decoding process is actually only determined by the language model. Similar to the "bag translation" experiment in (Brown et al., 1990), the candidate target sentences made up by a sequence of the optional target words are ranked by the trigram language model. The output sentence will be generated only if it is with maximum probability as follows,

$$\begin{aligned} c &= \text{argmax}\{p_\theta(c)p_\gamma(c|e)\} \\ &= \text{argmax}\ p_\theta(c) \\ &= \text{argmax}\prod p_\theta(w_c) \end{aligned}$$

A beam search algorithm is used for this process to find the best path from all the translation options; As the training stage, especially, the most time-consuming alignment sub-stage, is skipped, the translation only includes a decoding procedure that takes about 4.5 hours for about one million words of the PTB in a 2.8GHz PC.
3. After the target sentence is generated, the attached POS tags and dependency information of each English word will also be transferred to each corresponding Chinese word. As word order is often changed after translation, the pointer of each dependency relationship, represented by a serial number, should be re-calculated.

Although we try to perform an exact word-by-word translation, this aim cannot be fully reached in fact, as the following case is frequently encountered, multiple English words have to be translated into one Chinese word. To solve this problem, we use a policy that lets the output Chinese word only inherits the attached information of the highest syntactic head in the original multiple English words.

4 Dependency Parsing: Baseline

4.1 Learning Model and Features

According to (McDonald and Nivre, 2007), all data-driven models for dependency parsing that have been proposed in recent years can be described as either graph-based or transition-based.

Table 1: Feature Notations

Notation	Meaning
s	The word in the top of stack
s'	The first word below the top of stack.
s_{-1},s_1...	The first word before(after) the word in the top of stack.
i, i_{+1},...	The first (second) word in the unprocessed sequence, etc.
dir	Dependent direction
h	Head
lm	Leftmost child
rm	Rightmost child
rn	Right nearest child
form	word form
pos	POS tag of word
cpos1	coarse POS: the first letter of POS tag of word
cpos2	coarse POS: the first two POS tags of word
lnverb	the left nearest verb
$char_1$	The first character of a word
$char_2$	The first two characters of a word
$char_{-1}$	The last character of a word
$char_{-2}$	The last two characters of a word
.	's, i.e., '*s.dprel*' means dependent label of character in the top of stack
+	Feature combination, i.e., '*s.char+i.char*' means both *s.char* and *i.char* work as a feature function.

Although the former will be also used as comparison, the latter is chosen as the main parsing framework by this study for the sake of efficiency. In detail, a shift-reduce method is adopted as in (Nivre, 2003), where a classifier is used to make a parsing decision step by step. In each step, the classifier checks a word pair, namely, *s*, the top of a stack that consists of the processed words, and, *i*, the first word in the (input) unprocessed sequence, to determine if a dependent relation should be established between them. Besides two dependency arc building actions, a shift action and a reduce action are also defined to maintain the stack and the unprocessed sequence. In this work, we adopt a left-to-right arc-eager parsing model, that means that the parser scans the input sequence from left to right and right dependents are attached to their heads as soon as possible (Hall et al., 2007).

While memory-based and margin-based learning approaches such as support vector machines are popularly applied to shift-reduce parsing, we apply maximum entropy model as the learning model for efficient training and adopting overlapped features as our work in (Zhao and Kit, 2008), especially, those character-level ones for Chinese parsing. Our implementation of maximum entropy adopts L-BFGS algorithm for parameter optimization as usual.

With notations defined in Table 1, a feature set as shown in Table 2 is adopted. Here, we explain some terms in Tables 1 and 2. We used a large scale feature selection approach as in (Zhao et al., 2009) to obtain the feature set in Table 2. Some feature notations in this paper are also borrowed from that work.

The feature *curroot* returns the root of a partial parsing tree that includes a specified node. The feature *charseq* returns a character sequence whose members are collected from all identified children for a specified word.

In Table 2, as for concatenating multiple substrings into a feature string, there are two ways, seq and bag. The former is to concatenate all substrings without do something special. The latter will remove all duplicated substrings, sort the rest and concatenate all at last.

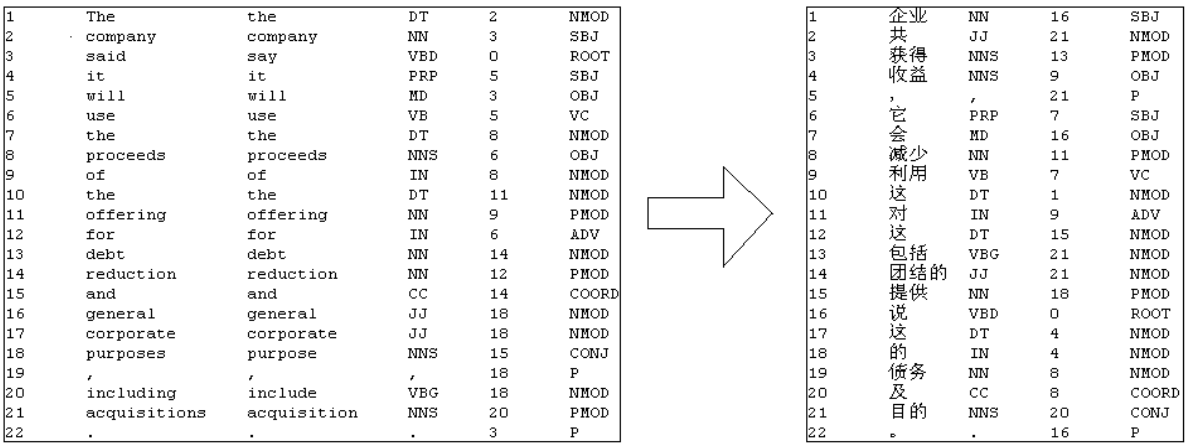
Note that we systemically use a group of character-level features. Surprisingly, as to our best knowledge, this is the first report on using this type of features in Chinese dependency parsing. Although (McDonald et al., 2005) used the prefix of each word form instead of word form itself as features, character-level features here for Chinese is essentially different from that. As Chinese is basically a character-based written language. Character plays an important role in many means, most characters can be formed as single-character words, and Chinese itself is character-order free rather than word-order free to some extent. In addition, there is often a close connection between the meaning of a Chinese word and its first or last character.

4.2 Parsing using a Beam Search Algorithm

In Table 2, the feature $preact_n$ returns the previous parsing action type, and the subscript n stands for the action order before the current action. These are a group of Markovian features. Without this type of features, a shift-reduce parser may directly scan through an input sequence in linear time. Otherwise, following the work of (Duan et al., 2007) and (Zhao, 2009), the parsing algorithm is to search a parsing action sequence with the maximal probability.

$$S_{d_i} = \text{argmax} \prod_i p(d_i|d_{i-1}d_{i-2}...),$$

where S_{d_i} is the object parsing action sequence, $p(d_i|d_{i-1}...)$ is the conditional probability, and d_i



1	The	the	DT	2	NMOD
2	company	company	NN	3	SBJ
3	said	say	VBD	0	ROOT
4	it	it	PRP	5	SBJ
5	will	will	MD	3	OBJ
6	use	use	VB	5	VC
7	the	the	DT	8	NMOD
8	proceeds	proceeds	NNS	6	OBJ
9	of	of	IN	8	NMOD
10	the	the	DT	11	NMOD
11	offering	offering	NN	9	PMOD
12	for	for	IN	6	ADV
13	debt	debt	NN	14	NMOD
14	reduction	reduction	NN	12	PMOD
15	and	and	CC	14	COORD
16	general	general	JJ	18	NMOD
17	corporate	corporate	JJ	18	NMOD
18	purposes	purpose	NNS	15	CONJ
19	,	,	,	18	P
20	including	include	VBG	18	NMOD
21	acquisitions	acquisition	NNS	20	PMOD
22	.	.	.	3	P

1	企业	NN	16	SBJ
2	共	JJ	21	NMOD
3	获得	NNS	13	PMOD
4	收益	NNS	9	OBJ
5	,	,	21	P
6	它	PRP	7	SBJ
7	会	MD	16	OBJ
8	减少	NN	11	PMOD
9	利用	VB	7	VC
10	这	DT	1	NMOD
11	对	IN	9	ADV
12	这	DT	15	NMOD
13	包括	VBG	21	NMOD
14	团结的	JJ	21	NMOD
15	提供	NN	18	PMOD
16	说	VBD	0	ROOT
17	这	DT	4	NMOD
18	的	IN	4	NMOD
19	债务	NN	8	NMOD
20	及	CC	8	COORD
21	目的	NNS	20	CONJ
22	。	.	16	P

Figure 1: A comparison before and after translation

Table 2: Features for Parsing

_	$i_n.form, n = 0, 1$
_	$i.form + i_1.form$
_	$i_n.char_2 + i_{n+1}.char_2, n = -1, 0$
_	$i.char_{-1} + i_1.char_{-1}$
_	$i_n.char_{-2}\ n = 0, 3$
_	$i_1.char_{-2} + i_2.char_{-2} + i_3.char_{-2}$
_	$i.lnverb.char_{-2}$
_	$i_3.pos$
_	$i_n.pos + i_{n+1}.pos, n = 0, 1$
_	$i_{-2}.cpos1 + i_{-1}.cpos1$
_	$i_1.cpos1 + i_2.cpos1 + i_3.cpos1$
_	$s'_2.char_1$
_	$s'.char_{-2} + s'_1.char_{-2}$
_	$s'_{-2}.cpos2$
_	$s'_{-1}.cpos2 + s'_1.cpos2$
_	$s'.cpos2 + s'_1.cpos2$
_	*s'.children.cpos2.seq*
_	*s'.children.dprel.seq*
_	*s'.subtree.depth*
_	$s'.h.form + s'.rm.cpos1$
_	$s'.lm.char_2 + s'.char_2$
_	*s.h.children.dprel.seq*
_	*s.lm.dprel*
_	$s.char_{-2} + i_1.char_{-2}$
_	$s.char_n + i.char_n, n = -1, 1$
_	$s_{-1}.pos + i_1.pos$
_	$s.pos + i_n.pos, n = -1, 0, 1$
_	$s : i|linePath.form.bag$
_	$s'.form + i.form$
_	$s'.char_2 + i_n.char_2, n = -1, 0, 1$
_	$s.curroot.pos + i.pos$
_	$s.curroot.char_2 + i.char_2$
_	$s.children.cpos2.seq + i.children.cpos2.seq$
_	$s.children.cpos2.seq + i.children.cpos2.seq + s.cpos2 + i.cpos2$
_	$s'.children.dprel.seq + i.children.dprel.seq$
_	$preact_{-1}$
_	$preact_{-2}$
_	$preact_{-2}$+$preact_{-1}$

is i-th parsing action. We use a beam search algorithm to find the object parsing action sequence.

5 Exploiting the Translated Treebank

As we cannot expect too much for a word-by-word translation, only word pairs with dependency relation in translated text are extracted as useful and reliable information. Then some features based on a query in these word pairs according to the current parsing state (namely, words in the current stack and input) will be derived to enhance the Chinese parser.

A translation sample can be seen in Figure 1. Although most words are satisfactorily translated, to generate effective features, what we still have to consider at first is the inconsistence between the translated text and the target text.

In Chinese, word lemma is always its word form itself, this is a convenient characteristic in computational linguistics and makes lemma features unnecessary for Chinese parsing at all. However, Chinese has a special primary processing task, i.e., word segmentation. Unfortunately, word definitions for Chinese are not consistent in various linguistical views, for example, seven segmentation conventions for computational purpose are formally proposed since the first Bakeoff[3].

Note that CTB or any other Chinese treebank has its own word segmentation guideline. Chinese word should be strictly segmented according to the guideline before POS tags and dependency relations are annotated. However, as we say the

[3]Bakeoff is a Chinese processing share task held by SIGHAN.

English treebank is translated into Chinese word by word, Chinese words in the translated text are exactly some entries from the bilingual lexicon, they are actually irregular phrases, short sentences or something else rather than words that follows any existing word segmentation convention. If the bilingual lexicon is not carefully selected or refined according to the treebank where the Chinese parser is trained from, then there will be a serious inconsistence on word segmentation conventions between the translated and the target treebanks.

As all concerned feature values here are calculated from the searching result in the translated word pair list according to the current parsing state, and a complete and exact match cannot be always expected, our solution to the above segmentation issue is using a partial matching strategy based on characters that the words include.

Above all, a translated word pair list, L, is extracted from the translated treebank. Each item in the list consists of three elements, dependant word (dp), head word (hd) and the frequency of this pair in the translated treebank, f.

There are two basic strategies to organize the features derived from the translated word pair list. The first is to find the most matching word pair in the list and extract some properties from it, such as the matched length, part-of-speech tags and so on, to generate features. Note that a matching priority serial should be defined aforehand in this case. The second is to check every matching models between the current parsing state and the partially matched word pair. In an early version of our approach, the former was implemented. However, It is proven to be quite inefficient in computation. Thus we adopt the second strategy at last. Two matching model feature functions, $\phi(\cdot)$ and $\psi(\cdot)$, are correspondingly defined as follows. The return value of $\phi(\cdot)$ or $\psi(\cdot)$ is the logarithmic frequency of the matched item. There are four input parameters required by the function $\phi(\cdot)$. Two parameters of them are about which part of the stack(input) words is chosen, and other two are about which part of each item in the translated word pair is chosen. These parameters could be set to $full$ or $char_n$ as shown in Table 1, where $n = ..., -2, -1, 1, 2, ...$ For example, a possible feature could be $\phi(s.full, i.char_1, dp.full, hd.char_1)$, it tries to find a match in L by comparing stack word and dp word, and the first character of input word and the first character of hd word. If such a match item in L is found, then $\phi(\cdot)$ returns $\log(f)$. There are three input parameters required by the function $\psi(\cdot)$. One parameter is about which part of the stack(input) words is chosen, and the other is about which part of each item in the translated word pair is chosen. The third is about the matching type that may be set to $dependant$, $head$, or $root$. For example, the function $\psi(i.char_1, hd.full, root)$ tries to find a match in L by comparing the first character of input word and the whole dp word. If such a match item in L is found, then $\psi(\cdot)$ returns $\log(f)$ as hd occurs as ROOT f times.

Table 3: Features based on the translated treebank

-	$\phi(i.char_3, s'.full, dp.char_3, hd.full)$+$i.char_3$ +$s'.form$
-	$\phi(i.char_3, s.char_2, dp.char_3, hd.char_2)$+$s.char_2$
-	$\phi(i.char_3, s.full, dp.char_3, hd.char_2)$+$s.form$
-	$\psi(s'.char_{-2}, hd.char_{-2}, head)$+$i.pos$+$s'.pos$
-	$\phi(i.char_3, s.full, dp.char_3, hd.char_2)$+$s.full$
-	$\phi(s'.full, i.char_4, dp.full, hd.char_4)$+$s'.pos$+$i.pos$
-	$\psi(i.full, hd.char_2, root)$+$i.pos$+$s.pos$
-	$\psi(i.full, hd.char_2, root)$+$i.pos$+$s'.pos$
-	$\psi(s.full, dp.full, dependant)$+$i.pos$
-	$pairscore(s'.pos, i.pos)$+$s'.form$+$i.form$
-	$rootscore(s'.pos)$+$s'.form$+$i.form$
-	$rootscore(s'.pos)$+$i.pos$

As having observed that CTB and PTB share a similar POS guideline. A POS pair list from PTB is also extract. Two types of features, $rootscore$ and $pairscore$ are used to make use of such information. Both of them returns the logarithmic value of the frequency for a given dependent event. The difference is, $rootscore$ counts for the given POS tag occurring as ROOT, and $pairscore$ counts for two POS tag combination occurring for a dependent relationship.

A full adapted feature list that is derived from the translated word pairs is in Table 3.

6 Evaluation Results

The quality of the parser is measured by the parsing accuracy or the unlabeled attachment score (UAS), i.e., the percentage of tokens with correct head. Two types of scores are reported for comparison: "UAS without p" is the UAS score without all punctuation tokens and "UAS with p" is the one with all punctuation tokens.

The results with different feature sets are in Table 4. As the features $preact_n$ are involved, a

beam search algorithm with width 5 is used for parsing, otherwise, a simple shift-reduce decoding is used. It is observed that the features derived from the translated text bring a significant performance improvement as high as 1.3%.

Table 4: The results with different feature sets

	features	with p	without p
baseline	-d	0.846	0.858
	+d[a]	0.848	0.860
+T[b]	-d	0.859	0.869
	+d	**0.861**	**0.870**

[a]+d: using three Markovian features *preact* and beam search decoding.

[b]+T: using features derived from the translated text as in Table 3.

To compare our parser to the state-of-the-art counterparts, we use the same testing data as (Wang et al., 2005) did, selecting the sentences length up to 40. Table 5 shows the results achieved by other researchers and ours (UAS with p), which indicates that our parser outperforms any other ones [4]. However, our results is only slightly better than that of (Chen et al., 2008) as only sentences whose lengths are less than 40 are considered. As our full result is much better than the latter, this comparison indicates that our approach improves the performance for those longer sentences.

Table 5: Comparison against the state-of-the-art

	full	up to 40
(McDonald and Pereira, 2006)[a]	-	0.825
(Wang et al., 2007)	-	0.866
(Chen et al., 2008)	0.852	0.884
Ours	**0.861**	**0.889**

[a]This results was reported in (Wang et al., 2007).

The experimental results in (McDonald and Nivre, 2007) show a negative impact on the parsing accuracy from too long dependency relation. For the proposed method, the improvement relative to dependency length is shown in Figure 2. From the figure, it is seen that our method gives observable better performance when dependency lengths are larger than 4. Although word order is changed, the results here show that the useful information from the translated treebank still help those long distance dependencies.

[4]There is a slight exception: using the same data splitting, (Yu et al., 2008) reported UAS without p as 0.873 versus ours, 0.870.

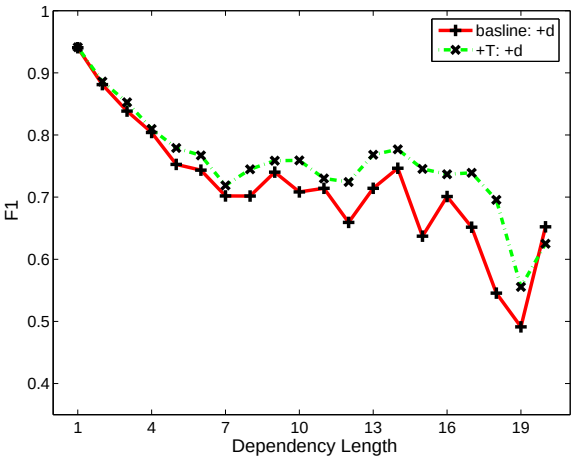


Figure 2: Performance vs. dependency length

7 Discussion

If a treebank in the source language can help improve parsing in the target language, then there must be something common between these two languages, or more precisely, these two corresponding treebanks. (Zeman and Resnik, 2008) assumed that the morphology and syntax in the language pair should be very similar, and that is so for the language pair that they considered, Danish and Swedish, two very close north European languages. Thus it is somewhat surprising that we show a translated English treebank may help Chinese parsing, as English and Chinese even belong to two different language systems. However, it will not be so strange if we recognize that PTB and CTB share very similar guidelines on POS and syntactics annotation. Since it will be too abstract in discussing the details of the annotation guidelines, we look into the similarities of two treebanks from the matching degree of two word pair lists. The reason is that the effectiveness of the proposed method actually relies on how many word pairs at every parsing states can find their full or partial matched partners in the translated word pair list. Table 6 shows such a statistics on the matching degree distribution from all training samples for Chinese parsing. The statistics in the table suggest that most to-be-check word pairs during parsing have a full or partial hitting in the translated word pair list. The latter then obtains an opportunity to provide a great deal of useful guideline information to help determine how the former should be tackled. Therefore we have cause for attributing the effectiveness of the proposed method to the similarity of these two treebanks. From Table 6,

we also find that the partial matching strategy defined in Section 5 plays a very important role in improving the whole matching degree. Note that our approach is not too related to the characteristics of two languages. Our discussion here brings an interesting issue, which difference is more important in cross language processing, between two languages themselves or the corresponding annotated corpora? This may be extensively discussed in the future work.

Table 6: Matching degree distribution

dependant-match	head-match	Percent (%)
None	None	9.6
None	Partial	16.2
None	Full	9.9
Partial	None	12.4
Partial	Partial	42.6
Partial	Full	7.3
Full	None	3.7
Full	Partial	7.0
Full	Full	0.2

Note that only a bilingual lexicon is adopted in our approach. We regard it one of the most merits for our approach. A lexicon is much easier to be obtained than an annotated corpus. One of the remained question about this work is if the bilingual lexicon should be very specific for this kind of tasks. According to our experiences, actually, it is not so sensitive to choose a highly refined lexicon or not. We once found many words, mostly named entities, were outside the lexicon. Thus we managed to collect a named entity translation dictionary to enhance the original one. However, this extra effort did not receive an observable performance improvement in return. Finally we realize that a lexicon that can guarantee two word pair lists highly matched is sufficient for this work, and this requirement may be conveniently satisfied only if the lexicon consists of adequate high-frequent words from the source treebank.

8 Conclusion and Future Work

We propose a method to enhance dependency parsing in one language by using a translated treebank from another language. A simple statistical machine translation technique, word-by-word decoding, where only a bilingual lexicon is necessary, is used to translate the source treebank. As dependency parsing is concerned with the relations of word pairs, only those word pairs with dependency relations in the translated treebank are chosen to generate some additional features to enhance the parser for the target language. The experimental results in English and Chinese treebanks show the proposed method is effective and helps the Chinese parser in this work achieve a state-of-the-art result.

Note that our method is evaluated in two treebanks with a similar annotation style and it avoids using too many linguistic properties. Thus the method is in the hope of being used in other similarly annotated treebanks [5]. For an immediate example, we may adopt a translated Chinese treebank to improve English parsing. Although there are still something to do, the remained key work has been as simple as considering how to determine the matching strategy for searching the translated word pair list in English according to the framework of our method. .

Acknowledgements

We'd like to give our thanks to three anonymous reviewers for their insightful comments, Dr. Chen Wenliang for for helpful discussions and Mr. Liu Jun for helping us fix a bug in our scoring program.

References

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *EMNLP-2008*, pages 877–886, Honolulu, Hawaii, USA.

Wenliang Chen, Daisuke Kawahara, Kiyotaka Uchimoto, Yujie Zhang, and Hitoshi Isahara. 2008. Dependency parsing with short dependency relations in unlabeled data. In *Proceedings of IJCNLP-2008*, Hyderabad, India, January 8-10.

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Probabilistic parsing action models for multi-lingual dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 940–946, Prague, Czech, June 28-30.

Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryiğit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or

[5]For example, Catalan and Spanish treebanks from the AnCora(-Es/Ca) Multilevel Annotated Corpus that are annotated by the Universitat de Barcelona (CLiC-UB) and the Universitat Politècnica de Catalunya (UPC).

blended? a study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 933–939, Prague, Czech, June.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio, USA, June.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of ACL-COLING 2006*, pages 337–344, Sydney, Australia, July.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 122–131, Prague, Czech, June 28-30.

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

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL-2005*, pages 91–98, Ann Arbor, Michigan, USA, June 25-30.

Paola Merlo, Suzanne Stevenson, Vivian Tsang, and Gianluca Allaria. 2002. A multilingual paradigm for automatic verb classification. In *ACL-2002*, pages 207–214, Philadelphia, Pennsylvania, USA.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The conll 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, page 915 - 932, Prague, Czech, June.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of IWPT-2003)*, pages 149–160, Nancy, France, April 23-25.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL-2002*, pages 295–302, Philadelphia, USA, July.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *Proceedings of ACL-2007*, pages 616–623, Prague, Czech Republic, June.

Kenji Sagae and Jun' ichi Tsujii. 2007. Dependency parsing and domain adaptation with lr models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, page 1044 - 1050, Prague, Czech, June 28-30.

Noah A. Smith and Jason Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of ACL-COLING 2006*, page 569 - 576, Sydney, Australia, July.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of EACL-2003*, page 331 - 338, Budapest, Hungary, April.

Qin Iris Wang and Dale Schuurmans. 2008. Semi-supervised convex training for dependency parsing. In *Proceedings of ACL-08: HLT*, pages 532–540, Columbus, Ohio, USA, June.

Qin Iris Wang, Dale Schuurmans, and Dekang Lin. 2005. Strictly lexical dependency parsing. In *Proceedings of IWPT-2005*, pages 152–159, Vancouver, BC, Canada, October.

Qin Iris Wang, Dekang Lin, and Dale Schuurmans. 2007. Simple training of dependency parsers via structured boosting. In *Proceedings of IJCAI 2007*, pages 1756–1762, Hyderabad, India, January.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT-2003)*, page 195 - 206, Nancy, France, April.

Kun Yu, Daisuke Kawahara, and Sadao Kurohashi. 2008. Chinese dependency parsing with large scale automatically constructed case structures. In *Proceedings of COLING-2008*, pages 1049–1056, Manchester, UK, August.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of IJCNLP 2008 Workshop on NLP for Less Privileged Languages*, pages 35–42, Hyderabad, India, January.

Hai Zhao and Chunyu Kit. 2008. Parsing syntactic and semantic dependencies with two single-stage maximum entropy models. In *Proceeding of CoNLL-2008*, pages 203–207, Manchester, UK.

Hai Zhao, Wenliang Chen, Chunyu Kit, and Guodong Zhou. 2009. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of CoNLL-2009*, Boulder, Colorado, USA.

Hai Zhao. 2009. Character-level dependencies in chinese: Usefulness and learning. In *EACL-2009*, pages 879–887, Athens, Greece.

Topological Field Parsing of German

Jackie Chi Kit Cheung
Department of Computer Science
University of Toronto
Toronto, ON, M5S 3G4, Canada
jcheung@cs.toronto.edu

Gerald Penn
Department of Computer Science
University of Toronto
Toronto, ON, M5S 3G4, Canada
gpenn@cs.toronto.edu

Abstract

Freer-word-order languages such as German exhibit linguistic phenomena that present unique challenges to traditional CFG parsing. Such phenomena produce discontinuous constituents, which are not naturally modelled by projective phrase structure trees. In this paper, we examine *topological field parsing*, a shallow form of parsing which identifies the major sections of a sentence in relation to the clausal main verb and the subordinating heads. We report the results of topological field parsing of German using the unlexicalized, latent variable-based Berkeley parser (Petrov et al., 2006) Without any language- or model-dependent adaptation, we achieve state-of-the-art results on the TüBa-D/Z corpus, and a modified NEGRA corpus that has been automatically annotated with topological fields (Becker and Frank, 2002). We also perform a qualitative error analysis of the parser output, and discuss strategies to further improve the parsing results.

1 Introduction

Freer-word-order languages such as German exhibit linguistic phenomena that present unique challenges to traditional CFG parsing. Topic focus ordering and word order constraints that are sensitive to phenomena other than grammatical function produce discontinuous constituents, which are not naturally modelled by projective (i.e., without crossing branches) phrase structure trees. In this paper, we examine topological field parsing, a shallow form of parsing which identifies the major sections of a sentence in relation to the clausal main verb and subordinating heads, when present. We report the results of parsing German using the unlexicalized, latent variable-based Berkeley parser (Petrov et al., 2006). Without any language- or model-dependent adaptation, we achieve state-of-the-art results on the TüBa-D/Z corpus (Telljohann et al., 2004), with a F_1-measure of 95.15% using gold POS tags. A further reranking of the parser output based on a constraint involving paired punctuation produces a slight additional performance gain. To facilitate comparison with previous work, we also conducted experiments on a modified NEGRA corpus that has been automatically annotated with topological fields (Becker and Frank, 2002), and found that the Berkeley parser outperforms the method described in that work. Finally, we perform a qualitative error analysis of the parser output on the TüBa-D/Z corpus, and discuss strategies to further improve the parsing results.

German syntax and parsing have been studied using a variety of grammar formalisms. Hockenmaier (2006) has translated the German TIGER corpus (Brants et al., 2002) into a CCG-based treebank to model word order variations in German. Foth et al. (2004) consider a version of dependency grammars known as *weighted constraint dependency grammars* for parsing German sentences. On the NEGRA corpus (Skut et al., 1998), they achieve an accuracy of 89.0% on parsing dependency edges. In Callmeier (2000), a platform for efficient HPSG parsing is developed. This parser is later extended by Frank et al. (2003) with a topological field parser for more efficient parsing of German. The system by Rohrer and Forst (2006) produces LFG parses using a manually designed grammar and a stochastic parse disambiguation process. They test on the TIGER corpus and achieve an F_1-measure of 84.20%. In Dubey and Keller (2003), PCFG parsing of NEGRA is improved by using sister-head dependencies, which outperforms standard head lexicalization as well as an unlexicalized model. The best

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 64–72,
Suntec, Singapore, 2-7 August 2009.

performing model with gold tags achieve an F_1 of 75.60%. Sister-head dependencies are useful in this case because of the flat structure of NEGRA's trees.

In contrast to the deeper approaches to parsing described above, topological field parsing identifies the major sections of a sentence in relation to the clausal main verb and subordinating heads, when present. Like other forms of shallow parsing, topological field parsing is useful as the first stage to further processing and eventual semantic analysis. As mentioned above, the output of a topological field parser is used as a guide to the search space of a HPSG parsing algorithm in Frank et al. (2003). In Neumann et al. (2000), topological field parsing is part of a divide-and-conquer strategy for shallow analysis of German text with the goal of improving an information extraction system.

Existing work in identifying topological fields can be divided into chunkers, which identify the lowest-level non-recursive topological fields, and parsers, which also identify sentence and clausal structure.

Veenstra et al. (2002) compare three approaches to topological field chunking based on finite state transducers, memory-based learning, and PCFGs respectively. It is found that the three techniques perform about equally well, with F_1 of 94.1% using POS tags from the TnT tagger, and 98.4% with gold tags. In Liepert (2003), a topological field chunker is implemented using a multi-class extension to the canonically two-class support vector machine (SVM) machine learning framework. Parameters to the machine learning algorithm are fine-tuned by a genetic search algorithm, with a resulting F_1-measure of 92.25%. Training the parameters to SVM does not have a large effect on performance, increasing the F_1-measure in the test set by only 0.11%.

The corpus-based, stochastic topological field parser of Becker and Frank (2002) is based on a standard treebank PCFG model, in which rule probabilities are estimated by frequency counts. This model includes several enhancements, which are also found in the Berkeley parser. First, they use parameterized categories, splitting nonterminals according to linguistically based intuitions, such as splitting different clause types (they do not distinguish different clause types as basic categories, unlike TüBa-D/Z). Second, they take into account punctuation, which may help identify clause boundaries. They also binarize the very flat topological tree structures, and prune rules that only occur once. They test their parser on a version of the NEGRA corpus, which has been annotated with topological fields using a semi-automatic method.

Ule (2003) proposes a process termed *Directed Treebank Refinement* (DTR). The goal of DTR is to refine a corpus to improve parsing performance. DTR is comparable to the idea of latent variable grammars on which the Berkeley parser is based, in that both consider the observed treebank to be less than ideal and both attempt to refine it by splitting and merging nonterminals. In this work, splitting and merging nonterminals are done by considering the nonterminals' contexts (i.e., their parent nodes) and the distribution of their productions. Unlike in the Berkeley parser, splitting and merging are distinct stages, rather than parts of a single iteration. Multiple splits are found first, then multiple rounds of merging are performed. No smoothing is done. As an evaluation, DTR is applied to topological field parsing of the TüBa-D/Z corpus. We discuss the performance of these topological field parsers in more detail below.

All of the topological parsing proposals predate the advent of the Berkeley parser. The experiments of this paper demonstrate that the Berkeley parser outperforms previous methods, many of which are specialized for the task of topological field chunking or parsing.

2 Topological Field Model of German

Topological fields are high-level linear fields in an enclosing syntactic region, such as a clause (Höhle, 1983). These fields may have constraints on the number of words or phrases they contain, and do not necessarily form a semantically coherent constituent. Although it has been argued that a few languages have no word-order constraints whatsoever, most "free word-order" languages (even Warlpiri) have at the very least some sort of sentence- or clause-initial topic field followed by a second position that is occupied by clitics, a finite verb or certain complementizers and subordinating conjunctions. In a few Germanic languages, including German, the topology is far richer than that, serving to identify all of the components of the verbal head of a clause, except for some cases of long-distance dependen-

cies. Topological fields are useful, because while Germanic word order is relatively free with respect to grammatical functions, the order of the topological fields is strict and unvarying.

Type	Fields
VL	(KOORD) (C) (MF) VC (NF)
V1	(KOORD) (LV) LK (MF) (VC) (NF)
V2	(KOORD) (LV) VF LK (MF) (VC) (NF)

Table 1: Topological field model of German. Simplified from TüBa-D/Z corpus's annotation schema (Telljohann et al., 2006).

In the German topological field model, clauses belong to one of three types: verb-last (VL), verb-second (V2), and verb-first (V1), each with a specific sequence of topological fields (Table 1). VL clauses include finite and non-finite subordinate clauses, V2 sentences are typically declarative sentences and WH-questions in matrix clauses, and V1 sentences include yes-no questions, and certain conditional subordinate clauses. Below, we give brief descriptions of the most common topological fields.

- VF (*Vorfeld* or 'pre-field') is the first constituent in sentences of the V2 type. This is often the topic of the sentence, though as an anonymous reviewer pointed out, this position does not correspond to a single function with respect to information structure. (*e.g., the reviewer suggested this case, where VF contains the focus: –Wer kommt zur Party? –**Peter** kommt zur Party.* –Who is coming to the Party? –**Peter** is coming to the party.)
- LK (*Linke Klammer* or 'left bracket') is the position for finite verbs in V1 and V2 sentences. It is replaced by a complementizer with the field label C in VL sentences.
- MF (*Mittelfeld* or 'middle field') is an optional field bounded on the left by LK and on the right by the verbal complex VC or by NF. Most verb arguments, adverbs, and prepositional phrases are found here, unless they have been fronted and put in the VF, or are prosodically heavy and postposed to the NF field.
- VC is the verbal complex field. It includes infinite verbs, as well as finite verbs in VL sentences.
- NF (*Nachfeld* or 'post-field') contains prosodically heavy elements such as postposed prepositional phrases or relative clauses.
- KOORD[1] (*Koordinationsfeld* or 'coordination field') is a field for clause-level conjunctions.
- LV (*Linksversetzung* or 'left dislocation') is used for resumptive constructions involving left dislocation. For a detailed linguistic treatment, see (Frey, 2004).

Exceptions to the topological field model as described above do exist. For instance, parenthetical constructions exist as a mostly syntactically independent clause inside another sentence. In our corpus, they are attached directly underneath a clausal node without any intervening topological field, as in the following example. In this example, the parenthetical construction is highlighted in bold print. Some clause and topological field labels under the NF field are omitted for clarity.

(1) (a) *(SIMPX "(VF Man) (LK muß) (VC verstehen) " , **(SIMPX sagte er)**, " (NF daß diese Minderheiten seit langer Zeit massiv von den Nazis bedroht werden)). "*
(b) Translation: "One must understand," **he said**, "that these minorities have been massively threatened by the Nazis for a long time."

3 A Latent Variable Parser

For our experiments, we used the latent variable-based Berkeley parser (Petrov et al., 2006). Latent variable parsing assumes that an observed treebank represents a coarse approximation of an underlying, optimally refined grammar which makes more fine-grained distinctions in the syntactic categories. For example, the noun phrase category *NP* in a treebank could be viewed as a coarse approximation of two noun phrase categories corresponding to subjects and object, *NP^S*, and *NP^VP*.

The Berkeley parser automates the process of finding such distinctions. It starts with a simple binarized X-bar grammar style backbone, and goes through iterations of splitting and merging nonterminals, in order to maximize the likelihood of the training set treebank. In the splitting stage,

[1]The TüBa-D/Z corpus distinguishes coordinating and non-coordinating particles, as well as clausal and field coordination. These distinctions need not concern us for this explanation.

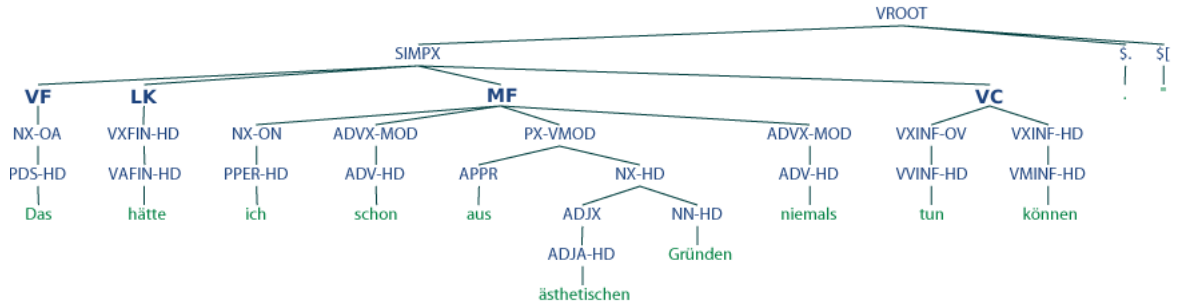


Figure 1: "I could never have done that just for aesthetic reasons." Sample TüBa-D/Z tree, with topological field annotations and edge labels. Topological field layer in bold.

an Expectation-Maximization algorithm is used to find a good split for each nonterminal. In the merging stage, categories that have been oversplit are merged together to keep the grammar size tractable and reduce sparsity. Finally, a smoothing stage occurs, where the probabilities of rules for each nonterminal are smoothed toward the probabilities of the other nonterminals split from the same syntactic category.

The Berkeley parser has been applied to the TüBaD/Z corpus in the constituent parsing shared task of the ACL-2008 Workshop on Parsing German (Petrov and Klein, 2008), achieving an F_1-measure of 85.10% and 83.18% with and without gold standard POS tags respectively[2]. We chose the Berkeley parser for topological field parsing because it is known to be robust across languages, and because it is an unlexicalized parser. Lexicalization has been shown to be useful in more general parsing applications due to lexical dependencies in constituent parsing (e.g. (Kübler et al., 2006; Dubey and Keller, 2003) in the case of German). However, topological fields explain a higher level of structure pertaining to clause-level word order, and we hypothesize that lexicalization is unlikely to be helpful.

4 Experiments

4.1 Data

For our experiments, we primarily used the TüBa-D/Z (Tübinger Baumbank des Deutschen / Schriftsprache) corpus, consisting of 26116 sentences (20894 training, 2611 development, 2089 test, with a further 522 sentences held out for future experiments)[3] taken from the German newspaper *die tageszeitung*. The corpus consists of four levels of annotation: clausal, topological, phrasal (other than clausal), and lexical. We define the task of topological field parsing to be recovering the first two levels of annotation, following Ule (2003).

We also tested the parser on a version of the NEGRA corpus derived by Becker and Frank (2002), in which syntax trees have been made projective and topological fields have been automatically added through a series of linguistically informed tree modifications. All internal phrasal structure nodes have also been removed. The corpus consists of 20596 sentences, which we split into subsets of the same size as described by Becker and Frank (2002)[4]. The set of topological fields in this corpus differs slightly from the one used in TüBa-D/Z, making no distinction between clause types, nor consistently marking field or clause conjunctions. Because of the automatic annotation of topological fields, this corpus contains numerous annotation errors. Becker and Frank (2002) manually corrected their test set and evaluated the automatic annotation process, reporting labelled precision and recall of 93.0% and 93.6% compared to their manual annotations. There are also punctuation-related errors, including missing punctuation, sentences ending in commas, and sentences composed of single punctuation marks. We test on this data in order to provide a better comparison with previous work. Although we could have trained the model in Becker and Frank (2002) on the TüBa-D/Z corpus, it would not have

[2]This evaluation considered grammatical functions as well as the syntactic category.

[3]These are the same splits into training, development, and test sets as in the ACL-08 Parsing German workshop. This corpus does not include sentences of length greater than 40.

[4]16476 training sentences, 1000 development, 1058 testing, and 2062 as held-out data. We were unable to obtain the exact subsets used by Becker and Frank (2002). We will discuss the ramifications of this on our evaluation procedure.

Gold tags	*Edge labels*	*LP%*	*LR%*	$F_1\%$	$\overline{CB}$	*CB0%*	$CB \leq 2\%$	*EXACT%*
-	-	93.53	93.17	93.35	0.08	94.59	99.43	79.50
+	-	95.26	95.04	95.15	0.07	95.35	99.52	83.86
-	+	92.38	92.67	92.52	0.11	92.82	99.19	77.79
+	+	92.36	92.60	92.48	0.11	92.82	99.19	77.64

Table 2: Parsing results for topological fields and clausal constituents on the TüBa-D/Z corpus.

been a fair comparison, as the parser depends quite heavily on NEGRA's annotation scheme. For example, TüBa-D/Z does not contain an equivalent of the modified NEGRA's parameterized categories; there exist edge labels in TüBaD/Z, but they are used to mark head-dependency relationships, not subtypes of syntactic categories.

4.2 Results

We first report the results of our experiments on the TüBa-D/Z corpus. For the TüBa-D/Z corpus, we trained the Berkeley parser using the default parameter settings. The grammar trainer attempts six iterations of splitting, merging, and smoothing before returning the final grammar. Intermediate grammars after each step are also saved. There were training and test sentences without clausal constituents or topological fields, which were ignored by the parser and by the evaluation. As part of our experiment design, we investigated the effect of providing gold POS tags to the parser, and the effect of incorporating edge labels into the nonterminal labels for training and parsing. In all cases, gold annotations which include gold POS tags were used when training the parser.

We report the standard PARSEVAL measures of parser performance in Table 2, obtained by the `evalb` program by Satoshi Sekine and Michael Collins. This table shows the results after five iterations of grammar modification, parameterized over whether we provide gold POS tags for parsing, and edge labels for training and parsing. The number of iterations was determined by experiments on the development set. In the evaluation, we do not consider edge labels in determining correctness, but do consider punctuation, as Ule (2003) did. If we ignore punctuation in our evaluation, we obtain an F_1-measure of 95.42% on the best model (+ *Gold tags, - Edge labels*).

Whether supplying gold POS tags improves performance depends on whether edge labels are considered in the grammar. Without edge labels, gold POS tags improve performance by almost two points, corresponding to a relative error reduction of 33%. In contrast, performance is negatively affected when edge labels are used and gold POS tags are supplied (i.e., + *Gold tags,* + *Edge labels*), making the performance *worse* than not supplying gold tags. Incorporating edge label information does not appear to improve performance, possibly because it oversplits the initial treebank and interferes with the parser's ability to determine optimal splits for refining the grammar.

Parser	*LP%*	*LR%*	$F_1\%$
TüBa-D/Z			
This work	95.26	95.04	95.15
Ule	unknown	unknown	91.98
NEGRA - from Becker and Frank (2002)			
BF02 (len. ≤ 40)	92.1	91.6	91.8
NEGRA - our experiments			
This work (len. ≤ 40)	90.74	90.87	90.81
BF02 (len. ≤ 40)	89.54	88.14	88.83
This work (all)	90.29	90.51	90.40
BF02 (all)	89.07	87.80	88.43

Table 3: BF02 = (Becker and Frank, 2002). Parsing results for topological fields and clausal constituents. Results from Ule (2003) and our results were obtained using different training and test sets. The first row of results of Becker and Frank (2002) are from that paper; the rest were obtained by our own experiments using that parser. All results consider punctuation in evaluation.

To facilitate a more direct comparison with previous work, we also performed experiments on the modified NEGRA corpus. In this corpus, topological fields are *parameterized*, meaning that they are labelled with further syntactic and semantic information. For example, VF is split into VF-REL for relative clauses, and VF-TOPIC for those containing topics in a verb-second sentence, among others. All productions in the corpus have also been binarized. Tuning the parameter settings on the development set, we found that parameterized categories, binarization, and including punctuation gave the best F_1 performance. First-order horizontal and zeroth order vertical markoviza-

tion after six iterations of splitting, merging, and smoothing gave the best F_1 result of 91.78%. We parsed the corpus with both the Berkeley parser and the best performing model of Becker and Frank (2002).

The results of these experiments on the test set for sentences of length 40 or less and for all sentences are shown in Table 3. We also show other results from previous work for reference. We find that we achieve results that are better than the model in Becker and Frank (2002) on the test set. The difference is statistically significant (p = 0.0029, Wilcoxon signed-rank).

The results we obtain using the parser of Becker and Frank (2002) are worse than the results described in that paper. We suggest the following reasons for this discrepancy. While the test set used in the paper was manually corrected for evaluation, we did not correct our test set, because it would be difficult to ensure that we adhered to the same correction guidelines. No details of the correction process were provided in the paper, and descriptive grammars of German provide insufficient guidance on many of the examples in NEGRA on issues such as ellipses, short infinitival clauses, and expanded participial constructions modifying nouns. Also, because we could not obtain the exact sets used for training, development, and testing, we had to recreate the sets by randomly splitting the corpus.

4.3 Category Specific Results

We now return to the TüBa-D/Z corpus for a more detailed analysis, and examine the category-specific results for our best performing model (+ *Gold tags, - Edge labels*). Overall, Table 4 shows that the best performing topological field categories are those that have constraints on the type of word that is allowed to fill it (finite verbs in LK, verbs in VC, complementizers and subordinating conjunctions in C). VF, in which only one constituent may appear, also performs relatively well. Topological fields that can contain a variable number of heterogeneous constituents, on the other hand, have poorer F_1-measure results. MF, which is basically defined relative to the positions of fields on either side of it, is parsed several points below LK, C, and VC in accuracy. NF, which contains different kinds of extraposed elements, is parsed at a substantially worse level.

Poorly parsed categories tend to occur infrequently, including LV, which marks a rare resumptive construction; FKOORD, which marks topological field coordination; and the discourse marker DM. The other clause-level constituents (PSIMPX for clauses in paratactic constructions, RSIMPX for relative clauses, and SIMPX for other clauses) also perform below average.

Topological Fields				
Category	#	*LP%*	*LR%*	F_1%
PARORD	20	100.00	100.00	100.00
VCE	3	100.00	100.00	100.00
LK	2186	99.68	99.82	99.75
C	642	99.53	98.44	98.98
VC	1777	98.98	98.14	98.56
VF	2044	96.84	97.55	97.20
KOORD	99	96.91	94.95	95.92
MF	2931	94.80	95.19	94.99
NF	643	83.52	81.96	82.73
FKOORD	156	75.16	73.72	74.43
LV	17	10.00	5.88	7.41
Clausal Constituents				
Category	#	*LP%*	*LR%*	F_1%
SIMPX	2839	92.46	91.97	92.21
RSIMPX	225	91.23	92.44	91.83
PSIMPX	6	100.00	66.67	80.00
DM	28	59.26	57.14	58.18

Table 4: Category-specific results using grammar with no edge labels and passing in gold POS tags.

4.4 Reranking for Paired Punctuation

While experimenting with the development set of TüBa-D/Z, we noticed that the parser sometimes returns parses, in which paired punctuation (e.g. quotation marks, parentheses, brackets) is not placed in the same clause–a linguistically implausible situation. In these cases, the high-level information provided by the paired punctuation is overridden by the overall likelihood of the parse tree. To rectify this problem, we performed a simple post-hoc reranking of the 50-best parses produced by the best parameter settings (+ *Gold tags, - Edge labels*), selecting the first parse that places paired punctuation in the same clause, or returning the best parse if none of the 50 parses satisfy the constraint. This procedure improved the F_1-measure to 95.24% (LP = 95.39%, LR = 95.09%).

Overall, 38 sentences were parsed with paired punctuation in different clauses, of which 16 were reranked. Of the 38 sentences, reranking improved performance in 12 sentences, did not affect performance in 23 sentences (of which 10 already had a perfect parse), and hurt performance in three sentences. A two-tailed sign test suggests that rerank-

ing improves performance (p = 0.0352). We discuss below why sentences with paired punctuation in different clauses can have perfect parse results.

To investigate the upper-bound in performance that this form of reranking is able to achieve, we calculated some statistics on our (+ *Gold tags, - Edge labels*) 50-best list. We found that the average rank of the best scoring parse by F_1-measure is 2.61, and the perfect parse is present for 1649 of the 2088 sentences at an average rank of 1.90. The oracle F_1-measure is 98.12%, indicating that a more comprehensive reranking procedure might allow further performance gains.

4.5 Qualitative Error Analysis

As a further analysis, we extracted the worst scoring fifty sentences by F_1-measure from the parsed test set (+ *Gold tags, - Edge labels*), and compared them against the gold standard trees, noting the cause of the error. We analyze the parses before reranking, to see how frequently the paired punctuation problem described above severely affects a parse. The major mistakes made by the parser are summarized in Table 5.

Problem	*Freq.*
Misidentification of Parentheticals	19
Coordination problems	13
Too few SIMPX	10
Paired punctuation problem	9
Other clause boundary errors	7
Other	6
Too many SIMPX	3
Clause type misidentification	2
MF/NF boundary	2
LV	2
VF/MF boundary	2

Table 5: Types and frequency of parser errors in the fifty worst scoring parses by F_1-measure, using parameters (+ *Gold tags, - Edge labels*).

Misidentification of Parentheticals Parenthetical constructions do not have any dependencies on the rest of the sentence, and exist as a mostly syntactically independent clause inside another sentence. They can occur at the beginning, end, or in the middle of sentences, and are often set off orthographically by punctuation. The parser has problems identifying parenthetical constructions, often positing a parenthetical construction when that constituent is actually attached to a topological field in a neighbouring clause. The following example shows one such misidentification in bracket notation. Clause internal topological fields are omitted for clarity.

(2)
(a) TüBa-D/Z: (SIMPX *Weder das Ausmaß der Schönheit noch der frühere oder spätere Zeitpunkt der Geburt macht einen der Zwillinge für eine Mutter mehr oder weniger echt / authentisch / überlegen*).
(b) Parser: (SIMPX *Weder das Ausmaß der Schönheit noch der frühere oder spätere Zeitpunkt der Geburt macht einen der Zwillinge für eine Mutter mehr oder weniger echt*) **(PARENTHETICAL / *authentisch / überlegen.*)**
(c) Translation: "Neither the degree of beauty nor the earlier or later time of birth makes one of the twins any more or less real/authentic/superior to a mother."

We hypothesized earlier that lexicalization is unlikely to give us much improvement in performance, because topological fields work on a domain that is higher than that of lexical dependencies such as subcategorization frames. However, given the locally independent nature of legitimate parentheticals, a limited form of lexicalization or some other form of stronger contextual information might be needed to improve identification performance.

Coordination Problems The second most common type of error involves field and clause coordinations. This category includes missing or incorrect FKOORD fields, and conjunctions of clauses that are misidentified. In the following example, the conjoined MFs and following NF in the correct parse tree are identified as a single long MF.

(3)
(a) TüBa-D/Z: *Auf dem europäischen Kontinent aber hat (FKOORD* ***(MF kein Land und keine Macht ein derartiges Interesse an guten Beziehungen zu Rußland)*** *und* ***(MF auch kein Land solche Erfahrungen im Umgang mit Rußland))*** *(NF wie Deutschland).*
(b) Parser: *Auf dem europäischen Kontinent aber hat* ***(MF kein Land und keine Macht ein derartiges Interesse an guten Beziehungen zu Rußland und auch kein Land solche Erfahrungen im Umgang mit Rußland wie Deutschland)***.
(c) Translation: "On the European continent, however, no land and no power has such an interest in good relations with Russia (as Germany), and also no land (has) such experience in dealing with Russia as Germany."

Other Clause Errors Other clause-level errors include the parser predicting too few or too many clauses, or misidentifying the clause type. Clauses are sometimes confused with NFs, and there is one case of a relative clause being misidentified as a

main clause with an intransitive verb, as the finite verb appears at the end of the clause in both cases. Some clause errors are tied to incorrect treatment of elliptical constructions, in which an element that is inferable from context is missing.

Paired Punctuation Problems with paired punctuation are the fourth most common type of error. Punctuation is often a marker of clause or phrase boundaries. Thus, predicting paired punctuation incorrectly can lead to incorrect parses, as in the following example.

(4) (a) *“ Auch (SIMPX wenn der Krieg heute ein Mobilisierungsfaktor ist) ” , so Pau , “ (SIMPX die Leute sehen , daß man für die Arbeit wieder auf die Straße gehen muß) . ”*

(b) Parser: ***(SIMPX** “ **(LV** Auch (SIMPX wenn der Krieg heute ein Mobilisierungsfaktor ist)) ” , so Pau , “ (SIMPX die Leute sehen , daß man für die Arbeit wieder auf die Straße gehen muß)) . ”*

(c) Translation: “Even if the war is a factor for mobilization,” said Pau, “the people see, that one must go to the street for employment again.”

Here, the parser predicts a spurious SIMPX clause spanning the text of the entire sentence, but this causes the second pair of quotation marks to be parsed as belonging to two different clauses. The parser also predicts an incorrect LV field. Using the paired punctuation constraint, our reranking procedure was able to correct these errors.

Surprisingly, there are cases in which paired punctuation does not belong inside the same clause in the gold parses. These cases are either extended quotations, in which each of the quotation mark pair occurs in a different sentence altogether, or cases where the second of the quotation mark pair must be positioned outside of other sentence-final punctuation due to orthographic conventions. Sentence-final punctuation is typically placed outside a clause in this version of TüBa-D/Z.

Other Issues Other incorrect parses generated by the parser include problems with the infrequently occurring topological fields like LV and DM, inability to determine the boundary between MF and NF in clauses without a VC field separating the two, and misidentifying appositive constructions. Another issue is that although the parser output may disagree with the gold standard tree in TüBa-D/Z, the parser output may be a well-formed topological field parse for the same sentence with a different interpretation, for example because of attachment ambiguity. Each of the authors independently checked the fifty worst-scoring parses, and determined whether each parse produced by the Berkeley parser could be a well-formed topological parse. Where there was disagreement, we discussed our judgments until we came to a consensus. Of the fifty parses, we determined that nine, or 18%, could be legitimate parses. Another five, or 10%, differ from the gold standard parse only in the placement of punctuation. Thus, the F_1-measures we presented above may be underestimating the parser's performance.

5 Conclusion and Future Work

In this paper, we examined applying the latent-variable Berkeley parser to the task of topological field parsing of German, which aims to identify the high-level surface structure of sentences. Without any language or model-dependent adaptation, we obtained results which compare favourably to previous work in topological field parsing. We further examined the results of doing a simple reranking process, constraining the output parse to put paired punctuation in the same clause. This reranking was found to result in a minor performance gain.

Overall, the parser performs extremely well in identifying the traditional left and right brackets of the topological field model; that is, the fields C, LK, and VC. The parser achieves basically perfect results on these fields in the TüBa-D/Z corpus, with F_1-measure scores for each at over 98.5%. These scores are higher than previous work in the simpler task of topological field chunking. The focus of future research should thus be on correctly identifying the infrequently occuring fields and constructions, with parenthetical constructions being a particular concern. Possible avenues of future research include doing a more comprehensive discriminative reranking of the parser output. Incorporating more contextual information might be helpful to identify discourse-related constructions such as parentheses, and the DM and LV topological fields.

Acknowledgements

We are grateful to Markus Becker, Anette Frank, Sandra Kuebler, and Slav Petrov for their invaluable help in gathering the resources necessary for our experiments. This work is supported in part by the Natural Sciences and Engineering Research Council of Canada.

References

M. Becker and A. Frank. 2002. A stochastic topological parser for German. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 71–77.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER Treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, pages 24–41.

U. Callmeier. 2000. PET–a platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6(01):99–107.

A. Dubey and F. Keller. 2003. Probabilistic parsing for German using sister-head dependencies. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 96–103.

K.A. Foth, M. Daum, and W. Menzel. 2004. A broad-coverage parser for German based on defeasible constraints. *Constraint Solving and Language Processing*.

A. Frank, M. Becker, B. Crysmann, B. Kiefer, and U. Schaefer. 2003. Integrated shallow and deep parsing: TopP meets HPSG. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

W. Frey. 2004. Notes on the syntax and the pragmatics of German Left Dislocation. In H. Lohnstein and S. Trissler, editors, *The Syntax and Semantics of the Left Periphery*, pages 203–233. Mouton de Gruyter, Berlin.

J. Hockenmaier. 2006. Creating a CCGbank and a Wide-Coverage CCG Lexicon for German. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 505–512.

T.N. Höhle. 1983. *Topologische Felder*. Ph.D. thesis, Köln.

S. Kübler, E.W. Hinrichs, and W. Maier. 2006. Is it really that difficult to parse German? In *Proceedings of EMNLP*.

M. Liepert. 2003. Topological Fields Chunking for German with SVM's: Optimizing SVM-parameters with GA's. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP), Bulgaria*.

G. Neumann, C. Braun, and J. Piskorski. 2000. A Divide-and-Conquer Strategy for Shallow Parsing of German Free Texts. In *Proceedings of the sixth conference on Applied natural language processing*, pages 239–246. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

S. Petrov and D. Klein. 2008. Parsing German with Latent Variable Grammars. In *Proceedings of the ACL-08: HLT Workshop on Parsing German (PaGe-08)*, pages 33–39.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

C. Rohrer and M. Forst. 2006. Improving coverage and parsing quality of a large-scale LFG for German. In *Proceedings of the Language Resources and Evaluation Conference (LREC-2006)*, Genoa, Italy.

W. Skut, T. Brants, B. Krenn, and H. Uszkoreit. 1998. A Linguistically Interpreted Corpus of German Newspaper Text. *Proceedings of the ESSLLI Workshop on Recent Advances in Corpus Annotation*.

H. Telljohann, E. Hinrichs, and S. Kubler. 2004. The TüBa-D/Z treebank: Annotating German with a context-free backbone. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 2229–2235.

H. Telljohann, E.W. Hinrichs, S. Kubler, and H. Zinsmeister. 2006. Stylebook for the Tubingen Treebank of Written German (TüBa-D/Z). *Seminar fur Sprachwissenschaft, Universitat Tubingen, Tubingen, Germany*.

T. Ule. 2003. Directed Treebank Refinement for PCFG Parsing. In *Proceedings of Workshop on Treebanks and Linguistic Theories (TLT) 2003*, pages 177–188.

J. Veenstra, F.H. Müller, and T. Ule. 2002. Topological field chunking for German. In *Proceedings of the Sixth Conference on Natural Language Learning*, pages 56–62.

Unsupervised Multilingual Grammar Induction

Benjamin Snyder, Tahira Naseem, and **Regina Barzilay**
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{bsnyder, tahira, regina}@csail.mit.edu

Abstract

We investigate the task of unsupervised constituency parsing from bilingual parallel corpora. Our goal is to use bilingual cues to learn improved parsing models for each language and to evaluate these models on held-out monolingual test data. We formulate a generative Bayesian model which seeks to explain the observed parallel data through a combination of bilingual and monolingual parameters. To this end, we adapt a formalism known as *unordered tree alignment* to our probabilistic setting. Using this formalism, our model loosely binds parallel trees while allowing language-specific syntactic structure. We perform inference under this model using Markov Chain Monte Carlo and dynamic programming. Applying this model to three parallel corpora (Korean-English, Urdu-English, and Chinese-English) we find substantial performance gains over the CCM model, a strong monolingual baseline. On average, across a variety of testing scenarios, our model achieves an 8.8 absolute gain in F-measure. [1]

1 Introduction

In this paper we investigate the task of unsupervised constituency parsing when bilingual parallel text is available. Our goal is to improve parsing performance on monolingual test data for each language by using unsupervised bilingual cues at training time. Multilingual learning has been successful for other linguistic induction tasks such as lexicon acquisition, morphological segmentation, and part-of-speech tagging (Genzel, 2005; Snyder and Barzilay, 2008; Snyder et al., 2008; Snyder et al., 2009). We focus here on the unsupervised induction of unlabeled constituency brackets. This task has been extensively studied in a monolingual setting and has proven to be difficult (Charniak and Carroll, 1992; Klein and Manning, 2002).

The key premise of our approach is that ambiguous syntactic structures in one language may correspond to less uncertain structures in the other language. For instance, the English sentence *I saw* [*the student* [*from MIT*]] exhibits the classic problem of PP-attachment ambiguity. However, its Urdu translation, literally glossed as *I* [[*MIT of*] *student*] *saw*, uses a genitive phrase that may only be attached to the adjacent noun phrase. Knowing the correspondence between these sentences should help us resolve the English ambiguity.

One of the main challenges of unsupervised multilingual learning is to exploit cross-lingual patterns discovered in data, while still allowing a wide range of language-specific idiosyncrasies. To this end, we adapt a formalism known as *unordered tree alignment* (Jiang et al., 1995) to a probabilistic setting. Under this formalism, any two trees can be embedded in an *alignment tree*. This alignment tree allows arbitrary parts of the two trees to diverge in structure, permitting language-specific grammatical structure to be preserved. Additionally, a computational advantage of this formalism is that the marginalized probability over all possible alignments for any two trees can be efficiently computed with a dynamic program in linear time.

We formulate a generative Bayesian model which seeks to explain the observed parallel data through a combination of bilingual and monolingual parameters. Our model views each pair of sentences as having been generated as follows: First an alignment tree is drawn. Each node in this alignment tree contains either a solitary monolingual constituent or a pair of coupled bilingual constituents. For each solitary mono-

[1] Code and the outputs of our experiments are available at http://groups.csail.mit.edu/rbg/code/multiling_induction.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 73–81,
Suntec, Singapore, 2-7 August 2009.

lingual constituent, a sequence of part-of-speech tags is drawn from a language-specific distribution. For each pair of coupled bilingual constituents, a pair of part-of-speech sequences are drawn jointly from a cross-lingual distribution. Word-level alignments are then drawn based on the tree alignment. Finally, parallel sentences are assembled from these generated part-of-speech sequences and word-level alignments.

To perform inference under this model, we use a Metropolis-Hastings within-Gibbs sampler. We sample pairs of trees and then compute marginalized probabilities over all possible alignments using dynamic programming.

We test the effectiveness of our bilingual grammar induction model on three corpora of parallel text: English-Korean, English-Urdu and English-Chinese. The model is trained using bilingual data with automatically induced word-level alignments, but is tested on purely monolingual data for each language. In all cases, our model outperforms a state-of-the-art baseline: the Constituent Context Model (CCM) (Klein and Manning, 2002), sometimes by substantial margins. On average, over all the testing scenarios that we studied, our model achieves an absolute increase in F-measure of 8.8 points, and a 19% reduction in error relative to a theoretical upper bound.

2 Related Work

The unsupervised grammar induction task has been studied extensively, mostly in a monolingual setting (Charniak and Carroll, 1992; Stolcke and Omohundro, 1994; Klein and Manning, 2002; Seginer, 2007). While PCFGs perform poorly on this task, the CCM model (Klein and Manning, 2002) has achieved large gains in performance and is among the state-of-the-art probabilistic models for unsupervised constituency parsing. We therefore use CCM as our basic model of monolingual syntax.

While there has been some previous work on bilingual CFG parsing, it has mainly focused on improving MT systems rather than monolingual parsing accuracy. Research in this direction was pioneered by (Wu, 1997), who developed Inversion Transduction Grammars to capture cross-lingual grammar variations such as phrase reorderings. More general formalisms for the same purpose were later developed (Wu and Wong, 1998; Chiang, 2005; Melamed, 2003; Eisner, 2003; Zhang and Gildea, 2005; Blunsom et al., 2008). We know of only one study which evaluates these bilingual grammar formalisms on the task of grammar induction itself (Smith and Smith, 2004). Both our model and even the monolingual CCM baseline yield far higher performance on the same Korean-English corpus.

Our approach is closer to the unsupervised bilingual parsing model developed by Kuhn (2004), which aims to improve monolingual performance. Assuming that trees induced over parallel sentences have to exhibit certain structural regularities, Kuhn manually specifies a set of rules for determining when parsing decisions in the two languages are inconsistent with GIZA++ word-level alignments. By incorporating these constraints into the EM algorithm he was able to improve performance over a monolingual unsupervised PCFG. Still, the performance falls short of state-of-the-art monolingual models such as the CCM.

More recently, there has been a body of work attempting to improve parsing performance by exploiting syntactically annotated parallel data. In one strand of this work, annotations are assumed only in a resource-rich language and are projected onto a resource-poor language using the parallel data (Hwa et al., 2005; Xi and Hwa, 2005). In another strand of work, syntactic annotations are assumed on both sides of the parallel data, and a model is trained to exploit the parallel data at test time as well (Smith and Smith, 2004; Burkett and Klein, 2008). In contrast to this work, our goal is to explore the benefits of multilingual grammar induction in a fully unsupervised setting.

We finally note a recent paper which uses parameter tying to improve unsupervised dependency parse induction (Cohen and Smith, 2009). While the primary performance gains occur when tying related parameters within a language, some additional benefit is observed through bilingual tying, even in the absence of a parallel corpus.

3 Model

We propose an unsupervised Bayesian model for learning bilingual syntactic structure using parallel corpora. Our key premise is that difficult-to-learn syntactic structures of one language may correspond to simpler or less uncertain structures in the other language. We treat the part-of-speech tag sequences of parallel sentences, as well as their

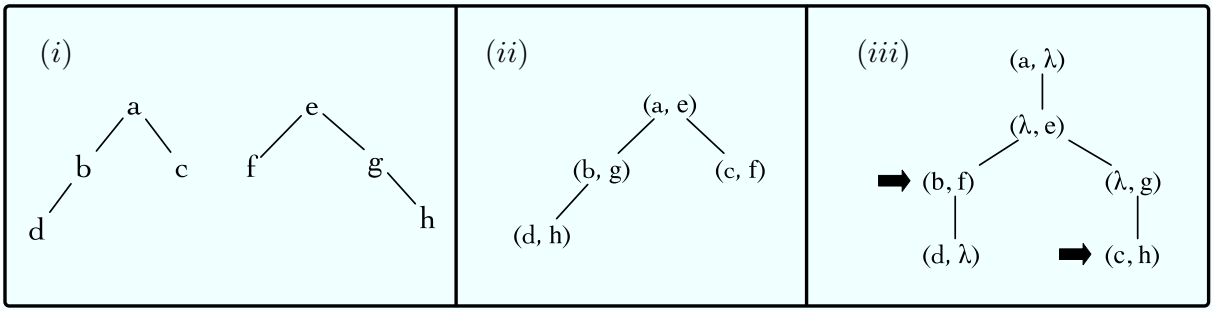


Figure 1: A pair of trees (i) and two possible alignment trees. In (ii), no empty spaces are inserted, but the order of one of the original tree's siblings has been reversed. In (iii), only two pairs of nodes have been aligned (indicated by arrows) and many empty spaces inserted.

word-level alignments, as observed data. We obtain these word-level alignments using GIZA++ (Och and Ney, 2003).

Our model seeks to explain this observed data through a generative process whereby two aligned parse trees are produced jointly. Though they are aligned, arbitrary parts of the two trees are permitted to diverge, accommodating language-specific grammatical structure. In effect, our model loosely binds the two trees: node-to-node alignments need only be used where repeated bilingual patterns can be discovered in the data.

3.1 Tree Alignments

We achieve this loose binding of trees by adapting *unordered tree alignment* (Jiang et al., 1995) to a probabilistic setting. Under this formalism, any two trees can be aligned using an *alignment tree*. The alignment tree embeds the original two trees within it: each node is labeled by a pair (x, y), (λ, y), or (x, λ) where x is a node from the first tree, y is a node from the second tree, and λ is an empty space. The individual structure of each tree must be preserved under the embedding with the exception of sibling order (to allow variations in phrase and word order).

The flexibility of this formalism can be demonstrated by two extreme cases: (1) an alignment between two trees may actually align *none* of their individual nodes, instead inserting an empty space λ for each of the original two trees' nodes. (2) if the original trees are isomorphic to one another, the alignment may match their nodes exactly, without inserting any empty spaces. See Figure 1 for an example.

3.2 Model overview

As our basic model of syntactic structure, we adopt the Constituent-Context Model (CCM) of Klein and Manning (2002). Under this model, the part-of-speech sequence of each span in a sentence is generated either as a *constituent yield* — if it is dominated by a node in the tree — or otherwise as a *distituent yield*. For example, in the bracketed sentence [*John*/NNP [*climbed*/VB [*the*/DT *tree*/NN]]], the sequence VB DT NN is generated as a constituent yield, since it constitutes a complete bracket in the tree. On the other hand, the sequence VB DT is generated as a distituent, since it does not. Besides these yields, the *contexts* (two surrounding POS tags) of constituents and distituents are generated as well. In this example, the context of the constituent VB DT NN would be (NNP, #), while the context of the distituent VB DT would be (NNP, NN). The CCM model employs separate multinomial distributions over constituents, distituents, constituent contexts, and distituent contexts. While this model is deficient — each observed subsequence of part-of-speech tags is generated many times over — its performance is far higher than that of unsupervised PCFGs.

Under our bilingual model, each pair of sentences is assumed to have been generated jointly in the following way: First, an unlabeled *alignment tree* is drawn uniformly from the set of all such trees. This alignment tree specifies the structure of each of the two individual trees, as well as the pairs of nodes which are aligned and those which are not aligned (i.e. paired with a λ).

For each pair of aligned nodes, a corresponding pair of constituents and contexts are jointly drawn from a bilingual distribution. For unaligned nodes (i.e. nodes paired with a λ in the alignment

tree), a single constituent and context are drawn, from language-specific distributions. Distituents and their contexts are also drawn from language-specific distributions. Finally, word-level alignments are drawn based on the structure of the alignment tree.

In the next two sections, we describe our model in more formal detail by specifying the parameters and generative process by which sentences are formed.

3.3 Parameters

Our model employs a number of multinomial distributions:

- π_i^C : over constituent yields of language i,
- π_i^D : over distituent yields of language i,
- ϕ_i^C : over constituent contexts of language i,
- ϕ_i^D : over distituent contexts of language i,
- ω : over *pairs* of constituent yields, one from the first language and the other from the second language,
- Gz_{pair} : over a finite set of integer values $\{-m, \ldots, -2, -1, 0, 1, 2, \ldots, m\}$, measuring the *Giza-score* of aligned tree node pairs (see below),
- Gz_{node} : over a finite set of integer values $\{-m, \ldots, -2, -1, 0\}$, measuring the *Giza-score* of unaligned tree nodes (see below).

The first four distributions correspond exactly to the parameters of the CCM model. Parameter ω is a "coupling parameter" which measures the compatibility of tree-aligned constituent yield pairs. The final two parameters measure the compatibility of syntactic alignments with the observed lexical GIZA++ alignments. Intuitively, aligned nodes should have a high density of word-level alignments between them, and unaligned nodes should have few lexical alignments.

More formally, consider a tree-aligned node pair (n_1, n_2) with corresponding yields (y_1, y_2). We call a word-level alignment *good* if it aligns a word in y_1 with a word in y_2. We call a word-level alignment *bad* if it aligns a word in y_1 with a word outside y_2, or vice versa. The *Giza-score* for (n_1, n_2) is the number of *good* word alignments minus the number of *bad* word alignments. For example, suppose the constituent *my long name* is node-aligned to its Urdu translation *mera lamba naam*. If only the word-pairs *my/mera* and *name/naam* are aligned, then the Giza-score for this node-alignment would be 2. If however, the English word *long* were (incorrectly) aligned under GIZA++ to some Urdu word outside the corresponding constituent, then the score would drop to 1. This score could even be negative if the number of *bad* alignments exceeds those that are *good*. Distribution Gz_{pair} provides a probability for these scores (up to some fixed absolute value).

For an unaligned node n with corresponding yield y, only *bad* GIZA++ alignments are possible, thus the Giza-score for these nodes will always be zero or negative. Distribution Gz_{node} provides a probability for these scores (down to some fixed value). We want our model to find tree alignments such that both aligned node pairs and unaligned nodes have high *Giza-score*.

3.4 Generative Process

Now we describe the stochastic process whereby the observed parallel sentences and their word-level alignments are generated, according to our model.

As the first step in the Bayesian generative process, all the multinomial parameters listed in the previous section are drawn from their conjugate priors — Dirichlet distributions of appropriate dimension. Then, each pair of word-aligned parallel sentences is generated through the following process:

1. A pair of binary trees T_1 and T_2 along with an alignment tree A are drawn according to $P(T_1, T_2, A)$. A is an alignment tree for T_1 and T_2 if it can be obtained by the following steps: First insert blank nodes (labeled by λ) into T_1 and T_2. Then permute the order of sibling nodes such that the two resulting trees T_1' and T_2' are identical in structure. Finally, overlay T_1' and T_2' to obtain A. We additionally require that A contain no extraneous nodes – that is no nodes with two blank labels (λ, λ). See Figure 1 for an example.

   We define the distribution $P(T_1, T_2, A)$ to be uniform over all pairs of binary trees and their alignments.

2. For each node in A of the form (n_1, λ) (i.e. nodes in T_1 left unaligned by A), draw

   (i) a constituent yield according to π_1^C,

(ii) a constituent context according to ϕ_1^C,

(iii) a Giza-score according to Gz_{node}.

3. For each node in A of the form (λ, n_2) (i.e. nodes in T_2 left unaligned by A), draw

   (i) a constituent yield according to π_2^C,

   (ii) a constituent context according to ϕ_2^C,

   (iii) a Giza-score according to Gz_{node}.

4. For each node in A of the form (n_1, n_2) (i.e. tree-aligned node pairs), draw

   (i) a pair of constituent yields (y_1, y_2) according to:

   $$\frac{\phi_1^C(y_1) \cdot \phi_2^C(y_2) \cdot \omega(y_1, y_2)}{Z} \quad (1)$$

   which is a product of experts combining the language specific context-yield distributions as well as the coupling distribution ω with normalization constant Z,

   (ii) a pair of contexts according to the appropriate language-specific parameters,

   (iii) a Giza-score according to Gz_{pair}.

5. For each span in T_i *not* dominated by a node (for each language $i \in \{1, 2\}$), draw a distituent yield according to π_i^D and a distituent context according to ϕ_i^D.

6. Draw actual word-level alignments consistent with the Giza-scores, according to a uniform distribution.

In the next section we turn to the problem of inference under this model when only the part-of-speech tag sequences of parallel sentences and their word-level alignments are observed.

3.5 Inference

Given a corpus of paired part-of-speech tag sequences $(\mathbf{s_1}, \mathbf{s_2})$ and their GIZA++ alignments $\mathbf{g}$, we would ideally like to predict the set of tree pairs $(\mathbf{T_1}, \mathbf{T_2})$ which have highest probability when conditioned on the observed data: $P(\mathbf{T_1}, \mathbf{T_2} | \mathbf{s_1}, \mathbf{s_2}, \mathbf{g})$. We could rewrite this by explicitly integrating over the yield, context, coupling, Giza-score parameters as well as the alignment trees. However, since maximizing this integral directly would be intractable, we resort to standard Markov chain sampling techniques. We use Gibbs sampling (Hastings, 1970) to draw trees for each sentence conditioned on those drawn for all other sentences. The samples form a Markov chain which is guaranteed to converge to the true joint distribution over all sentences.

In the monolingual setting, there is a well-known tree sampling algorithm (Johnson et al., 2007). This algorithm proceeds in top-down fashion by sampling individual split points using the marginal probabilities of all possible subtrees. These marginals can be efficiently pre-computed and form the "inside" table of the famous Inside-Outside algorithm. However, in our setting, trees come in pairs, and their joint probability crucially depends on their alignment.

For the i^{th} parallel sentence, we wish to jointly sample the pair of trees $(T_1, T_2)_i$ together with their alignment A_i. To do so directly would involve simultaneously marginalizing over all possible subtrees as well as all possible alignments between such subtrees when sampling upper-level split points. We know of no obvious algorithm for computing this marginal. We instead first sample the pair of trees $(T_1, T_2)_i$ from a simpler *proposal distribution* Q. Our proposal distribution assumes that *no* nodes of the two trees are aligned and therefore allows us to use the recursive top-down sampling algorithm mentioned above. After a new tree pair $T^* = (T_1^*, T_2^*)_i$ is drawn from Q, we accept the pair with the following probability:

$$\min\left\{1, \frac{P(T^*|\mathbf{T}_{-i}, \mathbf{A}_{-i})\ Q(T|\mathbf{T}_{-i}, \mathbf{A}_{-i})}{P(T|\mathbf{T}_{-i}, \mathbf{A}_{-i})\ Q(T^*|\mathbf{T}_{-i}, \mathbf{A}_{-i})}\right\}$$

where T is the previously sampled tree-pair for sentence i, P is the true model probability, and Q is the probability under the proposal distribution. This use of a tractable proposal distribution and acceptance ratio is known as the Metropolis-Hastings algorithm and it preserves the convergence guarantee of the Gibbs sampler (Hastings, 1970). To compute the terms $P(T^*|\mathbf{T}_{-i}, \mathbf{A}_{-i})$ and $P(T|\mathbf{T}_{-i}, \mathbf{A}_{-i})$ in the acceptance ratio above, we need to marginalize over all possible alignments between tree pairs.

Fortunately, for any given pair of trees T_1 and T_2 this marginalization can be computed using a dynamic program in time $O(|T_1||T_2|)$. Here we provide a very brief sketch. For every pair of nodes $n_1 \in T_1, n_2 \in T_2$, a table stores the marginal probability of the subtrees rooted at n_1 and n_2, respectively. A dynamic program builds this table from the bottom up: For each node pair n_1, n_2, we sum the probabilities of all local alignment configurations, each multiplied by the appro-

priate marginals already computed in the table for lower-level node pairs. This algorithm is an adaptation of the dynamic program presented in (Jiang et al., 1995) for finding minimum cost alignment trees (Fig. 5 of that publication).

Once a pair of trees (T_1, T_2) has been sampled, we can proceed to sample an alignment tree $A|T_1, T_2$.[2] We sample individual alignment decisions from the top down, at each step using the alignment marginals for the remaining subtrees (already computed using the afore-mentioned dynamic program). Once the triple (T_1, T_2, A) has been sampled, we move on to the next parallel sentence.

We avoid directly sampling parameter values, instead using the marginalized closed forms for multinomials with Dirichlet conjugate-priors using counts and hyperparameter pseudo-counts (Gelman et al., 2004). Note that in the case of yield pairs produced according to Distribution 1 (in step 4 of the generative process) conjugacy is technically broken, since the yield pairs are no longer produced by a single multinomial distribution. Nevertheless, we count the produced yields as if they had been generated separately by each of the distributions involved in the numerator of Distribution 1.

4 Experimental setup

We test our model on three corpora of bilingual parallel sentences: English-Korean, English-Urdu, and English-Chinese. Though the model is trained using parallel data, during testing it has access only to monolingual data. This set-up ensures that we are testing our model's ability to learn better parameters at training time, rather than its ability to exploit parallel data at test time. Following (Klein and Manning, 2002), we restrict our model to binary trees, though we note that the alignment trees do not follow this restriction.

Data The Penn Korean Treebank (Han et al., 2002) consists of 5,083 Korean sentences translated into English for the purposes of language training in a military setting. Both the Korean and English sentences are annotated with syntactic trees. We use the first 4,000 sentences for training and the last 1,083 sentences for testing. We note that in the Korean data, a separate tag is given for each morpheme. We simply concatenate all the morpheme tags given for each word and treat the concatenation as a single tag. This procedure results in 199 different tags. The English-Urdu parallel corpus[3] consists of 4,325 sentences from the first three sections of the Penn Treebank and their Urdu translations annotated at the part-of-speech level. The Urdu side of this corpus does not provide tree annotations so here we can test parse accuracy only on English. We use the remaining sections of the Penn Treebank for English testing. The English-Chinese treebank (Bies et al., 2007) consists of 3,850 Chinese newswire sentences translated into English. Both the English and Chinese sentences are annotated with parse trees. We use the first 4/5 for training and the final 1/5 for testing.

During preprocessing of the corpora we remove all punctuation marks and special symbols, following the setup in previous grammar induction work (Klein and Manning, 2002). To obtain lexical alignments between the parallel sentences we employ GIZA++ (Och and Ney, 2003). We use intersection alignments, which are one-to-one alignments produced by taking the intersection of one-to-many alignments in each direction. These one-to-one intersection alignments tend to have higher precision.

We initialize the trees by making uniform split decisions recursively from the top down for sentences in both languages. Then for each pair of parallel sentences we randomly sample an initial alignment tree for the two sampled trees.

Baseline We implement a Bayesian version of the CCM as a baseline. This model uses the same inference procedure as our bilingual model (Gibbs sampling). In fact, our model reduces to this Bayesian CCM when it is assumed that no nodes between the two parallel trees are ever aligned and when word-level alignments are ignored. We also reimplemented the original EM version of CCM and found virtually no difference in performance when using EM or Gibbs sampling. In both cases our implementation achieves F-measure in the range of 69-70% on WSJ10, broadly in line with the performance reported by Klein and Manning (2002).

Hyperparameters Klein (2005) reports using smoothing pseudo-counts of 2 for constituent

[2] Sampling the alignment tree is important, as it provides us with counts of aligned constituents for the coupling parameter.

[3] http://www.crulp.org

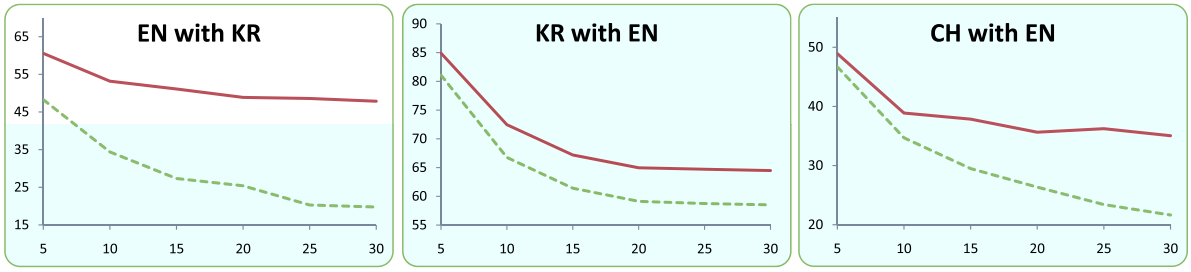


Figure 2: The F-measure of the CCM baseline (dotted line) and bilingual model (solid line) plotted on the y-axis, as the maximum sentence length in the test set is increased (x-axis). Results are averaged over all training scenarios given in Table 1.

yields and contexts and 8 for distituent yields and contexts. In our Bayesian model, these similar smoothing counts occur as the parameters of the Dirichlet priors. For Korean we found that the baseline performed well using these values. However, on our English and Chinese data, we found that somewhat higher smoothing values worked best, so we utilized values of 20 and 80 for constituent and distituent smoothing counts, respectively.

Our model additionally requires hyperparameter values for ω (the coupling distribution for aligned yields), Gz_{pair} and Gz_{node} (the distributions over Giza-scores for aligned nodes and unaligned nodes, respectively). For ω we used a symmetric Dirichlet prior with parameter 1. For Gz_{pair} and Gz_{node}, in order to create a strong bias towards high Giza-scores, we used non-symmetric Dirichlet priors. In both cases, we capped the absolute value of the scores at 3, to prevent count sparsity. In the case of Gz_{pair} we gave pseudo-counts of 1,000 for negative values and zero, and pseudo-counts of 1,000,000 for positive scores. For Gz_{node} we gave a pseudo-count of 1,000,000 for a score of zero, and 1,000 for all negative scores. This very strong prior bias encodes our intuition that syntactic alignments which respect lexical alignments should be preferred. Our method is not sensitive to these exact values and any reasonably strong bias gave similar results.

In all our experiments, we consider the hyperparameters fixed and observed values.

Testing and evaluation As mentioned above, we test our model only on monolingual data, where the parallel sentences are not provided to the model. To predict the bracketings of these monolingual test sentences, we take the smoothed counts accumulated in the final round of sampling over the training data and perform a maximum likelihood estimate of the monolingual CCM parameters. These parameters are then used to produce the highest probability bracketing of the test set.

To evaluate both our model as well as the baseline, we use (unlabeled) bracket precision, recall, and F-measure (Klein and Manning, 2002). Following previous work, we include the whole-sentence brackets but ignore single-word brackets. We perform experiments on different subsets of training and testing data based on the sentence-length. In particular we experimented with sentence length limits of 10, 20, and 30 for both the training and testing sets. We also report the upper bound on F-measure for binary trees. We average the results over 10 separate sampling runs.

5 Results

Table 1 reports the full results of our experiments. In all testing scenarios the bilingual model outperforms its monolingual counterpart in terms of both precision and recall. On average, the bilingual model gains 10.2 percentage points in precision, 7.7 in recall, and 8.8 in F-measure. The gap between monolingual performance and the binary tree upper bound is reduced by over 19%.

The extent of the gain varies across pairings. For instance, the smallest improvement is observed for English when trained with Urdu. The Korean-English pairing results in substantial improvements for Korean and quite large improvements for English, for which the absolute gain reaches 28 points in F-measure. In the case of Chinese and English, the gains for English are fairly minimal whereas those for Chinese are quite sub-

	Max Sent. Length		Monolingual			Bilingual			Upper Bound
	Test	Train	Precision	Recall	F1	Precision	Recall	F1	F1
EN with KR	10	10	52.74	39.53	45.19	57.76	43.30	49.50	85.6
		20	41.87	31.38	35.87	61.66	46.22	52.83	85.6
		30	33.43	25.06	28.65	64.41	48.28	**55.19**	85.6
	20	20	35.12	25.12	29.29	56.96	40.74	47.50	83.3
		30	26.26	18.78	21.90	60.07	42.96	**50.09**	83.3
	30	30	23.95	16.81	19.76	58.01	40.73	**47.86**	82.4
KR with EN	10	10	71.07	62.55	66.54	75.63	66.56	70.81	93.6
		20	71.35	62.79	66.80	77.61	68.30	72.66	93.6
		30	71.37	62.81	66.82	77.87	68.53	**72.91**	93.6
	20	20	64.28	54.73	59.12	70.44	59.98	64.79	91.9
		30	64.29	54.75	59.14	70.81	60.30	**65.13**	91.9
	30	30	63.63	54.17	58.52	70.11	59.70	**64.49**	91.9
EN with CH	10	10	50.09	34.18	40.63	37.46	25.56	30.39	81.0
		20	58.86	40.17	47.75	50.24	34.29	40.76	81.0
		30	64.81	44.22	52.57	68.24	46.57	**55.36**	81.0
	20	20	41.90	30.52	35.31	38.64	28.15	32.57	84.3
		30	52.83	38.49	44.53	58.50	42.62	**49.31**	84.3
	30	30	46.35	33.67	39.00	51.40	37.33	**43.25**	84.1
CH with EN	10	10	39.87	27.71	32.69	40.62	28.23	33.31	81.9
		20	43.44	30.19	35.62	47.54	33.03	38.98	81.9
		30	43.63	30.32	35.77	54.09	37.59	**44.36**	81.9
	20	20	29.80	23.46	26.25	36.93	29.07	32.53	88.0
		30	30.05	23.65	26.47	43.99	34.63	**38.75**	88.0
	30	30	24.46	19.41	21.64	39.61	31.43	**35.05**	88.4
EN with UR	10	10	57.98	45.68	51.10	73.43	57.85	64.71	88.1
		20	70.57	55.60	62.20	80.24	63.22	**70.72**	88.1
		30	75.39	59.40	66.45	79.04	62.28	69.67	88.1
	20	20	57.78	43.86	49.87	67.26	51.06	**58.05**	86.3
		30	63.12	47.91	54.47	64.45	48.92	55.62	86.3
	30	30	57.36	43.02	49.17	57.97	43.48	**49.69**	85.7

Table 1: Unlabeled precision, recall and F-measure for the monolingual baseline and the bilingual model on several test sets. We report results for different combinations of maximum sentence length in both the training and test sets. The right most column, in all cases, contains the maximum F-measure achievable using binary trees. The best performance for each test-length is highlighted in bold.

stantial. This asymmetry should not be surprising, as Chinese on its own seems to be quite a bit more difficult to parse than English.

We also investigated the impact of sentence length for both the training and testing sets. For our model, adding sentences of greater length to the training set leads to increases in parse accuracy for short sentences. For the baseline, however, adding this additional training data degrades performance in the case of English paired with Korean. Figure 2 summarizes the performance of our model for different sentence lengths on several of the test-sets. As shown in the figure, the largest improvements tend to occur at longer sentence lengths.

6 Conclusion

We have presented a probabilistic model for bilingual grammar induction which uses raw parallel text to learn tree pairs and their alignments. Our formalism loosely binds the two trees, using bilingual patterns when possible, but allowing substantial language-specific variation. We tested our model on three test sets and showed substantial improvement over a state-of-the-art monolingual baseline.[4]

[4]The authors acknowledge the support of the NSF (CAREER grant IIS-0448168, grant IIS-0835445, and grant IIS-0835652). Thanks to Amir Globerson and members of the MIT NLP group for their helpful suggestions. Any opinions, findings, or conclusions are those of the authors, and do not necessarily reflect the views of the funding organizations

References

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

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. Bayesian synchronous grammar induction. In *Proceedings of NIPS*.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proceedings of EMNLP*, pages 877–886.

Eugene Charniak and Glen Carroll. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In *Proceedings of the AAAI Workshop on Statistically-Based NLP Techniques*, pages 1–13.

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

Shay B. Cohen and Noah A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of the NAACL/HLT*.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *The Companion Volume to the Proceedings of the ACL*, pages 205–208.

Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. 2004. *Bayesian data analysis*. Chapman and Hall/CRC.

Dmitriy Genzel. 2005. Inducing a multilingual dictionary from a parallel multitext in related languages. In *Proceedings of EMNLP/HLT*, pages 875–882.

C. Han, N.R. Han, E.S. Ko, H. Yi, and M. Palmer. 2002. Penn Korean Treebank: Development and evaluation. In *Proc. Pacific Asian Conf. Language and Comp*.

W. K. Hastings. 1970. Monte carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109.

R. Hwa, P. Resnik, A. Weinberg, C. Cabezas, and O. Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Journal of Natural Language Engineering*, 11(3):311–325.

T. Jiang, L. Wang, and K. Zhang. 1995. Alignment of trees – an alternative to tree edit. *Theoretical Computer Science*, 143(1):137–148.

M. Johnson, T. Griffiths, and S. Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proceedings of the NAACL/HLT*, pages 139–146.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the ACL*, pages 128–135.

D. Klein. 2005. *The Unsupervised Learning of Natural Language Structure*. Ph.D. thesis, Stanford University.

Jonas Kuhn. 2004. Experiments in parallel-text based grammar induction. In *Proceedings of the ACL*, pages 470–477.

I. Dan Melamed. 2003. Multitext grammars and synchronous parsers. In *Proceedings of the NAACL/HLT*, pages 79–86.

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

Yoav Seginer. 2007. Fast unsupervised incremental parsing. In *Proceedings of the ACL*, pages 384–391.

David A. Smith and Noah A. Smith. 2004. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proceeding of EMNLP*, pages 49–56.

Benjamin Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of the ACL/HLT*, pages 737–745.

Benjamin Snyder, Tahira Naseem, Jacob Eisenstein, and Regina Barzilay. 2008. Unsupervised multilingual learning for POS tagging. In *Proceedings of EMNLP*, pages 1041–1050.

Benjamin Snyder, Tahira Naseem, Jacob Eisenstein, and Regina Barzilay. 2009. Adding more languages improves unsupervised multilingual part-of-speech tagging: A Bayesian non-parametric approach. In *Proceedings of the NAACL/HLT*.

Andreas Stolcke and Stephen M. Omohundro. 1994. Inducing probabilistic grammars by Bayesian model merging. In *Proceedings of ICGI*, pages 106–118.

Dekai Wu and Hongsing Wong. 1998. Machine translation with a stochastic grammatical channel. In *Proceedings of the ACL/COLING*, pages 1408–1415.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Chenhai Xi and Rebecca Hwa. 2005. A backoff model for bootstrapping resources for non-english languages. In *Proceedings of EMNLP*, pages 851 – 858.

Hao Zhang and Daniel Gildea. 2005. Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of the ACL*, pages 475–482.

Reinforcement Learning for Mapping Instructions to Actions

S.R.K. Branavan, Harr Chen, Luke S. Zettlemoyer, Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{branavan, harr, lsz, regina}@csail.mit.edu

Abstract

In this paper, we present a reinforcement learning approach for mapping natural language instructions to sequences of executable actions. We assume access to a reward function that defines the quality of the executed actions. During training, the learner repeatedly constructs action sequences for a set of documents, executes those actions, and observes the resulting reward. We use a policy gradient algorithm to estimate the parameters of a log-linear model for action selection. We apply our method to interpret instructions in two domains — Windows troubleshooting guides and game tutorials. Our results demonstrate that this method can rival supervised learning techniques while requiring few or no annotated training examples.[1]

1 Introduction

The problem of interpreting instructions written in natural language has been widely studied since the early days of artificial intelligence (Winograd, 1972; Di Eugenio, 1992). Mapping instructions to a sequence of executable actions would enable the automation of tasks that currently require human participation. Examples include configuring software based on how-to guides and operating simulators using instruction manuals. In this paper, we present a reinforcement learning framework for inducing mappings from text to actions without the need for annotated training examples.

For concreteness, consider instructions from a Windows troubleshooting guide on deleting temporary folders, shown in Figure 1. We aim to map this text to the corresponding low-level commands and parameters. For example, properly interpreting the third instruction requires clicking on a tab, finding the appropriate option in a tree control, and clearing its associated checkbox.

- Click start, point to search, and then click for files or folders.
- In the search results dialog box, on the tools menu, click folder options.
- In the folder options dialog box, on the view tab, under advanced settings, click *show hidden files and folders*, and then click to clear the *hide file extensions for known file types* check box.
- Click apply, and then click ok.
- In the search for files or folders named box, type msdownld.tmp.
- In the look in list, click my computer, and then click search now.
- In the search results pane, right-click msdownld.tmp and then click delete on the shortcut menu, a *confirm folder delete* message appears.
- Click yes.

Figure 1: A Windows troubleshooting article describing how to remove the "msdownld.tmp" temporary folder.

In this and many other applications, the validity of a mapping can be verified by executing the induced actions in the corresponding environment and observing their effects. For instance, in the example above we can assess whether the goal described in the instructions is achieved, i.e., the folder is deleted. The key idea of our approach is to leverage the validation process as the main source of supervision to guide learning. This form of supervision allows us to learn interpretations of natural language instructions when standard supervised techniques are not applicable, due to the lack of human-created annotations.

Reinforcement learning is a natural framework for building models using validation from an environment (Sutton and Barto, 1998). We assume that supervision is provided in the form of a reward function that defines the quality of executed actions. During training, the learner repeatedly constructs action sequences for a set of given documents, executes those actions, and observes the resulting reward. The learner's goal is to estimate a

[1] Code, data, and annotations used in this work are available at http://groups.csail.mit.edu/rbg/code/rl/

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 82–90,
Suntec, Singapore, 2-7 August 2009.

policy — a distribution over actions given instruction text and environment state — that maximizes future expected reward. Our policy is modeled in a log-linear fashion, allowing us to incorporate features of both the instruction text and the environment. We employ a policy gradient algorithm to estimate the parameters of this model.

We evaluate our method on two distinct applications: Windows troubleshooting guides and puzzle game tutorials. The key findings of our experiments are twofold. First, models trained only with simple reward signals achieve surprisingly high results, coming within 11% of a fully supervised method in the Windows domain. Second, augmenting unlabeled documents with even a small fraction of annotated examples greatly reduces this performance gap, to within 4% in that domain. These results indicate the power of learning from this new form of automated supervision.

2 Related Work

Grounded Language Acquisition Our work fits into a broader class of approaches that aim to learn language from a situated context (Mooney, 2008a; Mooney, 2008b; Fleischman and Roy, 2005; Yu and Ballard, 2004; Siskind, 2001; Oates, 2001). Instances of such approaches include work on inferring the meaning of words from video data (Roy and Pentland, 2002; Barnard and Forsyth, 2001), and interpreting the commentary of a simulated soccer game (Chen and Mooney, 2008). Most of these approaches assume some form of parallel data, and learn perceptual co-occurrence patterns. In contrast, our emphasis is on learning language by proactively interacting with an external environment.

Reinforcement Learning for Language Processing Reinforcement learning has been previously applied to the problem of dialogue management (Scheffler and Young, 2002; Roy et al., 2000; Litman et al., 2000; Singh et al., 1999). These systems converse with a human user by taking actions that emit natural language utterances. The reinforcement learning state space encodes information about the goals of the user and what they say at each time step. The learning problem is to find an optimal policy that maps states to actions, through a trial-and-error process of repeated interaction with the user.

Reinforcement learning is applied very differently in dialogue systems compared to our setup. In some respects, our task is more easily amenable to reinforcement learning. For instance, we are not interacting with a human user, so the cost of interaction is lower. However, while the state space can be designed to be relatively small in the dialogue management task, our state space is determined by the underlying environment and is typically quite large. We address this complexity by developing a policy gradient algorithm that learns efficiently while exploring a small subset of the states.

3 Problem Formulation

Our task is to learn a mapping between documents and the sequence of actions they express. Figure 2 shows how one example sentence is mapped to three actions.

Mapping Text to Actions As input, we are given a document d, comprising a sequence of sentences $(u_1, \ldots, u_\ell)$, where each u_i is a sequence of words. Our goal is to map d to a sequence of actions $\vec{a} = (a_0, \ldots, a_{n-1})$. Actions are predicted and executed sequentially.[2]

An action $a = (c, R, W')$ encompasses a *command* c, the command's *parameters* R, and the words W' specifying c and R. Elements of R refer to *objects* available in the *environment state*, as described below. Some parameters can also refer to words in document d. Additionally, to account for words that do not describe any actions, c can be a null command.

The Environment The environment state $\mathcal{E}$ specifies the set of objects available for interaction, and their properties. In Figure 2, $\mathcal{E}$ is shown on the right. The environment state $\mathcal{E}$ changes in response to the execution of command c with parameters R according to a transition distribution $p(\mathcal{E}'|\mathcal{E}, c, R)$. This distribution is *a priori* unknown to the learner. As we will see in Section 5, our approach avoids having to directly estimate this distribution.

State To predict actions sequentially, we need to track the state of the document-to-actions mapping over time. A *mapping state* s is a tuple $(\mathcal{E}, d, j, W)$, where $\mathcal{E}$ refers to the current environment state; j is the index of the sentence currently being interpreted in document d; and W contains words that were mapped by previous actions for

[2] That is, action a_i is executed before a_{i+1} is predicted.

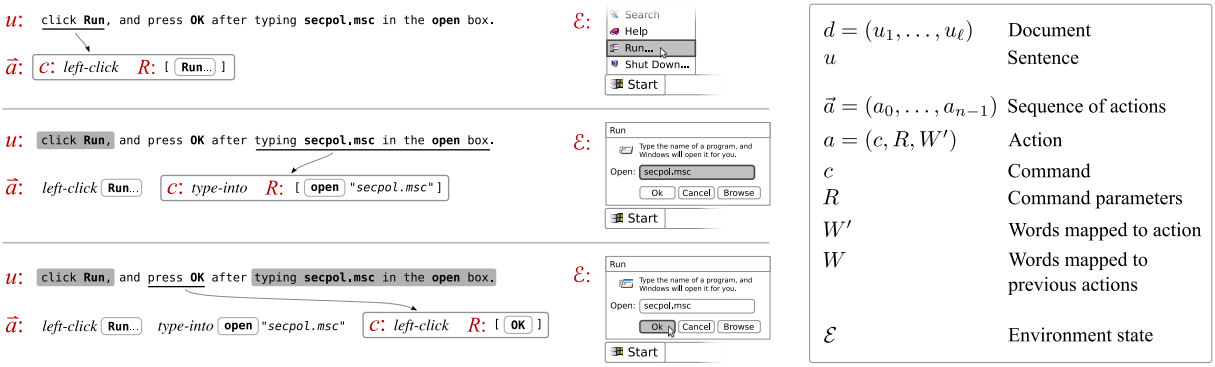


Figure 2: A three-step mapping from an instruction sentence to a sequence of actions in Windows 2000. For each step, the figure shows the words selected by the action, along with the corresponding system command and its parameters. The words of W' are underlined, and the words of W are highlighted in grey.

the same sentence. The mapping state s is observed after each action.

The initial mapping state s_0 for document d is $(\mathcal{E}_d, d, 0, \emptyset)$; $\mathcal{E}_d$ is the unique starting environment state for d. Performing action a in state $s = (\mathcal{E}, d, j, W)$ leads to a new state s' according to distribution $p(s'|s, a)$, defined as follows: $\mathcal{E}$ transitions according to $p(\mathcal{E}'|\mathcal{E}, c, R)$, W is updated with a's selected words, and j is incremented if all words of the sentence have been mapped. For the applications we consider in this work, environment state transitions, and consequently mapping state transitions, are deterministic.

Training During training, we are provided with a set D of documents, the ability to sample from the transition distribution, and a *reward function* $r(h)$. Here, $h = (s_0, a_0, \ldots, s_{n-1}, a_{n-1}, s_n)$ is a *history* of states and actions visited while interpreting one document. $r(h)$ outputs a real-valued score that correlates with correct action selection.[3] We consider both immediate reward, which is available after each action, and delayed reward, which does not provide feedback until the last action. For example, *task completion* is a delayed reward that produces a positive value after the final action only if the task was completed successfully. We will also demonstrate how manually annotated action sequences can be incorporated into the reward.

[3]In most reinforcement learning problems, the reward function is defined over state-action pairs, as $r(s, a)$ — in this case, $r(h) = \sum_t r(s_t, a_t)$, and our formulation becomes a standard finite-horizon Markov decision process. Policy gradient approaches allow us to learn using the more general case of history-based reward.

The goal of training is to estimate parameters θ of the action selection distribution $p(a|s, \theta)$, called the *policy*. Since the reward correlates with action sequence correctness, the θ that maximizes expected reward will yield the best actions.

4 A Log-Linear Model for Actions

Our goal is to predict a sequence of actions. We construct this sequence by repeatedly choosing an action given the current mapping state, and applying that action to advance to a new state.

Given a state $s = (\mathcal{E}, d, j, W)$, the space of possible next actions is defined by enumerating subspans of unused words in the current sentence (i.e., subspans of the jth sentence of d not in W), and the possible commands and parameters in environment state $\mathcal{E}$.[4] We model the policy distribution $p(a|s; \theta)$ over this action space in a log-linear fashion (Della Pietra et al., 1997; Lafferty et al., 2001), giving us the flexibility to incorporate a diverse range of features. Under this representation, the policy distribution is:

$$p(a|s; \theta) = \frac{e^{\theta \cdot \phi(s,a)}}{\sum_{a'} e^{\theta \cdot \phi(s,a')}}, \quad (1)$$

where $\phi(s, a) \in \mathbb{R}^n$ is an n-dimensional feature representation. During test, actions are selected according to the mode of this distribution.

[4]For parameters that refer to words, the space of possible values is defined by the unused words in the current sentence.

5 Reinforcement Learning

During training, our goal is to find the optimal policy $p(a|s;\theta)$. Since reward correlates with correct action selection, a natural objective is to maximize expected future reward — that is, the reward we expect while acting according to that policy from state s. Formally, we maximize the *value function*:

$$V_\theta(s) = E_{p(h|\theta)}\left[r(h)\right], \tag{2}$$

where the history h is the sequence of states and actions encountered while interpreting a single document $d \in D$. This expectation is averaged over all documents in D. The distribution $p(h|\theta)$ returns the probability of seeing history h when starting from state s and acting according to a policy with parameters θ. This distribution can be decomposed into a product over time steps:

$$p(h|\theta) = \prod_{t=0}^{n-1} p(a_t|s_t;\theta)p(s_{t+1}|s_t,a_t). \tag{3}$$

5.1 A Policy Gradient Algorithm

Our reinforcement learning problem is to find the parameters θ that maximize V_θ from equation 2. Although there is no closed form solution, *policy gradient* algorithms (Sutton et al., 2000) estimate the parameters θ by performing stochastic gradient ascent. The gradient of V_θ is approximated by interacting with the environment, and the resulting reward is used to update the estimate of θ. Policy gradient algorithms optimize a non-convex objective and are only guaranteed to find a local optimum. However, as we will see, they scale to large state spaces and can perform well in practice.

To find the parameters θ that maximize the objective, we first compute the derivative of V_θ. Expanding according to the product rule, we have:

$$\frac{\partial}{\partial\theta}V_\theta(s) = E_{p(h|\theta)}\left[r(h)\sum_t \frac{\partial}{\partial\theta}\log p(a_t|s_t;\theta)\right], \tag{4}$$

where the inner sum is over all time steps t in the current history h. Expanding the inner partial derivative we observe that:

$$\frac{\partial}{\partial\theta}\log p(a|s;\theta) = \phi(s,a) - \sum_{a'}\phi(s,a')p(a'|s;\theta), \tag{5}$$

which is the derivative of a log-linear distribution.

Equation 5 is easy to compute directly. However, the complete derivative of V_θ in equation 4 is intractable, because computing the expectation would require summing over all possible histories. Instead, policy gradient algorithms employ stochastic gradient ascent by computing a noisy estimate of the expectation using just a subset of the histories. Specifically, we draw samples from $p(h|\theta)$ by acting in the target environment, and use these samples to approximate the expectation in equation 4. In practice, it is often sufficient to sample a single history h for this approximation.

Input: A document set D,
Feature representation ϕ,
Reward function $r(h)$,
Number of iterations T

Initialization: Set θ to small random values.

1 **for** $i = 1 \dots T$ **do**
2 **foreach** $d \in D$ **do**
3 Sample history $h \sim p(h|\theta)$ where $h = (s_0, a_0, \dots, a_{n-1}, s_n)$ as follows:
3a **for** $t = 0 \dots n-1$ **do**
3b Sample action $a_t \sim p(a|s_t;\theta)$
3c Execute a_t on state s_t: $s_{t+1} \sim p(s|s_t,a_t)$
  end
4 $\Delta \leftarrow \sum_t \left(\phi(s_t,a_t) - \sum_{a'}\phi(s_t,a')p(a'|s_t;\theta)\right)$
5 $\theta \leftarrow \theta + r(h)\Delta$
  end
end

Output: Estimate of parameters θ

Algorithm 1: A policy gradient algorithm.

Algorithm 1 details the complete policy gradient algorithm. It performs T iterations over the set of documents D. Step 3 samples a history that maps each document to actions. This is done by repeatedly selecting actions according to the current policy, and updating the state by executing the selected actions. Steps 4 and 5 compute the empirical gradient and update the parameters θ.

In many domains, interacting with the environment is expensive. Therefore, we use two techniques that allow us to take maximum advantage of each environment interaction. First, a history $h = (s_0, a_0, \dots, s_n)$ contains subsequences $(s_i, a_i, \dots s_n)$ for $i = 1$ to $n-1$, each with its own reward value given by the environment as a side effect of executing h. We apply the update from equation 5 for each subsequence. Second, for a sampled history h, we can propose alternative histories h' that result in the same commands and parameters with different word spans. We can again apply equation 5 for each h', weighted by its probability under the current policy, $\frac{p(h'|\theta)}{p(h|\theta)}$.

The algorithm we have presented belongs to a family of policy gradient algorithms that have been successfully used for complex tasks such as robot control (Ng et al., 2003). Our formulation is unique in how it represents natural language in the reinforcement learning framework.

5.2 Reward Functions and ML Estimation

We can design a range of reward functions to guide learning, depending on the availability of annotated data and environment feedback. Consider the case when every training document $d \in D$ is annotated with its correct sequence of actions, and state transitions are deterministic. Given these examples, it is straightforward to construct a reward function that connects policy gradient to maximum likelihood. Specifically, define a reward function $r(h)$ that returns one when h matches the annotation for the document being analyzed, and zero otherwise. Policy gradient performs stochastic gradient ascent on the objective from equation 2, performing one update per document. For document d, this objective becomes:

$$E_{p(h|\theta)}[r(h)] = \sum_h r(h)p(h|\theta) = p(h_d|\theta),$$

where h_d is the history corresponding to the annotated action sequence. Thus, with this reward policy gradient is equivalent to stochastic gradient ascent with a maximum likelihood objective.

At the other extreme, when annotations are completely unavailable, learning is still possible given informative feedback from the environment. Crucially, this feedback only needs to correlate with action sequence quality. We detail environment-based reward functions in the next section. As our results will show, reward functions built using this kind of feedback can provide strong guidance for learning. We will also consider reward functions that combine annotated supervision with environment feedback.

6 Applying the Model

We study two applications of our model: following instructions to perform software tasks, and solving a puzzle game using tutorial guides.

6.1 Microsoft Windows Help and Support

On its Help and Support website,[5] Microsoft publishes a number of articles describing how to perform tasks and troubleshoot problems in the Windows operating systems. Examples of such tasks include installing patches and changing security settings. Figure 1 shows one such article.

[5] support.microsoft.com

Notation	
o	Parameter referring to an environment object
L	Set of object class names (e.g. "button")
V	Vocabulary
Features on W and object o	
Test if o is visible in s	
Test if o has input focus	
Test if o is in the foreground	
Test if o was previously interacted with	
Test if o came into existence since last action	
Min. edit distance between $w \in W$ and object labels in s	
Features on words in W, command c, and object o	
$\forall c' \in C, w \in V$: test if $c' = c$ and $w \in W$	
$\forall c' \in C, l \in L$: test if $c' = c$ and l is the class of o	

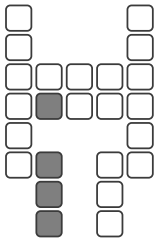
Table 1: Example features in the Windows domain. All features are binary, except for the normalized edit distance which is real-valued.

Our goal is to automatically execute these support articles in the Windows 2000 environment. Here, the environment state is the set of visible user interface (UI) objects, and object properties such as label, location, and parent window. Possible commands include *left-click*, *right-click*, *double-click*, and *type-into*, all of which take a UI object as a parameter; *type-into* additionally requires a parameter for the input text.

Table 1 lists some of the features we use for this domain. These features capture various aspects of the action under consideration, the current Windows UI state, and the input instructions. For example, one lexical feature measures the similarity of a word in the sentence to the UI labels of objects in the environment. Environment-specific features, such as whether an object is currently in focus, are useful when selecting the object to manipulate. In total, there are 4,438 features.

Reward Function Environment feedback can be used as a reward function in this domain. An obvious reward would be task completion (e.g., whether the stated computer problem was fixed). Unfortunately, verifying task completion is a challenging system issue in its own right.

Instead, we rely on a noisy method of checking whether execution can proceed from one sentence to the next: at least one word in each sentence has to correspond to an object in the envi-



Clear the bottom four from the second column from the left, the row of four, the second column from the right, the row of four, then the two columns.

Figure 3: Crossblock puzzle with tutorial. For this level, four squares in a row or column must be removed at once. The first move specified by the tutorial is greyed in the puzzle.

ronment.[6] For instance, in the sentence from Figure 2 the word "Run" matches the *Run...* menu item. If no words in a sentence match a current environment object, then one of the previous sentences was analyzed incorrectly. In this case, we assign the history a reward of -1. This reward is not guaranteed to penalize all incorrect histories, because there may be false positive matches between the sentence and the environment. When at least one word matches, we assign a positive reward that linearly increases with the percentage of words assigned to non-null commands, and linearly decreases with the number of output actions. This reward signal encourages analyses that interpret all of the words without producing spurious actions.

6.2 Crossblock: A Puzzle Game

Our second application is to a puzzle game called *Crossblock*, available online as a Flash game.[7] Each of 50 puzzles is played on a grid, where some grid positions are filled with squares. The object of the game is to clear the grid by drawing vertical or horizontal line segments that remove groups of squares. Each segment must exactly cross a specific number of squares, ranging from two to seven depending on the puzzle. Humans players have found this game challenging and engaging enough to warrant posting textual tutorials.[8] A sample puzzle and tutorial are shown in Figure 3.

The environment is defined by the state of the grid. The only command is *clear*, which takes a parameter specifying the orientation (*row* or *column*) and grid location of the line segment to be removed. The challenge in this domain is to segment the text into the phrases describing each action, and then correctly identify the line segments from references such as "the bottom four from the second column from the left."

For this domain, we use two sets of binary features on state-action pairs (s, a). First, for each vocabulary word w, we define a feature that is one if w is the last word of a's consumed words W'. These features help identify the proper text segmentation points between actions. Second, we introduce features for pairs of vocabulary word w and attributes of action a, e.g., the line orientation and grid locations of the squares that a would remove. This set of features enables us to match words (e.g., "row") with objects in the environment (e.g., a move that removes a horizontal series of squares). In total, there are 8,094 features.

Reward Function For Crossblock it is easy to directly verify task completion, which we use as the basis of our reward function. The reward $r(h)$ is -1 if h ends in a state where the puzzle cannot be completed. For solved puzzles, the reward is a positive value proportional to the percentage of words assigned to non-null commands.

7 Experimental Setup

Datasets For the Windows domain, our dataset consists of 128 documents, divided into 70 for training, 18 for development, and 40 for test. In the puzzle game domain, we use 50 tutorials, divided into 40 for training and 10 for test.[9] Statistics for the datasets are shown below.

	Windows	Puzzle
Total # of documents	128	50
Total # of words	5562	994
Vocabulary size	610	46
Avg. words per sentence	9.93	19.88
Avg. sentences per document	4.38	1.00
Avg. actions per document	10.37	5.86

The data exhibits certain qualities that make for a challenging learning problem. For instance, there are a surprising variety of linguistic constructs — as Figure 4 shows, in the Windows domain even a simple command is expressed in at least six different ways.

[6] We assume that a word maps to an environment object if the edit distance between the word and the object's name is below a threshold value.

[7] hexaditidom.deviantart.com/art/Crossblock-108669149

[8] www.jayisgames.com/archives/2009/01/crossblock.php

[9] For Crossblock, because the number of puzzles is limited, we did not hold out a separate development set, and report averaged results over five training/test splits.

```
On the tools menu, click internet options
Click tools, and then click internet options
Click tools, and then choose internet options
Click internet options on the tools menu
In internet explorer, click internet options on the tools menu
On the tools menu in internet explorer, click internet options
```

Figure 4: Variations of "click internet options on the tools menu" present in the Windows corpus.

Experimental Framework To apply our algorithm to the Windows domain, we use the Win32 application programming interface to simulate human interactions with the user interface, and to gather environment state information. The operating system environment is hosted within a virtual machine,[10] allowing us to rapidly save and reset system state snapshots. For the puzzle game domain, we replicated the game with an implementation that facilitates automatic play.

As is commonly done in reinforcement learning, we use a softmax temperature parameter to smooth the policy distribution (Sutton and Barto, 1998), set to 0.1 in our experiments. For Windows, the development set is used to select the best parameters. For Crossblock, we choose the parameters that produce the highest reward during training. During evaluation, we use these parameters to predict mappings for the test documents.

Evaluation Metrics For evaluation, we compare the results to manually constructed sequences of actions. We measure the number of correct actions, sentences, and documents. An action is correct if it matches the annotations in terms of command and parameters. A sentence is correct if all of its actions are correctly identified, and analogously for documents.[11] Statistical significance is measured with the sign test.

Additionally, we compute a word alignment score to investigate the extent to which the input text is used to construct correct analyses. This score measures the percentage of words that are aligned to the corresponding annotated actions in correctly analyzed documents.

Baselines We consider the following baselines to characterize the performance of our approach.

- **Full Supervision** Sequence prediction problems like ours are typically addressed using supervised techniques. We measure how a standard supervised approach would perform on this task by using a reward signal based on manual annotations of output action sequences, as defined in Section 5.2. As shown there, policy gradient with this reward is equivalent to stochastic gradient ascent with a maximum likelihood objective.

- **Partial Supervision** We consider the case when only a subset of training documents is annotated, and environment reward is used for the remainder. Our method seamlessly combines these two kinds of rewards.

- **Random and Majority (Windows)** We consider two naïve baselines. Both scan through each sentence from left to right. A command c is executed on the object whose name is encountered first in the sentence. This command c is either selected *randomly*, or set to the *majority* command, which is *left-click*. This procedure is repeated until no more words match environment objects.

- **Random (Puzzle)** We consider a baseline that *randomly* selects among the actions that are valid in the current game state.[12]

8 Results

Table 2 presents evaluation results on the test sets. There are several indicators of the difficulty of this task. The random and majority baselines' poor performance in both domains indicates that naïve approaches are inadequate for these tasks. The performance of the fully supervised approach provides further evidence that the task is challenging. This difficulty can be attributed in part to the large branching factor of possible actions at each step — on average, there are 27.14 choices per action in the Windows domain, and 9.78 in the Crossblock domain.

In both domains, the learners relying only on environment reward perform well. Although the fully supervised approach performs the best, adding just a few annotated training examples to the environment-based learner significantly reduces the performance gap.

[10] VMware Workstation, available at www.vmware.com

[11] In these tasks, each action depends on the correct execution of all previous actions, so a single error can render the remainder of that document's mapping incorrect. In addition, due to variability in document lengths, overall action accuracy is not guaranteed to be higher than document accuracy.

[12] Since action selection is among objects, there is no natural majority baseline for the puzzle.

	Windows				Puzzle		
	Action	Sent.	Doc.	Word	Action	Doc.	Word
Random baseline	0.128	0.101	0.000	—	0.081	0.111	—
Majority baseline	0.287	0.197	0.100	—	—	—	—
Environment reward	$*$ 0.647	$*$ 0.590	$*$ 0.375	0.819	$*$ 0.428	$*$ 0.453	0.686
Partial supervision	$\diamond$ 0.723	$*$ 0.702	0.475	0.989	0.575	$*$ 0.523	0.850
Full supervision	$\diamond$ 0.756	0.714	0.525	0.991	0.632	0.630	0.869

Table 2: Performance on the test set with different reward signals and baselines. Our evaluation measures the proportion of correct actions, sentences, and documents. We also report the percentage of correct word alignments for the successfully completed documents. Note the puzzle domain has only single-sentence documents, so its sentence and document scores are identical. The partial supervision line refers to 20 out of 70 annotated training documents for Windows, and 10 out of 40 for the puzzle. Each result marked with $*$ or $\diamond$ is a statistically significant improvement over the result immediately above it; $*$ indicates $p < 0.01$ and $\diamond$ indicates $p < 0.05$.

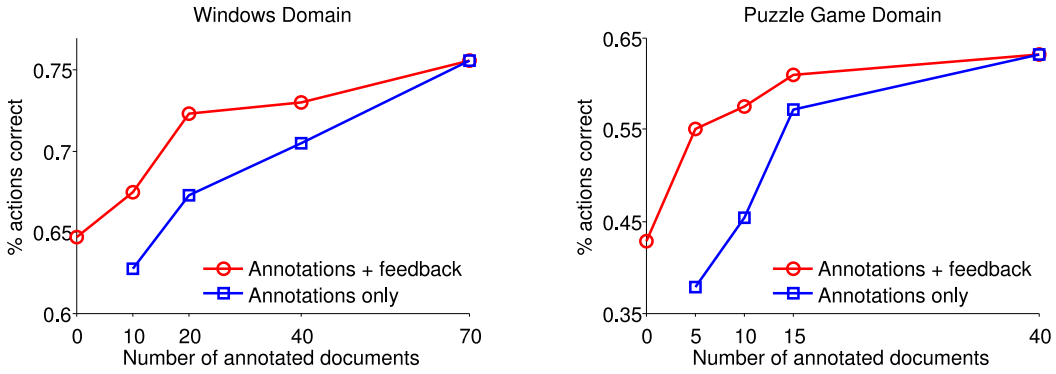


Figure 5: Comparison of two training scenarios where training is done using a subset of annotated documents, with and without environment reward for the remaining unannotated documents.

Figure 5 shows the overall tradeoff between annotation effort and system performance for the two domains. The ability to make this tradeoff is one of the advantages of our approach. The figure also shows that augmenting annotated documents with additional environment-reward documents invariably improves performance.

The word alignment results from Table 2 indicate that the learners are mapping the correct words to actions for documents that are successfully completed. For example, the models that perform best in the Windows domain achieve nearly perfect word alignment scores.

To further assess the contribution of the instruction text, we train a variant of our model without access to text features. This is possible in the game domain, where all of the puzzles share a single goal state that is independent of the instructions. This variant solves 34% of the puzzles, suggesting that access to the instructions significantly improves performance.

9 Conclusions

In this paper, we presented a reinforcement learning approach for inducing a mapping between instructions and actions. This approach is able to use environment-based rewards, such as task completion, to learn to analyze text. We showed that having access to a suitable reward function can significantly reduce the need for annotations.

Acknowledgments

The authors acknowledge the support of the NSF (CAREER grant IIS-0448168, grant IIS-0835445, grant IIS-0835652, and a Graduate Research Fellowship) and the ONR. Thanks to Michael Collins, Amir Globerson, Tommi Jaakkola, Leslie Pack Kaelbling, Dina Katabi, Martin Rinard, and members of the MIT NLP group for their suggestions and comments. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

Kobus Barnard and David A. Forsyth. 2001. Learning the semantics of words and pictures. In *Proceedings of ICCV*.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of ICML*.

Stephen Della Pietra, Vincent J. Della Pietra, and John D. Lafferty. 1997. Inducing features of random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(4):380–393.

Barbara Di Eugenio. 1992. Understanding natural language instructions: the case of purpose clauses. In *Proceedings of ACL*.

Michael Fleischman and Deb Roy. 2005. Intentional context in situated language learning. In *Proceedings of CoNLL*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Diane J. Litman, Michael S. Kearns, Satinder Singh, and Marilyn A. Walker. 2000. Automatic optimization of dialogue management. In *Proceedings of COLING*.

Raymond J. Mooney. 2008a. Learning language from its perceptual context. In *Proceedings of ECML/PKDD*.

Raymond J. Mooney. 2008b. Learning to connect language and perception. In *Proceedings of AAAI*.

Andrew Y. Ng, H. Jin Kim, Michael I. Jordan, and Shankar Sastry. 2003. Autonomous helicopter flight via reinforcement learning. In *Advances in NIPS*.

James Timothy Oates. 2001. *Grounding knowledge in sensors: Unsupervised learning for language and planning*. Ph.D. thesis, University of Massachusetts Amherst.

Deb K. Roy and Alex P. Pentland. 2002. Learning words from sights and sounds: a computational model. *Cognitive Science 26*, pages 113–146.

Nicholas Roy, Joelle Pineau, and Sebastian Thrun. 2000. Spoken dialogue management using probabilistic reasoning. In *Proceedings of ACL*.

Konrad Scheffler and Steve Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of HLT*.

Satinder P. Singh, Michael J. Kearns, Diane J. Litman, and Marilyn A. Walker. 1999. Reinforcement learning for spoken dialogue systems. In *Advances in NIPS*.

Jeffrey Mark Siskind. 2001. Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *J. Artif. Intell. Res. (JAIR)*, 15:31–90.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. The MIT Press.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in NIPS*.

Terry Winograd. 1972. *Understanding Natural Language*. Academic Press.

Chen Yu and Dana H. Ballard. 2004. On the integration of grounding language and learning objects. In *Proceedings of AAAI*.

Learning Semantic Correspondences with Less Supervision

Percy Liang
UC Berkeley
pliang@cs.berkeley.edu

Michael I. Jordan
UC Berkeley
jordan@cs.berkeley.edu

Dan Klein
UC Berkeley
klein@cs.berkeley.edu

Abstract

A central problem in grounded language acquisition is learning the correspondences between a rich world state and a stream of text which references that world state. To deal with the high degree of ambiguity present in this setting, we present a generative model that simultaneously segments the text into utterances and maps each utterance to a meaning representation grounded in the world state. We show that our model generalizes across three domains of increasing difficulty—Robocup sportscasting, weather forecasts (a new domain), and NFL recaps.

1 Introduction

Recent work in learning semantics has focused on mapping sentences to meaning representations (e.g., some logical form) given aligned sentence/meaning pairs as training data (Ge and Mooney, 2005; Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007; Lu et al., 2008). However, this degree of supervision is unrealistic for modeling human language acquisition and can be costly to obtain for building large-scale, broad-coverage language understanding systems.

A more flexible direction is grounded language acquisition: learning the meaning of sentences in the context of an observed world state. The grounded approach has gained interest in various disciplines (Siskind, 1996; Yu and Ballard, 2004; Feldman and Narayanan, 2004; Gorniak and Roy, 2007). Some recent work in the NLP community has also moved in this direction by relaxing the amount of supervision to the setting where each sentence is paired with a small set of candidate meanings (Kate and Mooney, 2007; Chen and Mooney, 2008).

The goal of this paper is to reduce the amount of supervision even further. We assume that we are given a *world state* represented by a set of *records* along with a *text*, an unsegmented sequence of words. For example, in the weather forecast domain (Section 2.2), the text is the weather report, and the records provide a structured representation of the temperature, sky conditions, etc.

In this less restricted data setting, we must resolve multiple ambiguities: (1) the segmentation of the text into *utterances*; (2) the identification of relevant *facts*, i.e., the choice of records and aspects of those records; and (3) the alignment of utterances to facts (facts are the meaning representations of the utterances). Furthermore, in some of our examples, much of the world state is not referenced at all in the text, and, conversely, the text references things which are not represented in our world state. This increased amount of ambiguity and noise presents serious challenges for learning. To cope with these challenges, we propose a probabilistic generative model that treats text segmentation, fact identification, and alignment in a single unified framework. The parameters of this hierarchical hidden semi-Markov model can be estimated efficiently using EM.

We tested our model on the task of aligning text to records in three different domains. The first domain is Robocup sportscasting (Chen and Mooney, 2008). Their best approach (KRISPER) obtains 67% F_1; our method achieves 76.5%. This domain is simplified in that the segmentation is known. The second domain is weather forecasts, for which we created a new dataset. Here, the full complexity of joint segmentation and alignment arises. Nonetheless, we were able to obtain reasonable results on this task. The third domain we considered is NFL recaps (Barzilay and Lapata, 2005; Snyder and Barzilay, 2007). The language used in this domain is richer by orders of magnitude, and much of it does *not* reference the world state. Nonetheless, taking the first unsupervised approach to this problem, we were able to make substantial progress: We achieve an F_1 of 53.2%, which closes over half of the gap between a heuristic baseline (26%) and supervised systems (68%–80%).

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 91–99,
Suntec, Singapore, 2-7 August 2009.

Dataset	# scenarios	$\lvert\mathbf{w}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathbf{s}\rvert$	$\lvert\mathcal{A}\rvert$
Robocup	1919	5.7	9	2.4	0.8
Weather	22146	28.7	12	36.0	5.8
NFL	78	969.0	44	329.0	24.3

Table 1: Statistics for the three datasets. We report average values across all scenarios in the dataset: $\lvert\mathbf{w}\rvert$ is the number of words in the text, $\lvert\mathcal{T}\rvert$ is the number of record types, $\lvert\mathbf{s}\rvert$ is the number of records, and $\lvert\mathcal{A}\rvert$ is the number of gold alignments.

2 Domains and Datasets

Our goal is to learn the correspondence between a text $\mathbf{w}$ and the world state $\mathbf{s}$ it describes. We use the term *scenario* to refer to such a $(\mathbf{w}, \mathbf{s})$ pair.

The *text* is simply a sequence of words $\mathbf{w} = (w_1, \ldots, w_{|\mathbf{w}|})$. We represent the world state $\mathbf{s}$ as a set of *records*, where each *record* $r \in \mathbf{s}$ is described by a *record type* $r.t \in \mathcal{T}$ and a tuple of *field values* $r.\mathbf{v} = (r.v_1, \ldots, r.v_m)$.[1] For example, temperature is a record type in the weather domain, and it has four fields: time, min, mean, and max.

The record type $r.t \in \mathcal{T}$ specifies the *field type* $r.t_f \in \{\text{INT}, \text{STR}, \text{CAT}\}$ of each field value $r.v_f$, $f = 1, \ldots, m$. There are three possible field types—integer (INT), string (STR), and categorical (CAT)—which are assumed to be known and fixed. Integer fields represent numeric properties of the world such as temperature, string fields represent surface-level identifiers such as names of people, and categorical fields represent discrete concepts such as score types in football (touchdown, field goal, and safety). The field type determines the way we expect the field value to be rendered in words: integer fields can be numerically perturbed, string fields can be spliced, and categorical fields are represented by open-ended word distributions, which are to be learned. See Section 3.3 for details.

2.1 Robocup Sportscasting

In this domain, a Robocup simulator generates the state of a soccer game, which is represented by a set of event records. For example, the record pass(arg1=`pink1`,arg2=`pink5`) denotes a passing event; this type of record has two fields: arg1 (the actor) and arg2 (the recipient). As the game is progressing, humans interject commentaries about notable events in the game, e.g., *pink1 passes back to pink5 near the middle of the field*. All of the fields in this domain are categorical, which means there is no a priori association between the field value `pink1` and the word *pink1*. This degree of flexibility is desirable because *pink1* is sometimes referred to as *pink goalie*, a mapping which does not arise from string operations but must instead be learned.

We used the dataset created by Chen and Mooney (2008), which contains 1919 scenarios from the 2001–2004 Robocup finals. Each scenario consists of a single sentence representing a fragment of a commentary on the game, paired with a set of candidate records. In the annotation, each sentence corresponds to at most one record (possibly one not in the candidate set, in which case we automatically get that sentence wrong). See Figure 1(a) for an example and Table 1 for summary statistics on the dataset.

2.2 Weather Forecasts

In this domain, the world state contains detailed information about a local weather forecast and the text is a short forecast report (see Figure 1(b) for an example). To create the dataset, we collected local weather forecasts for 3,753 cities in the US (those with population at least 10,000) over three days (February 7–9, 2009) from `www.weather.gov`. For each city and date, we created two scenarios, one for the day forecast and one for the night forecast. The forecasts consist of hour-by-hour measurements of temperature, wind speed, sky cover, chance of rain, etc., which represent the underlying world state.

This world state is summarized by records which aggregate measurements over selected time intervals. For example, one of the records states the minimum, average, and maximum temperature from 5pm to 6am. This aggregation process produced 22,146 scenarios, each containing $|\mathbf{s}| = 36$ multi-field records. There are 12 record types, each consisting of only integer and categorical fields.

To annotate the data, we split the text by punctuation into *lines* and labeled each line with the records to which the line refers. These lines are used only for evaluation and are not part of the model (see Section 5.1 for further discussion).

The weather domain is more complex than the Robocup domain in several ways: The text $\mathbf{w}$ is longer, there are more candidate records, and most notably, $\mathbf{w}$ references multiple records (5.8 on av-

[1] To simplify notation, we assume that each record has m fields, though in practice, m depends on the record type $r.t$.

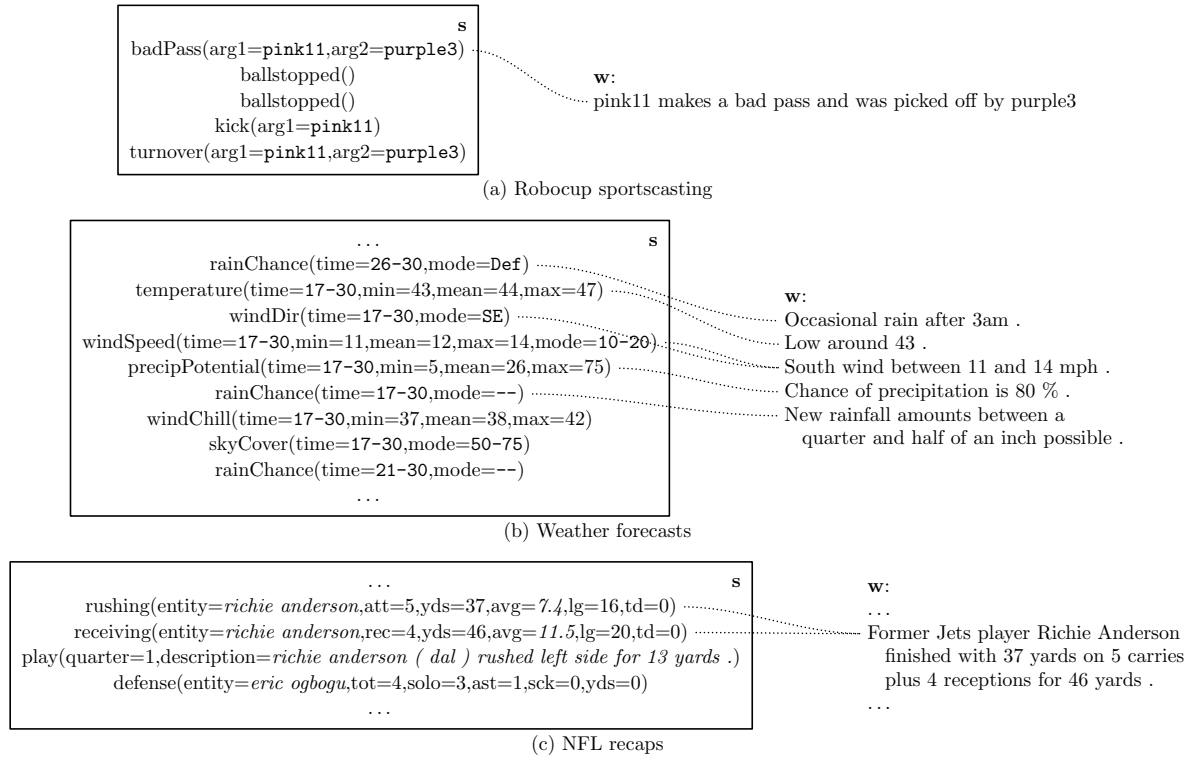


Figure 1: An example of a scenario for each of the three domains. Each scenario consists of a candidate set of records $\mathbf{s}$ and a text $\mathbf{w}$. Each record is specified by a record type (e.g., badPass) and a set of field values. Integer values are in Roman, string values are in *italics*, and categorical values are in `typewriter`. The gold alignments are shown.

erage), so the segmentation of $\mathbf{w}$ is unknown. See Table 1 for a comparison of the two datasets.

2.3 NFL Recaps

In this domain, each scenario represents a single NFL football game (see Figure 1(c) for an example). The world state (the things that happened during the game) is represented by database tables, e.g., scoring summary, team comparison, drive chart, play-by-play, etc. Each record is a database entry, for instance, the receiving statistics for a certain player. The text is the recap of the game—an article summarizing the game highlights. The dataset we used was collected by Barzilay and Lapata (2005). The data includes 466 games during the 2003–2004 NFL season. 78 of these games were annotated by Snyder and Barzilay (2007), who aligned each sentence to a set of records.

This domain is by far the most complicated of the three. Many records corresponding to inconsequential game statistics are not mentioned. Conversely, the text contains many general remarks (e.g., *it was just that type of game*) which are not present in any of the records. Furthermore, the complexity of the language used in the recap is far greater than what we can represent using our simple model. Fortunately, most of the fields are integer fields or string fields (generally names or brief descriptions), which provide important anchor points for learning the correspondences. Nonetheless, the same names and numbers occur in multiple records, so there is still uncertainty about which record is referenced by a given sentence.

3 Generative Model

To learn the correspondence between a text $\mathbf{w}$ and a world state $\mathbf{s}$, we propose a generative model $p(\mathbf{w} \mid \mathbf{s})$ with latent variables specifying this correspondence.

Our model combines segmentation with alignment. The segmentation aspect of our model is similar to that of Grenager et al. (2005) and Eisenstein and Barzilay (2008), but in those two models, the segments are clustered into topics rather than grounded to a world state. The alignment aspect of our model is similar to the HMM model for word alignment (Ney and Vogel, 1996). DeNero et al. (2008) perform joint segmentation and word alignment for machine translation, but the nature of that task is different from ours.

The model is defined by a generative process,

which proceeds in three stages (Figure 2 shows the corresponding graphical model):

1. Record choice: choose a sequence of records $\mathbf{r} = (r_1, \ldots, r_{|\mathbf{r}|})$ to describe, where each $r_i \in \mathbf{s}$.
2. Field choice: for each chosen record r_i, select a sequence of fields $\mathbf{f}_i = (f_{i1}, \ldots, f_{i|\mathbf{f}_i|})$, where each $f_{ij} \in \{1, \ldots, m\}$.
3. Word choice: for each chosen field f_{ij}, choose a number $c_{ij} > 0$ and generate a sequence of c_{ij} words.

The observed text $\mathbf{w}$ is the terminal yield formed by concatenating the sequences of words of all fields generated; note that the segmentation of $\mathbf{w}$ provided by $\mathbf{c} = \{c_{ij}\}$ is latent. Think of the words spanned by a record as constituting an utterance with a meaning representation given by the record and subset of fields chosen.

Formally, our probabilistic model places a distribution over $(\mathbf{r}, \mathbf{f}, \mathbf{c}, \mathbf{w})$ and factorizes according to the three stages as follows:

$$p(\mathbf{r}, \mathbf{f}, \mathbf{c}, \mathbf{w} \mid \mathbf{s}) = p(\mathbf{r} \mid \mathbf{s}) p(\mathbf{f} \mid \mathbf{r}) p(\mathbf{c}, \mathbf{w} \mid \mathbf{r}, \mathbf{f}, \mathbf{s})$$

The following three sections describe each of these stages in more detail.

3.1 Record Choice Model

The record choice model specifies a distribution over an ordered sequence of records $\mathbf{r} = (r_1, \ldots, r_{|\mathbf{r}|})$, where each record $r_i \in \mathbf{s}$. This model is intended to capture two types of regularities in the discourse structure of language. The first is *salience*, that is, some record types are simply more prominent than others. For example, in the NFL domain, 70% of scoring records are mentioned whereas only 1% of punting records are mentioned. The second is the idea of local *coherence*, that is, the order in which one mentions records tend to follow certain patterns. For example, in the weather domain, the sky conditions are generally mentioned first, followed by temperature, and then wind speed.

To capture these two phenomena, we define a Markov model on the record types (and given the record type, a record is chosen uniformly from the set of records with that type):

$$p(\mathbf{r} \mid \mathbf{s}) = \prod_{i=1}^{|\mathbf{r}|} p(r_i.t \mid r_{i-1}.t) \frac{1}{|\mathbf{s}(r_i.t)|}, \qquad (1)$$

where $\mathbf{s}(t) \stackrel{\text{def}}{=} \{r \in \mathbf{s} : r.t = t\}$ and $r_0.t$ is a dedicated START record type.[2] We also model the transition of the final record type to a designated STOP record type in order to capture regularities about the types of records which are described last. More sophisticated models of coherence could also be employed here (Barzilay and Lapata, 2008).

We assume that $\mathbf{s}$ includes a special *null record* whose type is NULL, responsible for generating parts of our text which do not refer to any real records.

3.2 Field Choice Model

Each record type $t \in \mathcal{T}$ has a separate field choice model, which specifies a distribution over a sequence of fields. We want to capture salience and coherence at the field level like we did at the record level. For instance, in the weather domain, the minimum and maximum fields of a temperature record are mentioned whereas the average is not. In the Robocup domain, the actor typically precedes the recipient in passing event records.

Formally, we have a Markov model over the fields:[3]

$$p(\mathbf{f} \mid \mathbf{r}) = \prod_{i=1}^{|\mathbf{r}|} \prod_{j=1}^{|\mathbf{f}_j|} p(f_{ij} \mid f_{i(j-1)}). \qquad (2)$$

Each record type has a dedicated *null field* with its own multinomial distribution over words, intended to model words which refer to that record type in general (e.g., the word *passes* for passing records). We also model transitions into the first field and transitions out of the final field with special START and STOP fields. This Markov structure allows us to capture a few elements of rudimentary syntax.

3.3 Word Choice Model

We arrive at the final component of our model, which governs how the information about a particular field of a record is rendered into words. For each field f_{ij}, we generate the number of words c_{ij} from a uniform distribution over $\{1, 2, \ldots, C_{\text{max}}\}$, where C_{max} is set larger than the length of the longest text we expect to see. Conditioned on

[2] We constrain our inference to only consider record types t that occur in $\mathbf{s}$, i.e., $\mathbf{s}(t) \neq \emptyset$.

[3] During inference, we prohibit consecutive fields from repeating.

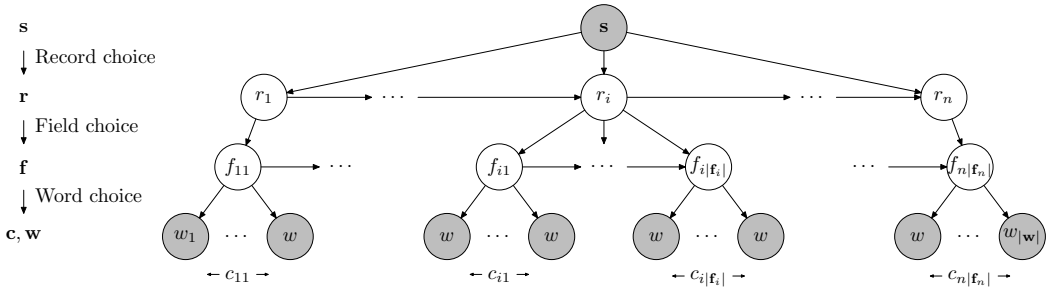


Figure 2: Graphical model representing the generative model. First, records are chosen and ordered from the set $\mathbf{s}$. Then fields are chosen for each record. Finally, words are chosen for each field. The world state $\mathbf{s}$ and the words $\mathbf{w}$ are observed, while $(\mathbf{r}, \mathbf{f}, \mathbf{c})$ are latent variables to be inferred (note that the number of latent variables itself is unknown).

the fields $\mathbf{f}$, the words $\mathbf{w}$ are generated independently:[4]

$$p(\mathbf{w} \mid \mathbf{r}, \mathbf{f}, \mathbf{c}, \mathbf{s}) = \prod_{k=1}^{|\mathbf{w}|} p_{\mathrm{W}}(w_k \mid r(k).t_{f(k)}, r(k).v_{f(k)}),$$

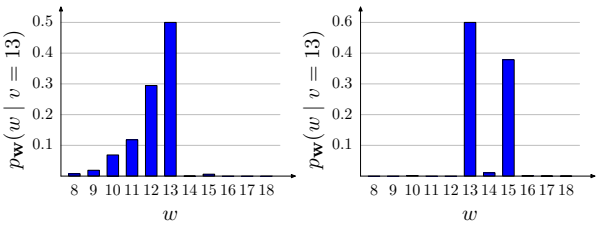
where $r(k)$ and $f(k)$ are the record and field responsible for generating word w_k, as determined by the segmentation $\mathbf{c}$. The word choice model $p_{\mathrm{W}}(w \mid t, v)$ specifies a distribution over words given the field type t and field value v. This distribution is a mixture of a global backoff distribution over words and a field-specific distribution which depends on the field type t.

Although we designed our word choice model to be relatively general, it is undoubtedly influenced by the three domains. However, we can readily extend or replace it with an alternative if desired; this modularity is one principal benefit of probabilistic modeling.

Integer Fields ($t = \textsc{int}$) For integer fields, we want to capture the intuition that a numeric quantity v is rendered in the text as a word which is possibly some other numerical value w due to stylistic factors. Sometimes the exact value v is used (e.g., in reporting football statistics). Other times, it might be customary to round v (e.g., wind speeds are typically rounded to a multiple of 5). In other cases, there might just be some unexplained error, where w deviates from v by some noise $\epsilon_+ = w - v > 0$ or $\epsilon_- = v - w > 0$. We model ϵ_+ and ϵ_- as geometric distributions.[5] In summary, we allow six possible ways of generating the word w given v:

$$v \quad \lceil v \rceil_5 \quad \lfloor v \rfloor_5 \quad \text{round}_5(v) \quad v - \epsilon_- \quad v + \epsilon_+$$

Separate probabilities for choosing among these possibilities are learned for each field type (see Figure 3 for an example).



(a) temperature.min (b) windSpeed.min

Figure 3: Two integer field types in the weather domain for which we learn different distributions over the ways in which a value v might appear in the text as a word w. Suppose the record field value is $v = 13$. Both distributions are centered around v, as is to be expected, but the two distributions have different shapes: For temperature.min, almost all the mass is to the left, suggesting that forecasters tend to report conservative lower bounds. For the wind speed, the mass is concentrated on 13 and 15, suggesting that forecasters frequently round wind speeds to multiples of 5.

String Fields ($t = \textsc{str}$) Strings fields are intended to represent values which we expect to be realized in the text via a simple surface-level transformation. For example, a name field with value $v =$ *Moe Williams* is sometimes referenced in the text by just *Williams*. We used a simple generic model of rendering string fields: Let w be a word chosen uniformly from those in v.

Categorical Fields ($t = \textsc{cat}$) Unlike string fields, categorical fields are not tied down to any lexical representation; in fact, the identities of the categorical field values are irrelevant. For each categorical field f and possible value v, we have a

[4] While a more sophisticated model of words would be useful if we intended to use this model for natural language generation, the false independence assumptions present here matter less for the task of learning the semantic correspondences because we always condition on $\mathbf{w}$.

[5] Specifically, $p(\epsilon_+; \alpha_+) = (1 - \alpha_+)^{\epsilon_+ - 1}\alpha_+$, where α_+ is a field-specific parameter; $p(\epsilon_-; \alpha_-)$ is defined analogously.

v	$p_{\mathrm{W}}(w \mid t, v)$
0-25	, clear mostly sunny
25-50	partly , cloudy increasing
50-75	mostly cloudy , partly
75-100	of inch an possible new a rainfall

Table 2: Highest probability words for the categorical field skyCover.mode in the weather domain. It is interesting to note that skyCover=75-100 is so highly correlated with rain that the model learns to connect an overcast sky in the world to the indication of rain in the text.

separate multinomial distribution over words from which w is drawn. An example of a categorical field is skyCover.mode in the weather domain, which has four values: 0-25, 25-50, 50-75, and 75-100. Table 2 shows the top words for each of these field values learned by our model.

4 Learning and Inference

Our learning and inference methodology is a fairly conventional application of Expectation Maximization (EM) and dynamic programming. The input is a set of scenarios $\mathcal{D}$, each of which is a text $\mathbf{w}$ paired with a world state $\mathbf{s}$. We maximize the marginal likelihood of our data, summing out the latent variables $(\mathbf{r}, \mathbf{f}, \mathbf{c})$:

$$\max_{\theta} \prod_{(\mathbf{w},\mathbf{s})\in\mathcal{D}} \sum_{\mathbf{r},\mathbf{f},\mathbf{c}} p(\mathbf{r}, \mathbf{f}, \mathbf{c}, \mathbf{w} \mid \mathbf{s}; \theta), \qquad (3)$$

where θ are the parameters of the model (all the multinomial probabilities). We use the EM algorithm to maximize (3), which alternates between the E-step and the M-step. In the E-step, we compute expected counts according to the posterior $p(\mathbf{r}, \mathbf{f}, \mathbf{c} \mid \mathbf{w}, \mathbf{s}; \theta)$. In the M-step, we optimize the parameters θ by normalizing the expected counts computed in the E-step. In our experiments, we initialized EM with a uniform distribution for each multinomial and applied add-0.1 smoothing to each multinomial in the M-step.

As with most complex discrete models, the bulk of the work is in computing expected counts under $p(\mathbf{r}, \mathbf{f}, \mathbf{c} \mid \mathbf{w}, \mathbf{s}; \theta)$. Formally, our model is a hierarchical hidden semi-Markov model conditioned on $\mathbf{s}$. Inference in the E-step can be done using a dynamic program similar to the inside-outside algorithm.

5 Experiments

Two important aspects of our model are the segmentation of the text and the modeling of the coherence structure at both the record and field levels. To quantify the benefits of incorporating these two aspects, we compare our full model with two simpler variants.

- Model 1 (no model of segmentation or coherence): Each record is chosen independently; each record generates one field, and each field generates one word. This model is similar in spirit to IBM model 1 (Brown et al., 1993).
- Model 2 (models segmentation but not coherence): Records and fields are still generated independently, but each field can now generate multiple words.
- Model 3 (our full model of segmentation and coherence): Records and fields are generated according to the Markov chains described in Section 3.

5.1 Evaluation

In the annotated data, each text $\mathbf{w}$ has been divided into a set of lines. These lines correspond to clauses in the weather domain and sentences in the Robocup and NFL domains. Each line is annotated with a (possibly empty) set of records. Let $\mathcal{A}$ be the gold set of these line-record alignment pairs.

To evaluate a learned model, we compute the Viterbi segmentation and alignment ($\mathrm{argmax}_{\mathbf{r},\mathbf{f},\mathbf{c}}\, p(\mathbf{r}, \mathbf{f}, \mathbf{c} \mid \mathbf{w}, \mathbf{s})$). We produce a predicted set of line-record pairs $\mathcal{A}'$ by aligning a line to a record r_i if the span of (the utterance corresponding to) r_i overlaps the line. The reason we evaluate indirectly using lines rather than using utterances is that it is difficult to annotate the segmentation of text into utterances in a simple and consistent manner.

We compute standard precision, recall, and F_1 of $\mathcal{A}'$ with respect to $\mathcal{A}$. Unless otherwise specified, performance is reported on all scenarios, which were also used for training. However, we did not tune any hyperparameters, but rather used generic values which worked well enough across all three domains.

5.2 Robocup Sportscasting

We ran 10 iterations of EM on Models 1–3. Table 3 shows that performance improves with increased model sophistication. We also compare

Method	Precision	Recall	F_1
Model 1	**78.6**	61.9	69.3
Model 2	74.1	**84.1**	78.8
Model 3	77.3	84.0	**80.5**

Table 3: Alignment results on the Robocup sportscasting dataset.

Method	F_1
Random baseline	48.0
Chen and Mooney (2008)	67.0
Model 3	**75.7**

Table 4: F_1 scores based on the 4-fold cross-validation scheme in Chen and Mooney (2008).

Method	Precision	Recall	F_1
Model 1	49.9	75.1	60.0
Model 2	67.3	70.4	68.8
Model 3	**76.3**	**73.8**	**75.0**

Table 5: Alignment results on the weather forecast dataset.

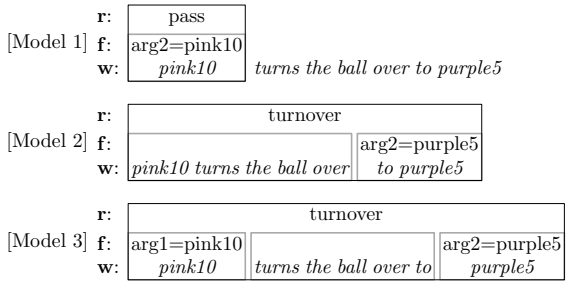


Figure 4: An example of predictions made by each of the three models on the Robocup dataset.

our model to the results of Chen and Mooney (2008) in Table 4.

Figure 4 provides a closer look at the predictions made by each of our three models for a particular example. Model 1 easily mistakes *pink10* for the recipient of a pass record because decisions are made independently for each word. Model 2 chooses the correct record, but having no model of the field structure inside a record, it proposes an incorrect field segmentation (although our evaluation is insensitive to this). Equipped with the ability to prefer a coherent field sequence, Model 3 fixes these errors.

Many of the remaining errors are due to the garbage collection phenomenon familiar from word alignment models (Moore, 2004; Liang et al., 2006). For example, the ballstopped record occurs frequently but is never mentioned in the text. At the same time, there is a correlation between ballstopped and utterances such as *pink2 holds onto the ball*, which are not aligned to any record in the annotation. As a result, our model incorrectly chooses to align the two.

5.3 Weather Forecasts

For the weather domain, staged training was necessary to get good results. For Model 1, we ran 15 iterations of EM. For Model 2, we ran 5 iterations of EM on Model 1, followed by 10 iterations on Model 2. For Model 3, we ran 5 iterations of Model 1, 5 iterations of a simplified variant of Model 3 where records were chosen independently, and finally, 5 iterations of Model 3. When going from one model to another, we used the final posterior distributions of the former to initialize the parameters of the latter.[6] We also prohibited utterances in Models 2 and 3 from crossing punctuation during inference.

Table 5 shows that performance improves substantially in the more sophisticated models, the gains being greater than in the Robocup domain. Figure 5 shows the predictions of the three models on an example. Model 1 is only able to form isolated (but not completely inaccurate) associations. By modeling segmentation, Model 2 accounts for the intermediate words, but errors are still made due to the lack of Markov structure. Model 3 remedies this. However, unexpected structures are sometimes learned. For example, the temperature.time=`6-21` field indicates daytime, which happens to be perfectly correlated with the word *high*, although *high* intuitively should be associated with the temperature.max field. In these cases of high correlation (Table 2 provides another example), it is very difficult to recover the proper alignment without additional supervision.

5.4 NFL Recaps

In order to scale up our models to the NFL domain, we first pruned for each sentence the records which have either no numerical values (e.g., 23, 23-10, 2/4) nor name-like words (e.g., those that appear only capitalized in the text) in common. This eliminated all but 1.5% of the record candidates per sentence, while maintaining an ora-

[6] It is interesting to note that this type of staged training is evocative of language acquisition in children: lexical associations are formed (Model 1) before higher-level discourse structure is learned (Model 3).

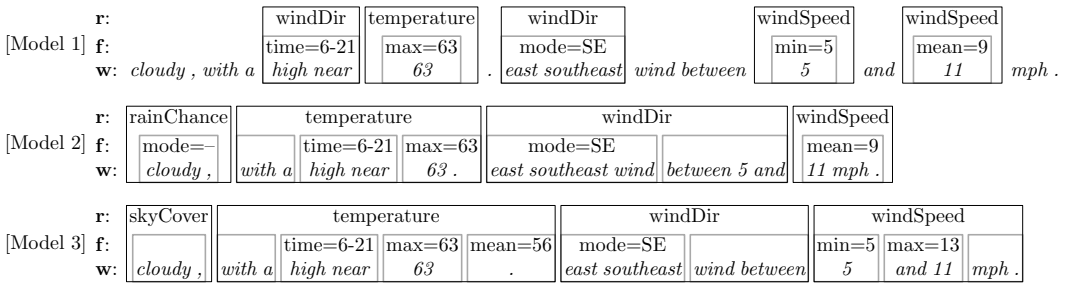


Figure 5: An example of predictions made by each of the three models on the weather dataset.

cle alignment F_1 score of 88.7. Guessing a single random record for each sentence yields an F_1 of 12.0. A reasonable heuristic which uses weighted number- and string-matching achieves 26.7.

Due to the much greater complexity of this domain, Model 2 was easily misled as it tried without success to find a coherent segmentation of the fields. We therefore created a variant, Model 2', where we constrained each field to generate exactly one word. To train Model 2', we ran 5 iterations of EM where each sentence is assumed to have exactly one record, followed by 5 iterations where the constraint was relaxed to also allow record boundaries at punctuation and the word *and*. We did not experiment with Model 3 since the discourse structure on records in this domain is not at all governed by a simple Markov model on record types—indeed, most regions do not refer to any records at all. We also fixed the backoff probability to 0.1 instead of learning it and enforced zero numerical deviation on integer field values.

Model 2' achieved an F_1 of 39.9, an improvement over Model 1, which attained 32.8. Inspection of the errors revealed the following problem: The alignment task requires us to sometimes align a sentence to multiple redundant records (e.g., play and score) referenced by the same part of the text. However, our model generates each part of text from only one record, and thus it can only allow an alignment to one record.[7] To cope with this incompatibility between the data and our notion of semantics, we used the following solution: We divided the records into three groups by type: play, score, and other. Each group has a copy of the model, but we enforce that they share the same segmentation. We also introduce a potential that couples the presence or absence of records across groups on the same segment to capture regular co-occurrences between redundant records.

[7]The model can align a sentence to multiple records provided that the records are referenced by non-overlapping parts of the text.

Method	Precision	Recall	F_1
Random (with pruning)	13.1	11.0	12.0
Baseline	29.2	24.6	26.7
Model 1	25.2	46.9	32.8
Model 2'	43.4	37.0	39.9
Model 2' (with groups)	46.5	62.1	53.2
Graph matching (sup.)	73.4	64.5	68.6
Multilabel global (sup.)	87.3	74.5	80.3

Table 6: Alignment results on the NFL dataset. Graph matching and multilabel are supervised results reported in Snyder and Barzilay (2007).[9]

Table 6 shows our results. With groups, we achieve an F_1 of 53.2. Though we still trail supervised techniques, which attain numbers in the 68–80 range, we have made substantial progress over our baseline using an unsupervised method. Furthermore, our model provides a more detailed analysis of the correspondence between the world state and text, rather than just producing a single alignment decision. Most of the remaining errors made by our model are due to a lack of calibration. Sometimes, our false positives are close calls where a sentence indirectly references a record, and our model predicts the alignment whereas the annotation standard does not. We believe that further progress is possible with a richer model.

6 Conclusion

We have presented a generative model of correspondences between a world state and an unsegmented stream of text. By having a joint model of salience, coherence, and segmentation, as well as a detailed rendering of the values in the world state into words in the text, we are able to cope with the increased ambiguity that arises in this new data setting, successfully pushing the limits of unsupervision.

References

R. Barzilay and M. Lapata. 2005. Collective content selection for concept-to-text generation. In *Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 331–338, Vancouver, B.C.

R. Barzilay and M. Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34:1–34.

P. F. Brown, S. A. D. Pietra, V. J. D. Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.

D. L. Chen and R. J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *International Conference on Machine Learning (ICML)*, pages 128–135. Omnipress.

J. DeNero, A. Bouchard-Côté, and D. Klein. 2008. Sampling alignment structure under a Bayesian translation model. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 314–323, Honolulu, HI.

J. Eisenstein and R. Barzilay. 2008. Bayesian unsupervised topic segmentation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 334–343.

J. Feldman and S. Narayanan. 2004. Embodied meaning in a neural theory of language. *Brain and Language*, 89:385–392.

R. Ge and R. J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Computational Natural Language Learning (CoNLL)*, pages 9–16, Ann Arbor, Michigan.

P. Gorniak and D. Roy. 2007. Situated language understanding as filtering perceived affordances. *Cognitive Science*, 31:197–231.

T. Grenager, D. Klein, and C. D. Manning. 2005. Unsupervised learning of field segmentation models for information extraction. In *Association for Computational Linguistics (ACL)*, pages 371–378, Ann Arbor, Michigan. Association for Computational Linguistics.

R. J. Kate and R. J. Mooney. 2007. Learning language semantics from ambiguous supervision. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 895–900, Cambridge, MA. MIT Press.

P. Liang, B. Taskar, and D. Klein. 2006. Alignment by agreement. In *North American Association for Computational Linguistics (NAACL)*, pages 104–111, New York City. Association for Computational Linguistics.

W. Lu, H. T. Ng, W. S. Lee, and L. S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 783–792.

R. C. Moore. 2004. Improving IBM word alignment model 1. In *Association for Computational Linguistics (ACL)*, pages 518–525, Barcelona, Spain. Association for Computational Linguistics.

H. Ney and S. Vogel. 1996. HMM-based word alignment in statistical translation. In *International Conference on Computational Linguistics (COLING)*, pages 836–841. Association for Computational Linguistics.

J. M. Siskind. 1996. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, 61:1–38.

B. Snyder and R. Barzilay. 2007. Database-text alignment via structured multilabel classification. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1713–1718, Hyderabad, India.

C. Yu and D. H. Ballard. 2004. On the integration of grounding language and learning objects. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 488–493, Cambridge, MA. MIT Press.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*, pages 658–666.

L. S. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 678–687.

Bayesian Unsupervised Word Segmentation with Nested Pitman-Yor Language Modeling

Daichi Mochihashi Takeshi Yamada Naonori Ueda
NTT Communication Science Laboratories
Hikaridai 2-4, Keihanna Science City, Kyoto, Japan
{daichi,yamada,ueda}@cslab.kecl.ntt.co.jp

Abstract

In this paper, we propose a new Bayesian model for fully unsupervised word segmentation and an efficient blocked Gibbs sampler combined with dynamic programming for inference. Our model is a nested hierarchical Pitman-Yor language model, where Pitman-Yor spelling model is embedded in the word model. We confirmed that it significantly outperforms previous reported results in both phonetic transcripts and standard datasets for Chinese and Japanese word segmentation. Our model is also considered as a way to construct an accurate word n-gram language model directly from characters of arbitrary language, without any "word" indications.

1 Introduction

"Word" is no trivial concept in many languages. Asian languages such as Chinese and Japanese have no explicit word boundaries, thus word segmentation is a crucial first step when processing them. Even in western languages, valid "words" are often not identical to space-separated tokens. For example, proper nouns such as "United Kingdom" or idiomatic phrases such as "with respect to" actually function as a single word, and we often condense them into the virtual words "UK" and "w.r.t.".

In order to extract "words" from text streams, unsupervised word segmentation is an important research area because the criteria for creating supervised training data could be arbitrary, and will be suboptimal for applications that rely on segmentations. It is particularly difficult to create "correct" training data for speech transcripts, colloquial texts, and classics where segmentations are often ambiguous, let alone is impossible for unknown languages whose properties computational linguists might seek to uncover.

From a scientific point of view, it is also interesting because it can shed light on how children learn "words" without the explicitly given boundaries for every word, which is assumed by supervised learning approaches.

Lately, model-based methods have been introduced for unsupervised segmentation, in particular those based on Dirichlet processes on words (Goldwater et al., 2006; Xu et al., 2008). This maximizes the probability of word segmentation $\mathbf{w}$ given a string s :

$$\hat{\mathbf{w}} = \operatorname*{argmax}_{\mathbf{w}} p(\mathbf{w}|s)\,. \qquad (1)$$

This approach often implicitly includes heuristic criteria proposed so far[1], while having a clear statistical semantics to find the most probable word segmentation that will maximize the probability of the data, here the strings.

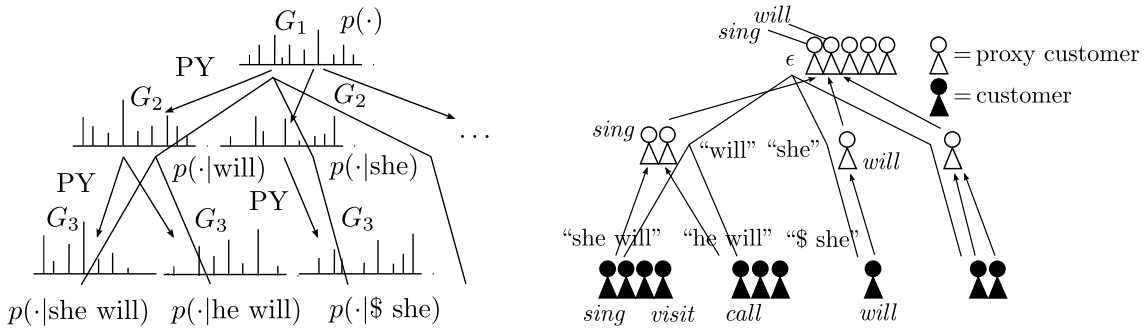
However, they are still naïve with respect to word spellings, and the inference is very slow owing to inefficient Gibbs sampling. Crucially, since they rely on sampling a word boundary between two neighboring words, they can leverage only up to bigram word dependencies.

In this paper, we extend this work to propose a more efficient and accurate unsupervised word segmentation that will optimize the performance of the word n-gram Pitman-Yor (i.e. Bayesian Kneser-Ney) language model, with an accurate character ∞-gram Pitman-Yor spelling model embedded in word models. Furthermore, it can be viewed as a method for building a high-performance n-gram language model directly from character strings of arbitrary language. It is carefully smoothed and has no "unknown words" problem, resulting from its model structure.

This paper is organized as follows. In Section 2,

[1] For instance, TANGO algorithm (Ando and Lee, 2003) essentially finds segments such that character n-gram probabilities are maximized blockwise, averaged over n.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 100–108,
Suntec, Singapore, 2-7 August 2009.



(a) Generating n-gram distributions G hierarchically from the Pitman-Yor process. Here, $n = 3$.

(b) Equivalent representation using a hierarchical Chinese Restaurant process. Each word in a training text is a "customer" shown in italic, and added to the leaf of its two words context.

Figure 1: Hierarchical Pitman-Yor Language Model.

we briefly describe a language model based on the Pitman-Yor process (Teh, 2006b), which is a generalization of the Dirichlet process used in previous research. By embedding a character n-gram in word n-gram from a Bayesian perspective, Section 3 introduces a novel language model for word segmentation, which we call the Nested Pitman-Yor language model. Section 4 describes an efficient blocked Gibbs sampler that leverages dynamic programming for inference. In Section 5 we describe experiments on the standard datasets in Chinese and Japanese in addition to English phonetic transcripts, and semi-supervised experiments are also explored. Section 6 is a discussion and Section 7 concludes the paper.

2 Pitman-Yor process and n-gram models

To compute a probability $p(\mathbf{w}|s)$ in (1), we adopt a Bayesian language model lately proposed by (Teh, 2006b; Goldwater et al., 2005) based on the Pitman-Yor process, a generalization of the Dirichlet process. As we shall see, this is a Bayesian theory of the best-performing Kneser-Ney smoothing of n-grams (Kneser and Ney, 1995), allowing an integrated modeling from a Bayesian perspective as persued in this paper.

The Pitman-Yor (PY) process is a stochastic process that generates discrete probability distribution G that is similar to another distribution G_0, called a *base measure*. It is written as

$$G \sim \mathrm{PY}(G_0, d, \theta)\,, \tag{2}$$

where d is a discount factor and θ controls how similar G is to G_0 on average.

Suppose we have a unigram word distribution $G_1 = \{\, p(\cdot)\,\}$ where $\cdot$ ranges over each word in the lexicon. The bigram distribution $G_2 = \{\, p(\cdot|v)\,\}$ given a word v is different from G_1, but will be similar to G_1 especially for high frequency words. Therefore, we can generate G_2 from a PY process of base measure G_1, as $G_2 \sim \mathrm{PY}(G_1, d, \theta)$. Similarly, trigram distribution $G_3 = \{\, p(\cdot|v'v)\,\}$ given an additional word v' is generated as $G_3 \sim \mathrm{PY}(G_2, d, \theta)$, and G_1, G_2, G_3 will form a tree structure shown in Figure 1(a).

In practice, we cannot observe G directly because it will be infinite dimensional distribution over the possible words, as we shall see in this paper. However, when we integrate out G it is known that Figure 1(a) can be represented by an equivalent hierarchical Chinese Restaurant Process (CRP) (Aldous, 1985) as in Figure 1(b).

In this representation, each n-gram context h (including the null context ϵ for unigrams) is a Chinese restaurant whose customers are the n-gram counts $c(w|h)$ seated over the tables $1 \cdots t_{hw}$. The seatings has been incrementally constructed by choosing the table k for each count in $c(w|h)$ with probability proportional to

$$\begin{cases} c_{hwk} - d & (k = 1, \cdots, t_{hw}) \\ \theta + d{\cdot}t_{h\cdot} & (k = new)\,, \end{cases} \tag{3}$$

where c_{hwk} is the number of customers seated at table k thus far and $t_{h\cdot} = \sum_w t_{hw}$ is the total number of tables in h. When $k = new$ is selected, t_{hw} is incremented, and this means that the count was actually generated from the shorter context h'. Therefore, in that case a proxy customer is sent to the parent restaurant and this process will recurse.

For example, if we have a sentence "she will sing" in the training data for trigrams, we add each word "she" "will" "sing" "$" as a customer to its two preceding words context node, as described in Figure 1(b). Here, "$" is a special token representing a sentence boundary in language model-

ing (Brown et al., 1992).

As a result, the n-gram probability of this hierarchical Pitman-Yor language model (HPYLM) is recursively computed as

$$p(w|h) = \frac{c(w|h) - d \cdot t_{hw}}{\theta + c(h)} + \frac{\theta + d \cdot t_{h\cdot}}{\theta + c(h)} p(w|h'), \quad (4)$$

where $p(w|h')$ is the same probability using a $(n-1)$-gram context h'. When we set $t_{hw} \equiv 1$, (4) recovers a Kneser-Ney smoothing: thus a HPYLM is a Bayesian Kneser-Ney language model as well as an extension of the hierarchical Dirichlet Process (HDP) used in Goldwater et al. (2006). θ, d are hyperparameters that can be learned as Gamma and Beta posteriors, respectively, given the data. For details, see Teh (2006a).

The inference of this model interleaves adding and removing a customer to optimize t_{hw}, d, and θ using MCMC. However, in our case "words" are not known a priori: the next section describes how to accomplish this by constructing a nested HPYLM of words and characters, with the associated inference algorithm.

3 Nested Pitman-Yor Language Model

Thus far we have assumed that the unigram G_1 is already given, but of course it should also be generated as $G_1 \sim \mathrm{PY}(G_0, d, \theta)$.

Here, a problem occurs: What should we use for G_0, namely the prior probabilities over words[2]? If a lexicon is finite, we can use a uniform prior $G_0(w) = 1/|V|$ for every word w in lexicon V. However, with word segmentation every substring could be a word, thus the lexicon is not limited but will be countably infinite.

Building an accurate G_0 is crucial for word segmentation, since it determines how the possible words will look like. Previous work using a Dirichlet process used a relatively simple prior for G_0, namely an uniform distribution over characters (Goldwater et al., 2006), or a prior solely dependent on word length with a Poisson distribution whose parameter is fixed by hand (Xu et al., 2008).

In contrast, in this paper we use a simple but more elaborate model, that is, a character n-gram language model that also employs HPYLM. This is important because in English, for example, words are likely to end in '–tion' and begin with

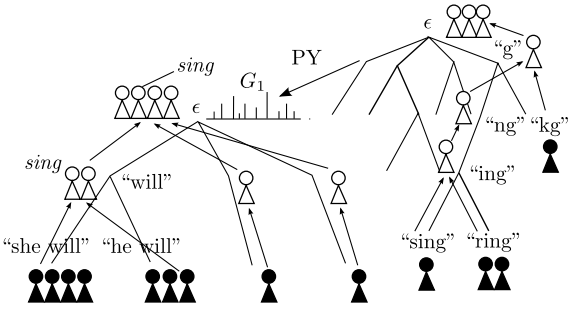


Figure 2: Chinese restaurant representation of our Nested Pitman-Yor Language Model (NPYLM).

're–', but almost never end in '–tio' nor begin with 'sre–' [3].

Therefore, we use

$$G_0(w) = p(c_1 \cdots c_k) \quad (5)$$

$$= \prod_{i=1}^{k} p(c_i | c_1 \cdots c_{i-1}) \quad (6)$$

where string $c_1 \cdots c_k$ is a spelling of w, and $p(c_i|c_1 \cdots c_{i-1})$ is given by the character HPYLM according to (4).

This language model, which we call Nested Pitman-Yor Language Model (NPYLM) hereafter, is the hierarchical language model shown in Figure 2, where the character HPYLM is embedded as a base measure of the word HPYLM.[4] As the final base measure for the character HPYLM, we used a uniform prior over the possible characters of a given language. To avoid dependency on n-gram order n, we actually used the ∞-gram language model (Mochihashi and Sumita, 2007), a variable order HPYLM, for characters. However, for generality we hereafter state that we used the HPYLM. The theory remains the same for ∞-grams, except sampling or marginalizing over n as needed.

Furthermore, we corrected (5) so that word length will have a Poisson distribution whose parameter can now be estimated for a given language and word type. We describe this in detail in Section 4.3.

Chinese Restaurant Representation

In our NPYLM, the word model and the character model are not separate but connected through a nested CRP. When a word w is generated from its parent at the unigram node, it means that w

[2]Note that this is different from unigrams, which are posterior distribution given data.

[3]Imagine we try to segment an English character string "*itisrecognizedasthe*$\cdots$."

[4]Strictly speaking, this is not "nested" in the sense of a Nested Dirichlet process (Rodriguez et al., 2008) and could be called "hierarchical HPYLM", which denotes another model for domain adaptation (Wood and Teh, 2008).

is drawn from the base measure, namely a character HPYLM. Then we divide w into characters $c_1 \cdots c_k$ to yield a "sentence" of characters and feed this into the character HPYLM as data.

Conversely, when a table becomes empty, this means that the data associated with the table are no longer valid. Therefore we remove the corresponding customers from the character HPYLM using the inverse procedure of adding a customer in Section 2.

All these processes will be invoked when a string is segmented into "words" and customers are added to the leaves of the word HPYLM. To segment a string into "words", we used efficient dynamic programming combined with MCMC, as described in the next section.

4 Inference

To find the hidden word segmentation $\mathbf{w}$ of a string $s = c_1 \cdots c_N$, which is equivalent to the vector of binary hidden variables $\mathbf{z} = z_1 \cdots z_N$, the simplest approach is to build a Gibbs sampler that randomly selects a character c_i and draw a binary decision z_i as to whether there is a word boundary, and then update the language model according to the new segmentation (Goldwater et al., 2006; Xu et al., 2008). When we iterate this procedure sufficiently long, it becomes a sample from the true distribution (1) (Gilks et al., 1996).

However, this sampler is too inefficient since time series data such as word segmentation have a very high correlation between neighboring words. As a result, the sampler is extremely slow to converge. In fact, (Goldwater et al., 2006) reports that the sampler would not mix without annealing, and the experiments needed 20,000 times of sampling for every character in the training data.

Furthermore, it has an inherent limitation that it cannot deal with larger than bigrams, because it uses only *local* statistics between directly contiguous words for word segmentation.

4.1 Blocked Gibbs sampler

Instead, we propose a sentence-wise Gibbs sampler of word segmentation using efficient dynamic programming, as shown in Figure 3.

In this algorithm, first we randomly select a string, and then remove the "sentence" data of its word segmentation from the NPYLM. Sampling a new segmentation, we update the NPYLM by adding a new "sentence" according to the new segmentation. When we repeat this process, it is expected to mix rapidly because it implicitly considers all possible segmentations of the given string at the same time.

```
for j = 1 ··· J do
  for s in randperm (s_1, ··· , s_D) do
    if j > 1 then
      Remove customers of w(s) from Θ
    end if
    Draw w(s) according to p(w|s, Θ)
    Add customers of w(s) to Θ
  end for
  Sample hyperparameters of Θ
end for
```

Figure 3: Blocked Gibbs Sampler of NPYLM Θ.

This is called a *blocked* Gibbs sampler that samples $\mathbf{z}$ block-wise for each sentence. It has an additional advantage in that we can accommodate higher-order relationships than bigrams, particularly trigrams, for word segmentation. [5]

4.2 Forward-Backward inference

Then, how can we sample a segmentation $\mathbf{w}$ for each string s? In accordance with the Forward filtering Backward sampling of HMM (Scott, 2002), this is achieved by essentially the same algorithm employed to sample a PCFG parse tree within MCMC (Johnson et al., 2007) and grammar-based segmentation (Johnson and Goldwater, 2009).

Forward Filtering. For this purpose, we maintain a forward variable $\alpha[t][k]$ in the bigram case. $\alpha[t][k]$ is the probability of a string $c_1 \cdots c_t$ with the final k characters being a word (see Figure 4). Segmentations before the final k characters are marginalized using the following recursive relationship:

$$\alpha[t][k] = \sum_{j=1}^{t-k} p(c_{t-k+1}^{t} | c_{t-k-j+1}^{t-k}) \cdot \alpha[t-k][j] \quad (7)$$

where $\alpha[0][0] = 1$ and we wrote $c_n \cdots c_m$ as c_n^m.[6] The rationale for (7) is as follows. Since maintaining binary variables $z_1, \cdots, z_N$ is equivalent to maintaining a distance to the nearest backward

[5] In principle fourgrams or beyond are also possible, but will be too complex while the gain will be small. For this purpose, Particle MCMC (Doucet et al., 2009) is promising but less efficient in a preliminary experiment.

[6] As Murphy (2002) noted, in semi-HMM we cannot use a standard trick to avoid underflow by normalizing $\alpha[t][k]$ into $p(k|t)$, since the model is asynchronous. Instead we always compute (7) using `logsumexp()`.

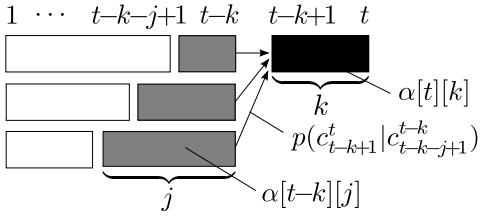


Figure 4: Forward filtering of $\alpha[t][k]$ to marginalize out possible segmentations j before $t-k$.

for $t = 1$ to N **do**
  for $k = \max(1, t-L)$ to t **do**
    Compute $\alpha[t][k]$ according to (7).
  end for
end for
Initialize $t \leftarrow N$, $i \leftarrow 0$, $w_0 \leftarrow \$$
while $t > 0$ **do**
  Draw $k \propto p(w_i | c_{t-k+1}^t, \Theta) \cdot \alpha[t][k]$
  Set $w_i \leftarrow c_{t-k+1}^t$
  Set $t \leftarrow t-k$, $i \leftarrow i+1$
end while
Return $\mathbf{w} = w_i, w_{i-1}, \cdots, w_1$.

Figure 5: Forward-Backward sampling of word segmentation $\mathbf{w}$. (in bigram case)

word boundary for each t as q_t, we can write

$$\alpha[t][k] = p(c_1^t, q_t = k) \tag{8}$$
$$= \sum_j p(c_1^t, q_t = k, q_{t-k} = j) \tag{9}$$
$$= \sum_j p(c_1^{t-k}, c_{t-k+1}^t, q_t = k, q_{t-k} = j) \tag{10}$$
$$= \sum_j p(c_{t-k+1}^t | c_1^{t-k}) p(c_1^{t-k}, q_{t-k} = j) \tag{11}$$
$$= \sum_j p(c_{t-k+1}^t | c_1^{t-k}) \alpha[t-k][j]\,, \tag{12}$$

where we used conditional independency of q_t given q_{t-k} and uniform prior over q_t in (11) above.

Backward Sampling. Once the probability table $\alpha[t][k]$ is obtained, we can sample a word segmentation backwards. Since $\alpha[N][k]$ is a marginal probability of string c_1^N with the last k characters being a word, and there is always a sentence boundary token \$ at the end of the string, with probability proportional to $p(\$ | c_{N-k}^N) \cdot \alpha[N][k]$ we can sample k to choose the boundary of the final word. The second final word is similarly sampled using the probability of preceding the last word just sampled: we continue this process until we arrive at the beginning of the string (Figure 5).

Trigram case. For simplicity, we showed the algorithm for bigrams above. For trigrams, we maintain a forward variable $\alpha[t][k][j]$, which represents a marginal probability of string $c_1 \cdots c_t$ with both the final k characters and further j characters preceding it being words. Forward-Backward algorithm becomes complicated thus omitted, but can be derived following the extended algorithm for second order HMM (He, 1988).

Complexity This algorithm has a complexity of $O(NL^2)$ for bigrams and $O(NL^3)$ for trigrams for each sentence, where N is the length of the sentence and L is the maximum allowed length of a word ($\leq N$).

4.3 Poisson correction

As Nagata (1996) noted, when only (5) is used inadequately low probabilities are assigned to long words, because it has a largely exponential distribution over length. To correct this, we assume that word length k has a Poisson distribution with a mean λ:

$$\mathrm{Po}(k|\lambda) = e^{-\lambda} \frac{\lambda^k}{k!}\,. \tag{13}$$

Since the appearance of $c_1 \cdots c_k$ is equivalent to that of length k and the content, by making the character n-gram model explicit as Θ we can set

$$p(c_1 \cdots c_k) = p(c_1 \cdots c_k, k) \tag{14}$$
$$= \frac{p(c_1 \cdots c_k, k|\Theta)}{p(k|\Theta)} \mathrm{Po}(k|\lambda) \tag{15}$$

where $p(c_1 \cdots c_k, k|\Theta)$ is an n-gram probability given by (6), and $p(k|\Theta)$ is a probability that a word of length k will be generated from Θ. While previous work used $p(k|\Theta) = (1 - p(\$))^{k-1} p(\$)$, this is only true for unigrams. Instead, we employed a Monte Carlo method that generates words randomly from Θ to obtain the empirical estimates of $p(k|\Theta)$.

Estimating λ. Of course, we do not leave λ as a constant. Instead, we put a Gamma distribution

$$p(\lambda) = \mathrm{Ga}(a, b) = \frac{b^a}{\Gamma(a)} \lambda^{a-1} e^{-b\lambda} \tag{16}$$

to estimate λ from the data for given language and word type.[7] Here, $\Gamma(x)$ is a Gamma function and a, b are the hyperparameters chosen to give a nearly uniform prior distribution.[8]

[7] We used different λ for different word types, such as digits, alphabets, hiragana, CJK characters, and their mixtures. W is a set of words of each such type, and (13) becomes a mixture of Poisson distributions in this case.

[8] In the following experiments, we set $a = 0.2, b = 0.1$.

Denoting W as a set of "words" obtained from word segmentation, the posterior distribution of λ used for (13) is

$$p(\lambda|W) \propto p(W|\lambda)p(\lambda)$$
$$= \mathrm{Ga}\left(a+\sum_{w\in W} t(w)|w|,\ b+\sum_{w\in W} t(w)\right), \quad (17)$$

where $t(w)$ is the number of times word w is generated from the character HPYLM, i.e. the number of tables $t_{\epsilon w}$ for w in word unigrams. We sampled λ from this posterior for each Gibbs iteration.

5 Experiments

To validate our model, we conducted experiments on standard datasets for Chinese and Japanese word segmentation that are publicly available, as well as the same dataset used in (Goldwater et al., 2006). Note that NPYLM maximizes the probability of strings, equivalently, minimizes the perplexity per character. Therefore, the recovery of the "ground truth" that is not available for inference is a byproduct in unsupervised learning.

Since our implementation is based on Unicode and learns all hyperparameters from the data, we also confirmed that NPYLM segments the Arabic Gigawords equally well.

5.1 English phonetic transcripts

In order to directly compare with the previously reported result, we first used the same dataset as Goldwater et al. (2006). This dataset consists of 9,790 English phonetic transcripts from CHILDES data (MacWhinney and Snow, 1985).

Since our algorithm converges rather fast, we ran the Gibbs sampler of trigram NPYLM for 200 iterations to obtain the results in Table 1. Among the token precision (P), recall (R), and F-measure (F), the recall is especially higher to outperform the previous result based on HDP in F-measure. Meanwhile, the same measures over the obtained lexicon (LP, LR, LF) are not always improved. Moreover, the average length of words inferred was surprisingly similar to ground truth: 2.88, while the ground truth is 2.87.

Table 2 shows the empirical computational time needed to obtain these results. Although the convergence in MCMC is not uniquely identified, improvement in efficiency is also outstanding.

5.2 Chinese and Japanese word segmentation

To show applicability beyond small phonetic transcripts, we used standard datasets for Chinese and Japanese word segmentation, with all supervised segmentations removed in advance.

Model	P	R	F	LP	LR	LF
NPY(3)	74.8	75.2	75.0	47.8	**59.7**	53.1
NPY(2)	74.8	**76.7**	**75.7**	57.3	56.6	57.0
HDP(2)	**75.2**	69.6	72.3	**63.5**	55.2	**59.1**

Table 1: Segmentation accuracies on English phonetic transcripts. NPY(n) means n-gram NPYLM. Results for HDP(2) are taken from Goldwater et al. (2009), which corrects the errors in Goldwater et al. (2006).

Model	time	iterations
NPYLM	17min	200
HDP	10h 55min	20000

Table 2: Computations needed for Table 1. Iterations for "HDP" is the same as described in Goldwater et al. (2009). Actually, NPYLM approximately converged around 50 iterations, 4 minutes.

Chinese For Chinese, we used a publicly available SIGHAN Bakeoff 2005 dataset (Emerson, 2005). To compare with the latest unsupervised results (using a closed dataset of Bakeoff 2006), we chose the common sets prepared by Microsoft Research Asia (MSR) for simplified Chinese, and by City University of Hong Kong (CITYU) for traditional Chinese. We used a random subset of 50,000 sentences from each dataset for training, and the evaluation was conducted on the enclosed test data. [9]

Japanese For Japanese, we used the Kyoto Corpus (Kyoto) (Kurohashi and Nagao, 1998): we used random subset of 1,000 sentences for evaluation and the remaining 37,400 sentences for training. In all cases we removed all whitespaces to yield raw character strings for inference, and set $L = 4$ for Chinese and $L = 8$ for Japanese to run the Gibbs sampler for 400 iterations.

The results (in token F-measures) are shown in Table 3. Our NPYLM significantly ourperforms the best results using a heuristic approach reported in Zhao and Kit (2008). While Japanese accuracies appear lower, subjective qualities are much higher. This is mostly because NPYLM segments inflectional suffixes and combines frequent proper names, which are inconsistent with the "correct"

[9] Notice that analyzing a test data is not easy for character-wise Gibbs sampler of previous work. Meanwhile, NPYLM easily finds the best segmentation using the Viterbi algorithm once the model is learned.

Model	MSR	CITYU	Kyoto
NPY(2)	80.2 (51.9)	**82.4** (**126.5**)	62.1 (23.1)
NPY(3)	**80.7** (**48.8**)	81.7 (128.3)	**66.6** (**20.6**)
ZK08	66.7 (—)	69.2 (—)	—

Table 3: Accuracies and perplexities per character (in parentheses) on actual corpora. "ZK08" are the best results reported in Zhao and Kit (2008). We used ∞-gram for characters.

	MSR	CITYU	Kyoto
Semi	0.895 (**48.8**)	0.898 (**124.7**)	0.913 (**20.3**)
Sup	**0.945** (81.4)	**0.941** (194.8)	**0.971** (21.3)

Table 4: Semi-supervised and supervised results. Semi-supervised results used only 10K sentences (1/5) of supervised segmentations.

segmentations. Bigram and trigram performances are similar for Chinese, but trigram performs better for Japanese. In fact, although the difference in perplexity per character is not so large, the perplexity per word is radically reduced: 439.8 (bigram) to 190.1 (trigram). This is because trigram models can leverage complex dependencies over words to yield shorter words, resulting in better predictions and increased tokens.

Furthermore, NPYLM is easily amenable to semi-supervised or even supervised learning. In that case, we have only to replace the word segmentation $\mathbf{w}(s)$ in Figure 3 to the supervised one, for all or part of the training data. Table 4 shows the results using 10,000 sentences (1/5) or complete supervision. Our completely generative model achieves the performance of 94% (Chinese) or even 97% (Japanese) in supervised case. The result also shows that the supervised segmentations are suboptimal with respect to the perplexity per character, and even worse than unsupervised results. In semi-supervised case, using only 10K reference segmentations gives a performance of around 90% accuracy and the lowest perplexity, thanks to a combination with unsupervised data in a principled fashion.

5.3 Classics and English text

Our model is particularly effective for spoken transcripts, colloquial texts, classics, or unknown languages where supervised segmentation data is difficult or even impossible to create. For example, we are pleased to say that we can now analyze (and build a language model on) "*The Tale of Genji*", the core of Japanese classics written 1,000 years ago (Figure 6). The inferred segmentations are

いづれ の 御 時 に か 、 女御 更衣 あまた さぶらひ たまひ ける 中に 、 いと やむごとなき 際 に は あら ぬ が 、 すぐれて 時めき たまふ あり けり 。 はじめより 我 は と 思ひ あがり たまへ る 御 方々 、 めざまし き もの に おとしめ そねみ たまふ 。 同じ ほど 、 それ より 下臈 の 更衣 たち は 、 まして やすから ず 。 朝夕 の 宮仕 につけても 、 ...

Figure 6: Unsupervised segmentation result for "*The Tale of Genji*". (16,443 sentences, 899,668 characters in total)

mostly correct, with some inflectional suffixes being recognized as words, which is also the case with English.

Finally, we note that our model is also effective for western languages: Figure 7 shows a training text of "*Alice in Wonderland*" with all whitespaces removed, and the segmentation result.

While the data is extremely small (only 1,431 lines, 115,961 characters), our trigram NPYLM can infer the words surprisingly well. This is because our model contains both word and character models that are combined and carefully smoothed, from a Bayesian perspective.

6 Discussion

In retrospect, our NPYLM is essentially a hierarchical Markov model where the units (=words) evolve as the Markov process, and each unit has subunits (=characters) that also evolve as the Markov process. Therefore, for such languages as English that have already space-separated tokens, we can also begin with tokens besides the character-based approach in Section 5.3. In this case, each token is a "character" whose code is the integer token type, and a sentence is a sequence of "characters." Figure 8 shows a part of the result computed over 100K sentences from Penn Treebank. We can see that some frequent phrases are identified as "words", using a fully unsupervised approach. Notice that this is only attainable with NPYLM where each phrase is described as a n-gram model on its own, here a word ∞-gram language model.

While we developed an efficient forward-backward algorithm for unsupervised segmentation, it is reminiscent of CRF in the discriminative approach. Therefore, it is also interesting to combine them in a discriminative way as persued in POS tagging using CRF+HMM (Suzuki et al., 2007), let alone a simple semi-supervised approach in Section 5.2. This paper provides a foundation of such possibilities.

lastly,shepicturedtoherselfhowthissamelittlesisterofhersw
ould,intheafter-time,beherselfagrownwoman;andhowshe
wouldkeep,throughallherriperyears,thesimpleandlovingh
eartofherchildhood:andhowshewouldgatheraboutherothe
rlittlechildren,andmaketheireyesbrightandeagerwithmany
astrangetale,perhapsevenwiththedreamofwonderlandoflo
ngago:andhowshewouldfeelwithalltheirsimplesorrows,an
dfindapleasureinalltheirsimplejoys,rememberingherownc
hild-life,andthehappysummerdays.

(a) Training data (in part).

last ly , she pictured to herself how this same little sis-
ter of her s would , inthe after - time , be herself agrown
woman ; and how she would keep , through allher ripery
ears , the simple and loving heart of her child hood : and
how she would gather about her other little children ,and
make theireyes bright and eager with many a strange tale
, perhaps even with the dream of wonderland of longago
: and how she would feel with all their simple sorrow s ,
and find a pleasure in all their simple joys , remember ing
her own child - life , and thehappy summerday s .

(b) Segmentation result. Note we used no dictionary.

Figure 7: Word segmentation of "*Alice in Wonderland*".

7 Conclusion

In this paper, we proposed a much more efficient and accurate model for fully unsupervised word segmentation. With a combination of dynamic programming and an accurate spelling model from a Bayesian perspective, our model significantly outperforms the previous reported results, and the inference is very efficient.

This model is also considered as a way to build a Bayesian Kneser-Ney smoothed word n-gram language model directly from characters with no "word" indications. In fact, it achieves lower perplexity per character than that based on supervised segmentations. We believe this will be particularly beneficial to build a language model on such texts as speech transcripts, colloquial texts or unknown languages, where word boundaries are hard or even impossible to identify a priori.

Acknowledgments

We thank Vikash Mansinghka (MIT) for a motivating discussion leading to this research, and Satoru Takabayashi (Google) for valuable technical advice.

References

David Aldous, 1985. *Exchangeability and Related Topics*, pages 1–198. Springer Lecture Notes in Math. 1117.

Rie Kubota Ando and Lillian Lee. 2003. Mostly-Unsupervised Statistical Segmentation of Japanese Kanji Sequences. *Natural Language Engineering*, 9(2):127–149.

Peter F. Brown, Vincent J. Della Pietra, Robert L. Mercer, Stephen A. Della Pietra, and Jennifer C. Lai. 1992. An Estimate of an Upper Bound for the Entropy of English. *Computational Linguistics*, 18:31–40.

Arnaud Doucet, Christophe Andrieu, and Roman Holenstein. 2009. Particle Markov Chain Monte Carlo. *in submission*.

Tom Emerson. 2005. SIGHAN Bakeoff 2005. http://www.sighan.org/bakeoff2005/.

W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. 1996. *Markov Chain Monte Carlo in Practice*. Chapman & Hall / CRC.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2005. Interpolating Between Types and Tokens by Estimating Power-Law Generators. In *NIPS 2005*.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual Dependencies in Unsupervised Word Segmentation. In *Proceedings of ACL/COLING 2006*, pages 673–680.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, *in press*.

Yang He. 1988. Extended Viterbi algorithm for second order hidden Markov process. In *Proceedings of ICPR 1988*, pages 718–720.

nevertheless ,
he was admired
by many of his immediate subordinates
for his long work hours
and dedication to building northwest
into what he called a " mega carrier
. "

---

although
preliminary findings
were reported
more than a year ago ,
the latest results
appear
in today 's
new england journal of medicine ,
a forum
likely to bring new attention to the problem
.

---

south korea
registered a trade deficit of $ 101 million
in october
, reflecting the country 's economic sluggishness
, according to government figures released wednesday
.

Figure 8: Generative phrase segmentation of Penn Treebank text computed by NPYLM. Each line is a "word" consisting of actual words.

Mark Johnson and Sharon Goldwater. 2009. Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *NAACL 2009*.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Bayesian Inference for PCFGs via Markov Chain Monte Carlo. In *Proceedings of HLT/NAACL 2007*, pages 139–146.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of ICASSP*, volume 1, pages 181–184.

Sadao Kurohashi and Makoto Nagao. 1998. Building a Japanese Parsed Corpus while Improving the Parsing System. In *Proceedings of LREC 1998*, pages 719–724. http://nlp.kuee.kyoto-u.ac.jp/nl-resource/corpus.html.

Brian MacWhinney and Catherine Snow. 1985. The Child Language Data Exchange System. *Journal of Child Language*, 12:271–296.

Daichi Mochihashi and Eiichiro Sumita. 2007. The Infinite Markov Model. In *NIPS 2007*.

Kevin Murphy. 2002. Hidden semi-Markov models (segment models). http://www.cs.ubc.ca/˜murphyk/Papers/segment.pdf.

Masaaki Nagata. 1996. Automatic Extraction of New Words from Japanese Texts using Generalized Forward-Backward Search. In *Proceedings of EMNLP 1996*, pages 48–59.

Abel Rodriguez, David Dunson, and Alan Gelfand. 2008. The Nested Dirichlet Process. *Journal of the American Statistical Association*, 103:1131–1154.

Steven L. Scott. 2002. Bayesian Methods for Hidden Markov Models. *Journal of the American Statistical Association*, 97:337–351.

Jun Suzuki, Akinori Fujino, and Hideki Isozaki. 2007. Semi-Supervised Structured Output Learning Based on a Hybrid Generative and Discriminative Approach. In *Proceedings of EMNLP-CoNLL 2007*, pages 791–800.

Yee Whye Teh. 2006a. A Bayesian Interpretation of Interpolated Kneser-Ney. Technical Report TRA2/06, School of Computing, NUS.

Yee Whye Teh. 2006b. A Hierarchical Bayesian Language Model based on Pitman-Yor Processes. In *Proceedings of ACL/COLING 2006*, pages 985–992.

Frank Wood and Yee Whye Teh. 2008. A Hierarchical, Hierarchical Pitman-Yor Process Language Model. In *ICML 2008 Workshop on Nonparametric Bayes*.

Jia Xu, Jianfeng Gao, Kristina Toutanova, and Hermann Ney. 2008. Bayesian Semi-Supervised Chinese Word Segmentation for Statistical Machine Translation. In *Proceedings of COLING 2008*, pages 1017–1024.

Hai Zhao and Chunyu Kit. 2008. An Empirical Comparison of Goodness Measures for Unsupervised Chinese Word Segmentation with a Unified Framework. In *Proceedings of IJCNLP 2008*.

Knowing the Unseen: Estimating Vocabulary Size over Unseen Samples

Suma Bhat
Department of ECE
University of Illinois
spbhat2@illinois.edu

Richard Sproat
Center for Spoken Language Understanding
Oregon Health & Science University
rws@xoba.com

Abstract

Empirical studies on corpora involve making measurements of several quantities for the purpose of comparing corpora, creating language models or to make generalizations about specific linguistic phenomena in a language. Quantities such as average word length are stable across sample sizes and hence can be reliably estimated from large enough samples. However, quantities such as *vocabulary size* change with sample size. Thus measurements based on a given sample will need to be *extrapolated* to obtain their estimates over larger unseen samples. In this work, we propose a novel *nonparametric* estimator of vocabulary size. Our main result is to show the *statistical consistency* of the estimator – the first of its kind in the literature. Finally, we compare our proposal with the state of the art estimators (both parametric and nonparametric) on large standard corpora; apart from showing the favorable performance of our estimator, we also see that the classical Good-Turing estimator consistently underestimates the vocabulary size.

1 Introduction

Empirical studies on corpora involve making measurements of several quantities for the purpose of comparing corpora, creating language models or to make generalizations about specific linguistic phenomena in a language. Quantities such as average word length or average sentence length are stable across sample sizes. Hence empirical measurements from large enough samples tend to be reliable for even larger sample sizes. On the other hand, quantities associated with word frequencies, such as the number of *hapax legomena* or the number of distinct word types changes are strictly sample size dependent. Given a sample we can obtain the seen vocabulary and the seen number of *hapax legomena*. However, for the purpose of comparison of corpora of different sizes or linguistic phenomena based on samples of different sizes it is imperative that these quantities be compared based on similar sample sizes. We thus need methods to extrapolate empirical measurements of these quantities to arbitrary sample sizes.

Our focus in this study will be estimators of vocabulary size for samples larger than the sample available. There is an abundance of estimators of population size (in our case, vocabulary size) in existing literature. Excellent survey articles that summarize the state-of-the-art are available in (Bunge and Fitzpatrick, 1993) and (Gandolfi and Sastri, 2004). Of particular interest to us is the set of estimators that have been shown to model word frequency distributions well. This study proposes a nonparametric estimator of vocabulary size and evaluates its theoretical and empirical performance. For comparison we consider some state-of-the-art parametric and nonparametric estimators of vocabulary size.

The proposed non-parametric estimator for the number of unseen elements assumes a regime characterizing word frequency distributions. This work is motivated by a scaling formulation to address the problem of unlikely events proposed in (Baayen, 2001; Khmaladze, 1987; Khmaladze and Chitashvili, 1989; Wagner et al., 2006). We also demonstrate that the estimator is strongly consistent under the natural scaling formulation. While compared with other vocabulary size estimates, we see that our estimator performs at least as well as some of the state of the art estimators.

2 Previous Work

Many estimators of vocabulary size are available in the literature and a comparison of several non

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 109–117,
Suntec, Singapore, 2-7 August 2009.

parametric estimators of population size occurs in (Gandolfi and Sastri, 2004). While a definite comparison including parametric estimators is lacking, there is also no known work comparing methods of extrapolation of vocabulary size. Baroni and Evert, in (Baroni and Evert, 2005), evaluate the performance of some estimators in extrapolating vocabulary size for arbitrary sample sizes but limit the study to parametric estimators. Since we consider both parametric and nonparametric estimators here, we consider this to be the first study comparing a set of estimators for extrapolating vocabulary size.

Estimators of vocabulary size that we compare can be broadly classified into two types:

1. *Nonparametric estimators*- here word frequency information from the given sample alone is used to estimate the vocabulary size. A good survey of the state of the art is available in (Gandolfi and Sastri, 2004). In this paper, we compare our proposed estimator with the canonical estimators available in (Gandolfi and Sastri, 2004).

2. *Parametric estimators*- here a probabilistic model capturing the relation between expected vocabulary size and sample size is the estimator. Given a sample of size n, the sample serves to calculate the parameters of the model. The expected vocabulary for a given sample size is then determined using the explicit relation. The parametric estimators considered in this study are (Baayen, 2001; Baroni and Evert, 2005),

   (a) Zipf-Mandelbrot estimator (ZM);
   (b) finite Zipf-Mandelbrot estimator (fZM).

In addition to the above estimators we consider a novel non parametric estimator. It is the nonparametric estimator that we propose, taking into account the characteristic feature of word frequency distributions, to which we will turn next.

3 Novel Estimator of Vocabulary size

We observe $(X_1, \ldots, X_n)$, an i.i.d. sequence drawn according to a probability distribution $\mathbb{P}$ from a large, but finite, vocabulary Ω. Our goal is in estimating the "essential" size of the vocabulary Ω using only the observations. In other words, having seen a sample of size n we wish to know, given another sample from the same population, how many unseen elements we would expect to see. Our nonparametric estimator for the number of unseen elements is motivated by the characteristic property of word frequency distributions, the *Large Number of Rare Events* (LNRE) (Baayen, 2001). We also demonstrate that the estimator is strongly consistent under a natural scaling formulation described in (Khmaladze, 1987).

3.1 A Scaling Formulation

Our main interest is in probability distributions $\mathbb{P}$ with the property that a large number of words in the vocabulary Ω are unlikely, i.e., the chance any word appears eventually in an arbitrarily long observation is strictly between 0 and 1. The authors in (Baayen, 2001; Khmaladze and Chitashvili, 1989; Wagner et al., 2006) propose a natural scaling formulation to study this problem; specifically, (Baayen, 2001) has a tutorial-like summary of the theoretical work in (Khmaladze, 1987; Khmaladze and Chitashvili, 1989). In particular, the authors consider a *sequence* of vocabulary sets and probability distributions, indexed by the observation size n. Specifically, the observation $(X_1, \ldots, X_n)$ is drawn i.i.d. from a vocabulary Ω_n according to probability $\mathbb{P}_n$. If the probability of a word, say $\omega \in \Omega_n$ is p, then the probability that this specific word ω does not occur in an observation of size n is

$$(1-p)^n .$$

For ω to be an unlikely word, we would like this probability for large n to remain strictly between 0 and 1. This implies that

$$\frac{\check{c}}{n} \leq p \leq \frac{\hat{c}}{n}, \tag{1}$$

for some strictly positive constants $0 < \check{c} < \hat{c} < \infty$. We will assume throughout this paper that $\check{c}$ and $\hat{c}$ are the same for every word $\omega \in \Omega_n$. This implies that the vocabulary size is growing *linearly* with the observation size:

$$\frac{n}{\hat{c}} \leq |\Omega_n| \leq \frac{n}{\check{c}}.$$

This model is called the *LNRE zone* and its applicability in natural language corpora is studied in detail in (Baayen, 2001).

3.2 Shadows

Consider the observation string $(X_1, \ldots, X_n)$ and let us denote the quantity of interest – the number

of word types in the vocabulary Ω_n that are not observed – by $\mathbb{O}_n$. This quantity is random since the observation string itself is. However, we note that the distribution of $\mathbb{O}_n$ is unaffected if one relabels the words in Ω_n. This motivates studying of the probabilities assigned by $\mathbb{P}_n$ without reference to the labeling of the word; this is done in (Khmaladze and Chitashvili, 1989) via the *structural distribution function* and in (Wagner et al., 2006) via the *shadow*. Here we focus on the latter description:

Definition 1 *Let X_n be a random variable on Ω_n with distribution $\mathbb{P}_n$. The* shadow *of $\mathbb{P}_n$ is defined to be the distribution of the random variable $\mathbb{P}_n(\{X_n\})$.*

For the finite vocabulary situation we are considering, specifying the shadow is *exactly equivalent* to specifying the unordered components of $\mathbb{P}_n$, viewed as a probability vector.

3.3 Scaled Shadows Converge

We will follow (Wagner et al., 2006) and suppose that the scaled shadows, the distribution of $n \cdot \mathbb{P}_n(X_n)$, denoted by Q_n converge to a distribution Q. As an example, if $\mathbb{P}_n$ is a uniform distribution over a vocabulary of size cn, then $n \cdot \mathbb{P}_n(X_n)$ equals $\frac{1}{c}$ almost surely for each n (and hence it converges in distribution). From this convergence assumption we can, further, infer the following:

1. Since the probability of each word ω is lower and upper bounded as in Equation (1), we know that the distribution Q_n is non-zero only in the range $[\check{c}, \hat{c}]$.

2. The "essential" size of the vocabulary, i.e., the number of words of Ω_n on which $\mathbb{P}_n$ puts non-zero probability can be evaluated directly from the scaled shadow, scaled by $\frac{1}{n}$ as

$$\int_{\check{c}}^{\hat{c}} \frac{1}{y}\, dQ_n(y). \tag{2}$$

   Using the dominated convergence theorem, we can conclude that the convergence of the scaled shadows guarantees that the size of the vocabulary, scaled by $1/n$, converges as well:

$$\frac{|\Omega_n|}{n} \to \int_{\check{c}}^{\hat{c}} \frac{1}{y}\, dQ(y). \tag{3}$$

3.4 Profiles and their Limits

Our goal in this paper is to estimate the size of the underlying vocabulary, i.e., the expression in (2),

$$\int_{\check{c}}^{\hat{c}} \frac{n}{y}\, dQ_n(y), \tag{4}$$

from the observations $(X_1, \ldots, X_n)$. We observe that since the scaled shadow Q_n does not depend on the labeling of the words in Ω_n, a *sufficient statistic* to estimate (4) from the observation $(X_1, \ldots, X_n)$ is the *profile* of the observation: $(\varphi_1^n, \ldots, \varphi_n^n)$, defined as follows. φ_k^n is the number of word types that appear exactly k times in the observation, for $k = 1, \ldots, n$. Observe that

$$\sum_{k=1}^{n} k\varphi_k^n = n,$$

and that

$$V \stackrel{\text{def}}{=} \sum_{k=1}^{n} \varphi_k^n \tag{5}$$

is the number of *observed* words. Thus, the object of our interest is,

$$\mathbb{O}_n = |\Omega_n| - V. \tag{6}$$

3.5 Convergence of Scaled Profiles

One of the main results of (Wagner et al., 2006) is that the scaled profiles converge to a deterministic probability vector under the scaling model introduced in Section 3.3. Specifically, we have from Proposition 1 of (Wagner et al., 2006):

$$\sum_{k=1}^{n} \left| \frac{k\varphi_k}{n} - \lambda_{k-1} \right| \longrightarrow 0, \quad \text{almost surely,} \tag{7}$$

where

$$\lambda_k := \int_{\check{c}}^{\check{c}} \frac{y^k \exp(-y)}{k!}\, dQ(y) \quad k = 0, 1, 2, \ldots. \tag{8}$$

This convergence result suggests a natural estimator for $\mathbb{O}_n$, expressed in Equation (6).

3.6 A Consistent Estimator of $\mathbb{O}_n$

We start with the limiting expression for scaled profiles in Equation (7) and come up with a natural estimator for $\mathbb{O}_n$. Our development leading to the estimator is somewhat heuristic and is aimed at motivating the structure of the estimator for the number of unseen words, $\mathbb{O}_n$. We formally state and prove its consistency at the end of this section.

3.6.1 A Heuristic Derivation

Starting from (7), let us first make the approximation that

$$\frac{k\varphi_k}{n} \approx \lambda_{k-1}, \quad k = 1, \ldots, n. \tag{9}$$

We now have the formal calculation

$$\begin{aligned}
\sum_{k=1}^{n} \frac{\varphi_k^n}{n} &\approx \sum_{k=1}^{n} \frac{\lambda_{k-1}}{k} && (10)\\
&= \sum_{k=1}^{n} \int_{\check{c}}^{\hat{c}} \frac{e^{-y} y^{k-1}}{k!}\, dQ(y)\\
&\approx \int_{\check{c}}^{\hat{c}} \frac{e^{-y}}{y} \left(\sum_{k=1}^{n} \frac{y^k}{k!} \right) dQ(y) && (11)\\
&\approx \int_{\check{c}}^{\hat{c}} \frac{e^{-y}}{y} \left(e^y - 1\right)\, dQ(y) && (12)\\
&\approx \frac{|\Omega_n|}{n} - \int_{\check{c}}^{\hat{c}} \frac{e^{-y}}{y}\, dQ(y). && (13)
\end{aligned}$$

Here the approximation in Equation (10) follows from the approximation in Equation (9), the approximation in Equation (11) involves swapping the outer discrete summation with integration and is justified formally later in the section, the approximation in Equation (12) follows because

$$\sum_{k=1}^{n} \frac{y^k}{k!} \to e^y - 1,$$

as $n \to \infty$, and the approximation in Equation (13) is justified from the convergence in Equation (3). Now, comparing Equation (13) with Equation (6), we arrive at an approximation for our quantity of interest:

$$\frac{\mathbb{O}_n}{n} \approx \int_{\check{c}}^{\hat{c}} \frac{e^{-y}}{y}\, dQ(y). \tag{14}$$

The geometric series allows us to write

$$\frac{1}{y} = \frac{1}{\hat{c}} \sum_{\ell=0}^{\infty} \left(1 - \frac{y}{\hat{c}}\right)^{\ell}, \quad \forall y \in (0, \hat{c})\,. \tag{15}$$

Approximating this infinite series by a finite summation, we have for all $y \in (\check{c}, \hat{c})$,

$$\begin{aligned}
\frac{1}{y} - \frac{1}{\hat{c}} \sum_{\ell=0}^{M} \left(1 - \frac{y}{\hat{c}}\right)^{\ell} &= \frac{\left(1 - \frac{y}{\hat{c}}\right)^M}{y}\\
&\leq \frac{\left(1 - \frac{\check{c}}{\hat{c}}\right)^M}{\check{c}}. && (16)
\end{aligned}$$

It helps to write the truncated geometric series as a power series in y:

$$\begin{aligned}
&\frac{1}{\hat{c}} \sum_{\ell=0}^{M} \left(1 - \frac{y}{\hat{c}}\right)^{\ell}\\
&= \frac{1}{\hat{c}} \sum_{\ell=0}^{M} \sum_{k=0}^{\ell} \binom{\ell}{k} (-1)^k \left(\frac{y}{\hat{c}}\right)^k\\
&= \frac{1}{\hat{c}} \sum_{k=0}^{M} \left(\sum_{\ell=k}^{M} \binom{\ell}{k} \right) (-1)^k \left(\frac{y}{\hat{c}}\right)^k\\
&= \sum_{k=0}^{M} (-1)^k\, a_k^M\, y^k, && (17)
\end{aligned}$$

where we have written

$$a_k^M := \frac{1}{\hat{c}^{k+1}} \left(\sum_{\ell=k}^{M} \binom{\ell}{k} \right).$$

Substituting the finite summation approximation in Equation 16 and its power series expression in Equation (17) into Equation (14) and swapping the discrete summation with the integral, we can continue

$$\begin{aligned}
\frac{\mathbb{O}_n}{n} &\approx \sum_{k=0}^{M} (-1)^k\, a_k^M \int_{\check{c}}^{\hat{c}} e^{-y} y^k\, dQ(y)\\
&= \sum_{k=0}^{M} (-1)^k\, a_k^M\, k! \lambda_k. && (18)
\end{aligned}$$

Here, in Equation (18), we used the definition of λ_k from Equation (8). From the convergence in Equation (7), we finally arrive at our estimate:

$$\mathbb{O}_n \approx \sum_{k=0}^{M} (-1)^k\, a_k^M\, (k+1)!\ \varphi_{k+1}. \tag{19}$$

3.6.2 Consistency

Our main result is the demonstration of the consistency of the estimator in Equation (19).

Theorem 1 *For any $\epsilon > 0$,*

$$\lim_{n \to \infty} \frac{\left| \mathbb{O}_n - \sum_{k=0}^{M} (-1)^k\, a_k^M\, (k+1)!\, \varphi_{k+1} \right|}{n} \leq \epsilon$$

almost surely, as long as

$$M \geq \frac{\check{c} \log_2 e + \log_2 (\epsilon \check{c})}{\log_2 (\hat{c} - \check{c}) - 1 - \log_2 (\hat{c})}. \tag{20}$$

Proof: From Equation (6), we have

$$\begin{aligned}\frac{\mathbb{O}_n}{n} &= \frac{|\Omega_n|}{n} - \sum_{k=1}^{n} \frac{\varphi_k}{n} \\ &= \frac{|\Omega_n|}{n} - \sum_{k=1}^{n} \frac{\lambda_{k-1}}{k} - \\ &\quad \sum_{k=1}^{n} \frac{1}{k} \left(\frac{k\varphi_k}{n} - \lambda_{k-1} \right). \quad (21)\end{aligned}$$

The first term in the right hand side (RHS) of Equation (21) converges as seen in Equation (3). The third term in the RHS of Equation (21) converges to zero, almost surely, as seen from Equation (7). The second term in the RHS of Equation (21), on the other hand,

$$\begin{aligned}\sum_{k=1}^{n} \frac{\lambda_{k-1}}{k} &= \int_{\check{c}}^{\hat{c}} \frac{e^{-y}}{y} \left(\sum_{k=1}^{n} \frac{y^k}{k!} \right) \, dQ(y) \\ &\rightarrow \int_{\check{c}}^{\hat{c}} \frac{e^{-y}}{y} \left(e^y - 1 \right) \, dQ(y), n \rightarrow \infty, \\ &= \int_{\check{c}}^{\hat{c}} \frac{1}{y} \, dQ(y) - \int_{\check{c}}^{\hat{c}} \frac{e^{-y}}{y} \, dQ(y).\end{aligned}$$

The monotone convergence theorem justifies the convergence in the second step above. Thus we conclude that

$$\lim_{n \rightarrow \infty} \frac{\mathbb{O}_n}{n} = \int_{\check{c}}^{\hat{c}} \frac{e^{-y}}{y} \, dQ(y) \quad (22)$$

almost surely. Coming to the estimator, we can write it as the sum of two terms:

$$\begin{aligned}&\sum_{k=0}^{M} (-1)^k \, a_k^M \, k! \lambda_k \quad (23) \\ &\quad + \sum_{k=0}^{M} (-1)^k \, a_k^M \, k! \left(\frac{(k+1)\, \varphi_{k+1}}{n} - \lambda_k \right).\end{aligned}$$

The second term in Equation (23) above is seen to converge to zero almost surely as $n \rightarrow \infty$, using Equation (7) and noting that M is a constant not depending on n. The first term in Equation (23) can be written as, using the definition of λ_k from Equation (8),

$$\int_{\check{c}}^{\hat{c}} e^{-y} \left(\sum_{k=0}^{M} (-1)^k \, a_k^M y^k \right) \, dQ(y). \quad (24)$$

Combining Equations (22) and (24), we have that, almost surely,

$$\begin{aligned}&\lim_{n \rightarrow \infty} \frac{\mathbb{O}_n - \sum_{k=0}^{M} (-1)^k \, a_k^M \, (k+1)! \, \varphi_{k+1}}{n} = \\ &\int_{\check{c}}^{\hat{c}} e^{-y} \left(\frac{1}{y} - \sum_{k=0}^{M} (-1)^k \, a_k^M y^k \right) \, dQ(y). \quad (25)\end{aligned}$$

Combining Equation (16) with Equation (17), we have

$$0 < \frac{1}{y} - \sum_{k=0}^{M} (-1)^k \, a_k^M y^k \leq \frac{\left(1 - \frac{\check{c}}{\hat{c}}\right)^M}{\check{c}}. \quad (26)$$

The quantity in Equation (25) can now be upper bounded by, using Equation (26),

$$\frac{e^{-\check{c}} \left(1 - \frac{\check{c}}{\hat{c}}\right)^M}{\check{c}}.$$

For M that satisfy Equation (20) this term is less than ϵ. The proof concludes.

3.7 Uniform Consistent Estimation

One of the main issues with actually employing the estimator for the number of unseen elements (cf. Equation (19)) is that it involves knowing the parameter $\hat{c}$. In practice, there is no natural way to obtain any estimate on this parameter $\hat{c}$. It would be most useful if there were a way to modify the estimator in a way that it does not depend on the unobservable quantity $\hat{c}$. In this section we see that such a modification is possible, while still retaining the main theoretical performance result of consistency (cf. Theorem 1).

The first step to see the modification is in observing where the need for $\hat{c}$ arises: it is in writing the geometric series for the function $\frac{1}{y}$ (cf. Equations (15) and (16)). If we could let $\hat{c}$ along with the number of elements M itself depend on the sample size n, then we could still have the geometric series formula. More precisely, we have

$$\begin{aligned}\frac{1}{y} - \frac{1}{\hat{c}_n} \sum_{\ell=0}^{M_n} \left(1 - \frac{y}{\hat{c}_n} \right)^{\ell} &= \frac{1}{y} \left(1 - \frac{y}{\hat{c}_n} \right)^{M_n} \\ &\rightarrow 0, \quad n \rightarrow \infty,\end{aligned}$$

as long as

$$\frac{\hat{c}_n}{M_n} \rightarrow 0, \quad n \rightarrow \infty. \quad (27)$$

This simple calculation suggests that we can replace $\hat{c}$ and M in the formula for the estimator (cf. Equation (19)) by terms that depend on n and satisfy the condition expressed by Equation (27).

4 Experiments

4.1 Corpora

In our experiments we used the following corpora:

1. The *British National Corpus* (BNC): A corpus of about 100 million words of written and spoken British English from the years 1975-1994.

2. The *New York Times Corpus* (NYT): A corpus of about 5 million words.

3. The *Malayalam Corpus* (MAL): A collection of about 2.5 million words from varied articles in the Malayalam language from the Central Institute of Indian Languages.

4. The *Hindi Corpus* (HIN): A collection of about 3 million words from varied articles in the Hindi language also from the Central Institute of Indian Languages.

4.2 Methodology

We would like to see how well our estimator performs in terms of estimating the number of unseen elements. A natural way to study this is to expose only half of an existing corpus to be observed and estimate the number of unseen elements (assuming the the actual corpus is twice the observed size). We can then check numerically how well our estimator performs with respect to the "true" value. We use a subset (the first 10%, 20%, 30%, 40% and 50%) of the corpus as the *observed sample* to estimate the vocabulary over twice the sample size. The following estimators have been compared.

Nonparametric: Along with our proposed estimator (in Section 3), the following canonical estimators available in (Gandolfi and Sastri, 2004) and (Baayen, 2001) are studied.

1. Our proposed estimator $\mathbb{O}_n$ (cf. Section 3): since the estimator is rather involved we consider only small values of M (we see empirically that the estimator converges for very small values of M itself) and choose $\hat{c} = M$. This allows our estimator for the number of unseen elements to be of the following form, for different values of M:

M	$\mathbb{O}_n$
1	$2(\varphi_1 - \varphi_2)$
2	$\frac{3}{2}(\varphi_1 - \varphi_2) + \frac{3}{4}\varphi_3$
3	$\frac{4}{3}(\varphi_1 - \varphi_2) + \frac{8}{9}\left(\varphi_3 - \frac{\varphi_4}{3}\right)$

Using this, the estimator of the true vocabulary size is simply,

$$\mathbb{O}_n + V. \tag{28}$$

Here (cf. Equation (5))

$$V = \sum_{k=1}^{n} \varphi_k^n. \tag{29}$$

In the simulations below, we have considered M large enough until we see numerical convergence of the estimators: in all the cases, no more than a value of 4 is needed for M. For the English corpora, very small values of M suffice – in particular, we have considered the average of the first three different estimators (corresponding to the first three values of M). For the non-English corpora, we have needed to consider $M = 4$.

2. Gandolfi-Sastri estimator,

$$V_{\mathrm{GS}} \stackrel{\mathrm{def}}{=} \frac{n}{n - \varphi_1}\left(V + \varphi_1\gamma^2\right), \tag{30}$$

where

$$\gamma^2 = \frac{\varphi_1 - n - V}{2n} + \frac{\sqrt{5n^2 + 2n(V - 3\varphi_1) + (V - \varphi_1)^2}}{2n};$$

3. Chao estimator,

$$V_{\mathrm{Chao}} \stackrel{\mathrm{def}}{=} V + \frac{\varphi_1^2}{2\varphi_2}; \tag{31}$$

4. Good-Turing estimator,

$$V_{\mathrm{GT}} \stackrel{\mathrm{def}}{=} \frac{V}{\left(1 - \frac{\varphi_1}{n}\right)}; \tag{32}$$

5. "Simplistic" estimator,

$$V_{\mathrm{Smpl}} \stackrel{\mathrm{def}}{=} V\left(\frac{n_{\mathrm{new}}}{n}\right); \tag{33}$$

here the supposition is that the vocabulary size scales linearly with the sample size (here n_{new} is the new sample size);

6. Baayen estimator,

$$V_{\mathrm{Byn}} \stackrel{\mathrm{def}}{=} V + \left(\frac{\varphi_1}{n}\right) n_{new}; \tag{34}$$

here the supposition is that the vocabulary growth rate at the observed sample size is given by the ratio of the number of *hapax legomena* to the sample size (cf. (Baayen, 2001) pp. 50).

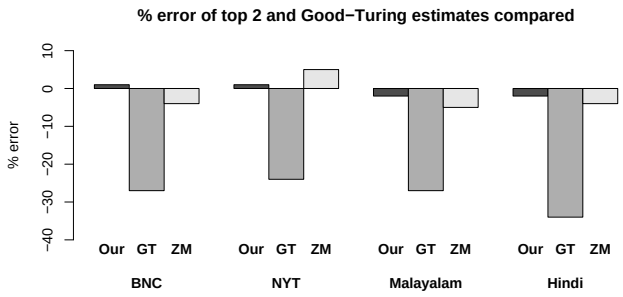


Figure 1: Comparison of error estimates of the 2 best estimators-ours and the ZM, with the Good-Turing estimator using 10% sample size of all the corpora. A bar with a positive height indicates and overestimate and that with a negative height indicates and underestimate. Our estimator *outperforms* ZM. Good-Turing estimator widely *underestimates* vocabulary size.

Parametric: Parametric estimators use the observations to first estimate the parameters. Then the corresponding models are used to estimate the vocabulary size over the larger sample. Thus the frequency spectra of the observations are only *indirectly* used in extrapolating the vocabulary size. In this study we consider state of the art parametric estimators, as surveyed by (Baroni and Evert, 2005). We are aided in this study by the availability of the implementations provided by the `ZipfR` package and their default settings.

5 Results and Discussion

The performance of the different estimators as percentage errors of the true vocabulary size using different corpora are tabulated in tables 1-4. We now summarize some important observations.

- From the Figure 1, we see that our estimator compares quite favorably with the best of the state of the art estimators. The best of the state of the art estimator is a parametric one (ZM), while ours is a nonparametric estimator.
- In table 1 and table 2 we see that our estimate is quite close to the true vocabulary, at all sample sizes. Further, it compares very favorably to the state of the art estimators (both parametric and nonparametric).
- Again, on the two non-English corpora (tables 3 and 4) we see that our estimator compares favorably with the best estimator of vocabulary size and at some sample sizes even surpasses it.
- Our estimator has theoretical performance guarantees and its empirical performance is comparable to that of the state of the art estimators. However, this performance comes at a very small fraction of the computational cost of the parametric estimators.
- The state of the art nonparametric Good-Turing estimator wildly underestimates the vocabulary; this is true in each of the four corpora studied and at all sample sizes.

6 Conclusion

In this paper, we have proposed a new nonparametric estimator of vocabulary size that takes into account the LNRE property of word frequency distributions and have shown that it is statistically consistent. We then compared the performance of the proposed estimator with that of the state of the art estimators on large corpora. While the performance of our estimator seems favorable, we also see that the widely used classical Good-Turing estimator consistently underestimates the vocabulary size. Although as yet untested, with its computational simplicity and favorable performance, our estimator may serve as a more reliable alternative to the Good-Turing estimator for estimating vocabulary sizes.

Acknowledgments

This research was partially supported by Award IIS-0623805 from the National Science Foundation.

References

R. H. Baayen. 2001. *Word Frequency Distributions*, Kluwer Academic Publishers.

Marco Baroni and Stefan Evert. 2001. “Testing the extrapolation quality of word frequency models”, *Proceedings of Corpus Linguistics , volume 1 of The Corpus Linguistics Conference Series*, P. Danielsson and M. Wagenmakers (eds.).

J. Bunge and M. Fitzpatrick. 1993. “Estimating the number of species: a review”, *Journal of the American Statistical Association*, Vol. 88(421), pp. 364-373.

Sample (% of corpus)	True value	% error w.r.t the true value							
		Our	GT	ZM	fZM	Smpl	Byn	Chao	GS
10	153912	1	-27	-4	-8	46	23	8	-11
20	220847	-3	-30	-9	-12	39	19	4	-15
30	265813	-2	-30	-9	-11	39	20	6	-15
40	310351	1	-29	-7	-9	42	23	9	-13
50	340890	2	-28	-6	-8	43	24	10	-12

Table 1: Comparison of estimates of vocabulary size for the **BNC corpus** as percentage errors w.r.t the true value. A negative value indicates an underestimate. Our estimator *outperforms* the other estimators at all sample sizes.

Sample (% of corpus)	True value	% error w.r.t the true value							
		Our	GT	ZM	fZM	Smpl	Byn	Chao	GS
10	37346	1	-24	5	-8	48	28	4	-8
20	51200	-3	-26	0	-11	46	22	-1	-11
30	60829	-2	-25	1	-10	48	23	1	-10
40	68774	-3	-25	0	-10	49	21	-1	-11
50	75526	-2	-25	0	-10	50	21	0	-10

Table 2: Comparison of estimates of vocabulary size for the **NYT corpus** as percentage errors w.r.t the true value. A negative value indicates an underestimate. Our estimator *compares favorably* with ZM and Chao.

Sample (% of corpus)	True value	% error w.r.t the true value							
		Our	GT	ZM	fZM	Smpl	Byn	Chao	GS
10	146547	-2	-27	-5	-10	9	34	82	-2
20	246723	8	-23	4	-2	19	47	105	5
30	339196	4	-27	0	-5	16	42	93	-1
40	422010	5	-28	1	-4	17	43	95	-1
50	500166	5	-28	1	-4	18	44	94	-2

Table 3: Comparison of estimates of vocabulary size for the **Malayalam corpus** as percentage errors w.r.t the true value. A negative value indicates an underestimate. Our estimator *compares favorably* with ZM and GS.

Sample (% of corpus)	True value	% error w.r.t the true value							
		Our	GT	ZM	fZM	Smpl	Byn	Chao	GS
10	47639	-2	-34	-4	-9	25	32	31	-12
20	71320	7	-30	2	-1	34	43	51	-7
30	93259	2	-33	-1	-5	30	38	42	-10
40	113186	0	-35	-5	-7	26	34	39	-13
50	131715	-1	-36	-6	-8	24	33	40	-14

Table 4: Comparison of estimates of vocabulary size for the **Hindi corpus** as percentage errors w.r.t the true value. A negative value indicates an underestimate. Our estimator *outperforms* the other estimators at certain sample sizes.

A. Gandolfi and C. C. A. Sastri. 2004. “Nonparametric Estimations about Species not Observed in a Random Sample”, *Milan Journal of Mathematics*, Vol. 72, pp. 81-105.

E. V. Khmaladze. 1987. “The statistical analysis of large number of rare events”, *Technical Report, Department of Mathematics and Statistics.*, CWI, Amsterdam, MS-R8804.

E. V. Khmaladze and R. J. Chitashvili. 1989. “Statistical analysis of large number of rate events and related problems”, *Probability theory and mathematical statistics* (Russian), Vol. 92, pp. 196-245.

. P. Santhanam, A. Orlitsky, and K. Viswanathan, “New tricks for old dogs: Large alphabet probability estimation”, in Proc. 2007 *IEEE Information Theory Workshop*, Sept. 2007, pp. 638–643.

A. B. Wagner, P. Viswanath and S. R. Kulkarni. 2006. “Strong Consistency of the Good-Turing estimator”, *IEEE Symposium on Information Theory*, 2006.

A Ranking Approach to Stress Prediction for Letter-to-Phoneme Conversion

Qing Dou, Shane Bergsma, Sittichai Jiampojamarn and Grzegorz Kondrak
Department of Computing Science
University of Alberta
Edmonton, AB, T6G 2E8, Canada
{qdou,bergsma,sj,kondrak}@cs.ualberta.ca

Abstract

Correct stress placement is important in text-to-speech systems, in terms of both the overall accuracy and the naturalness of pronunciation. In this paper, we formulate stress assignment as a sequence prediction problem. We represent words as sequences of substrings, and use the substrings as features in a Support Vector Machine (SVM) ranker, which is trained to rank possible stress patterns. The ranking approach facilitates inclusion of arbitrary features over both the input sequence and output stress pattern. Our system advances the current state-of-the-art, predicting primary stress in English, German, and Dutch with up to 98% word accuracy on phonemes, and 96% on letters. The system is also highly accurate in predicting secondary stress. Finally, when applied in tandem with an L2P system, it substantially reduces the word error rate when predicting both phonemes and stress.

1 Introduction

In many languages, certain syllables in words are phonetically more prominent in terms of duration, pitch, and loudness. This phenomenon is referred to as *lexical stress*. In some languages, the location of stress is entirely predictable. For example, lexical stress regularly falls on the initial syllable in Hungarian, and on the penultimate syllable in Polish. In other languages, such as English and Russian, any syllable in the word can be stressed.

Correct stress placement is important in text-to-speech systems because it affects the accuracy of human word recognition (Tagliapietra and Tabossi, 2005; Arciuli and Cupples, 2006). However, the issue has often been ignored in previous letter-to-phoneme (L2P) systems. The systems that do generate stress markers often do not report separate figures on stress prediction accuracy, or they only provide results on a single language. Some only predict primary stress markers (Black et al., 1998; Webster, 2004; Demberg et al., 2007), while those that predict both primary and secondary stress generally achieve lower accuracy (Bagshaw, 1998; Coleman, 2000; Pearson et al., 2000).

In this paper, we formulate stress assignment as a sequence prediction problem. We divide each word into a sequence of substrings, and use these substrings as features for a Support Vector Machine (SVM) ranker. For a given sequence length, there is typically only a small number of stress patterns in use. The task of the SVM is to rank the true stress pattern above the small number of acceptable alternatives. This is the first system to predict stress within a powerful discriminative learning framework. By using a ranking approach, we enable the use of arbitrary features over the entire (input) sequence and (output) stress pattern. We show that the addition of a feature for the entire output sequence improves prediction accuracy.

Our experiments on English, German, and Dutch demonstrate that our ranking approach substantially outperforms previous systems. The SVM ranker achieves exceptional 96.2% word accuracy on the challenging task of predicting the full stress pattern in English. Moreover, when combining our stress predictions with a state-of-the-art L2P system (Jiampojamarn et al., 2008), we set a new standard for the combined prediction of phonemes and stress.

The paper is organized as follows. Section 2 provides background on lexical stress and a task definition. Section 3 presents our automatic stress prediction algorithm. In Section 4, we confirm the power of the discriminative approach with experiments on three languages. Section 5 describes how stress is integrated into L2P conversion.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 118–126,
Suntec, Singapore, 2-7 August 2009.

2 Background and Task Definition

There is a long history of research into the principles governing lexical stress placement. Zipf (1929) showed that stressed syllables are often those with low frequency in speech, while unstressed syllables are usually very common. Chomsky and Halle (1968) proposed a set of context-sensitive rules for producing English stress from underlying word forms. Due to its importance in text-to-speech, there is also a long history of computational stress prediction systems (Fudge, 1984; Church, 1985; Williams, 1987). While these early approaches depend on human definitions of vowel tensity, syllable weight, word etymology, etc., our work follows a recent trend of purely data-driven approaches to stress prediction (Black et al., 1998; Pearson et al., 2000; Webster, 2004; Demberg et al., 2007).

In many languages, only two levels of stress are distinguished: stressed and unstressed. However, some languages exhibit more than two levels of stress. For example, in the English word *economic*, the first and the third syllable are stressed, with the former receiving weaker emphasis than the latter. In this case, the initial syllable is said to carry a secondary stress. Although each word has only one primary stress, it may have any number of secondary stresses. Predicting the full stress pattern is therefore inherently more difficult than predicting the location of primary stress only.

Our objective is to automatically assign primary and, where possible, secondary stress to out-of-vocabulary words. Stress is an attribute of syllables, but syllabification is a non-trivial task in itself (Bartlett et al., 2008). Rather than assuming correct syllabification of the input word, we instead follow Webster (2004) in placing the stress on the vowel which constitutes the nucleus of the stressed syllable. If the syllable boundaries are known, the mapping from the vowel to the corresponding syllable is straightforward.

We investigate the assignment of stress to two related but different entities: the spoken word (represented by its phonetic transcription), and the written word (represented by its orthographic form). Although stress is a prosodic feature, assigning stress to written words ("stressed orthography") has been utilized as a preprocessing stage for the L2P task (Webster, 2004). This preprocessing is motivated by two factors. First, stress greatly influences the pronunciation of vowels in English (c.f., *allow* vs. *alloy*). Second, since phoneme predictors typically utilize only local context around a letter, they do not incorporate the global, long-range information that is especially predictive of stress, such as penultimate syllable emphasis associated with the suffix *-ation*. By taking stressed orthography as input, the L2P system is able to implicitly leverage morphological information beyond the local context.

Indicating stress on letters can also be helpful to humans, especially second-language learners. In some languages, such as Spanish, orthographic markers are obligatory in words with irregular stress. The location of stress is often explicitly marked in textbooks for students of Russian. In both languages, the standard method of indicating stress is to place an acute accent above the vowel bearing primary stress, e.g., *adiós*. The secondary stress in English can be indicated with a grave accent (Coleman, 2000), e.g., *prècéde*.

In summary, our task is to assign primary and secondary stress markers to stress-bearing vowels in an input word. The input word may be either phonemes or letters. If a stressed vowel is represented by more than one letter, we adopt the convention of marking the first vowel of the vowel sequence, e.g., *méeting*. In this way, we are able to focus on the task of stress prediction, without having to determine at the same time the exact syllable boundaries, or whether a vowel letter sequence represents one or more spoken vowels (e.g., *beating* vs. *be-at-i-fy*).

3 Automatic Stress Prediction

Our stress assignment system maps a word, $\mathbf{w}$, to a stressed-form of the word, $\bar{\mathbf{w}}$. We formulate stress assignment as a sequence prediction problem. The assignment is made in three stages:

(1) First, we map words to substrings ($\mathbf{s}$), the basic units in our sequence (Section 3.1).

(2) Then, a particular stress pattern ($\mathbf{t}$) is chosen for each substring sequence. We use a support vector machine (SVM) to rank the possible patterns for each sequence (Section 3.2).

(3) Finally, the stress pattern is used to produce the stressed-form of the word (Section 3.3).

Table 1 gives examples of words at each stage of the algorithm. We discuss each step in more detail.

Word		Substrings		Pattern		Word'
$\mathbf{w}$	$\rightarrow$	$\mathbf{s}$	$\rightarrow$	$\mathbf{t}$	$\rightarrow$	$\bar{\mathbf{w}}$
worker	$\rightarrow$	*wor-ker*	$\rightarrow$	1-0	$\rightarrow$	*wórker*
overdo	$\rightarrow$	*ov-ver-do*	$\rightarrow$	2-0-1	$\rightarrow$	*òverdó*
react	$\rightarrow$	*re-ac*	$\rightarrow$	0-1	$\rightarrow$	*reáct*
æbstrækt	$\rightarrow$	æb-træk	$\rightarrow$	0-1	$\rightarrow$	æbstrækt
prisid	$\rightarrow$	ri-sid	$\rightarrow$	2-1	$\rightarrow$	prìsíd

Table 1: The steps in our stress prediction system (with orthographic and phonetic prediction examples): (1) word splitting, (2) support vector ranking of stress patterns, and (3) pattern-to-vowel mapping.

3.1 Word Splitting

The first step in our approach is to represent the word as a sequence of N individual units: $\mathbf{w} \rightarrow \mathbf{s} = \{s_1\text{-}s_2\text{-...-}s_N\}$. These units are used to define the features and outputs used by the SVM ranker. Although we are ultimately interested in assigning stress to individual vowels in the phoneme and letter sequence, it is beneficial to represent the task in units larger than individual letters.

Our substrings are similar to syllables; they have a vowel as their nucleus and include consonant context. By approximating syllables, our substring patterns will allow us to learn recurrent stress regularities, as well as dependencies between neighboring substrings. Since determining syllable breaks is a non-trivial task, we instead adopt the following simple splitting technique. Each vowel in the word forms the nucleus of a substring. Any single preceding or following consonant is added to the substring unit. Thus, each substring consists of at most three symbols (Table 1).

Using shorter substrings reduces the sparsity of our training data; words like *cryer*, *dryer* and *fryer* are all mapped to the same form: *ry-er*. The SVM can thus generalize from observed words to similarly-spelled, unseen examples.

Since the number of vowels equals the number of syllables in the phonetic form of the word, applying this approach to phonemes will always generate the correct number of syllables. For letters, splitting may result in a different number of units than the true syllabification, e.g., *pronounce* $\rightarrow$ *ron-no-un-ce*. This does not prevent the system from producing the correct stress assignment after the pattern-to-vowel mapping stage (Section 3.3) is complete.

3.2 Stress Prediction with SVM Ranking

After creating a sequence of substring units, $\mathbf{s} = \{s_1\text{-}s_2\text{-...-}s_N\}$, the next step is to choose an output sequence, $\mathbf{t} = \{t_1\text{-}t_2\text{-...-}t_N\}$, that encodes whether each unit is stressed or unstressed. We use the number '1' to indicate that a substring receives primary stress, '2' for secondary stress, and '0' to indicate no stress. We call this output sequence the *stress pattern* for a word. Table 1 gives examples of words, substrings, and stress patterns.

We use supervised learning to train a system to predict the stress pattern. We generate training $(\mathbf{s}, \mathbf{t})$ pairs in the obvious way from our stress-marked training words, $\bar{\mathbf{w}}$. That is, we first extract the letter/phoneme portion, $\mathbf{w}$, and use it to create the substrings, $\mathbf{s}$. We then create the stress pattern, $\mathbf{t}$, using $\bar{\mathbf{w}}$'s stress markers. Given the training pairs, any sequence predictor can be used, for example a Conditional Random Field (CRF) (Lafferty et al., 2001) or a structured perceptron (Collins, 2002). However, we can take advantage of a unique property of our problem to use a more expressive framework than is typically used in sequence prediction.

The key observation is that the output space of possible stress patterns is actually fairly limited. Clopper (2002) shows that people have strong preferences for particular sequences of stress, and this is confirmed by our training data (Section 4.1). In English, for example, we find that for each set of spoken words with the same number of syllables, there are no more than fifteen different stress patterns. In total, among 55K English training examples, there are only 70 different stress patterns. In both German and Dutch there are only about 50 patterns in 250K examples.[1] Therefore, for a particular input sequence, we can safely limit our consideration to only the small set of output patterns of the same length.

Thus, unlike typical sequence predictors, we do not have to search for the highest-scoring output according to our model. We can enumerate the full set of outputs and simply choose the highest-scoring one. This enables a more expressive representation. We can define arbitrary features over the entire output sequence. In a typical CRF or structured perceptron approach, only output features that can be computed incrementally during search are used (e.g. Markov transition features that permit Viterbi search). Since search is not

[1] See (Dou, 2009) for more details.

needed here, we can exploit longer-range features.

Choosing the highest-scoring output from a fixed set is a ranking problem, and we provide the full ranking formulation below. Unlike previous ranking approaches (e.g. Collins and Koo (2005)), we do not rely on a generative model to produce a list of candidates. Candidates are chosen in advance from observed training patterns.

3.2.1 Ranking Formulation

For a substring sequence, $\mathbf{s}$, of length N, our task is to select the correct output pattern from the set of all length-N patterns observed in our training data, a set we denote as $\mathbf{T}_N$. We score each possible input-output combination using a linear model. Each substring sequence and possible output pattern, $(\mathbf{s}, \mathbf{t})$, is represented with a set of features, $\mathbf{\Phi}(\mathbf{s}, \mathbf{t})$. The score for a particular $(\mathbf{s}, \mathbf{t})$ combination is a weighted sum of these features, $\boldsymbol{\lambda} \cdot \mathbf{\Phi}(\mathbf{s}, \mathbf{t})$. The specific features we use are described in Section 3.2.2.

Let $\mathbf{t}^j$ be the stress pattern for the jth training sequence $\mathbf{s}^j$, both of length N. At training time, the weights, $\boldsymbol{\lambda}$, are chosen such that for each $\mathbf{s}^j$, the correct output pattern receives a higher score than other patterns of the same length: $\forall \mathbf{u} \in \mathbf{T}_N, \mathbf{u} \neq \mathbf{t}^j$,

$$\boldsymbol{\lambda} \cdot \mathbf{\Phi}(\mathbf{s}^j, \mathbf{t}^j) > \boldsymbol{\lambda} \cdot \mathbf{\Phi}(\mathbf{s}^j, \mathbf{u}) \qquad (1)$$

The set of constraints generated by Equation 1 are called *rank constraints*. They are created separately for every $(\mathbf{s}^j, \mathbf{t}^j)$ training pair. Essentially, each training pair is matched with a set of automatically-created negative examples. Each negative has an incorrect, but plausible, stress pattern, $\mathbf{u}$.

We adopt a Support Vector Machine (SVM) solution to these ranking constraints as described by Joachims (2002). The learner finds the weights that ensure a maximum (soft) margin separation between the correct scores and the competitors. We use an SVM because it has been successful in similar settings (learning with thousands of sparse features) for both ranking and classification tasks, and because an efficient implementation is available (Joachims, 1999).

At test time we simply score each possible output pattern using the learned weights. That is, for an input sequence $\mathbf{s}$ of length N, we compute $\boldsymbol{\lambda} \cdot \mathbf{\Phi}(\mathbf{s}, \mathbf{t})$ for all $\mathbf{t} \in \mathbf{T}_N$, and we take the highest scoring $\mathbf{t}$ as our output. Note that because we only consider previously-observed output patterns, it is impossible for our system to produce a nonsensical result, such as having two primary stresses in one word. Standard search-based sequence predictors need to be specially augmented with hard constraints in order to prevent such output (Roth and Yih, 2005).

Substring	s_i, t_i
	s_i, i, t_i
Context	s_{i-1}, t_i
	$s_{i-1}s_i, t_i$
	s_{i+1}, t_i
	$s_i s_{i+1}, t_i$
	$s_{i-1}s_i s_{i+1}, t_i$
Stress Pattern	$t_1 t_2 \ldots t_N$

Table 2: Feature Template

3.2.2 Features

The power of our ranker to identify the correct stress pattern depends on how expressive our features are. Table 2 shows the feature templates used to create the features $\mathbf{\Phi}(\mathbf{s}, \mathbf{t})$ for our ranker. We use binary features to indicate whether each combination occurs in the current $(\mathbf{s},\mathbf{t})$ pair.

For example, if a substring *tion* is unstressed in a $(\mathbf{s}, \mathbf{t})$ pair, the *Substring* feature $\{s_i, t_i = \textit{tion}, 0\}$ will be true.[2] In English, often the penultimate syllable is stressed if the final syllable is *tion*. We can capture such a regularity with the *Context* feature s_{i+1}, t_i. If the following syllable is *tion* and the current syllable is stressed, the feature $\{s_{i+1}, t_i = \textit{tion}, 1\}$ will be true. This feature will likely receive a positive weight, so that output sequences with a stress before *tion* receive a higher rank.

Finally, the full *Stress Pattern* serves as an important feature. Note that such a feature would not be possible in standard sequence predictors, where such information must be decomposed into Markov transition features like $t_{i-1}t_i$. In a ranking framework, we can score output sequences using their full output pattern. Thus we can easily learn the rules in languages with regular stress rules. For languages that do not have a fixed stress rule, preferences for particular patterns can be learned using this feature.

[2] *tion* is a substring composed of three phonemes but we use its orthographic representation here for clarity.

3.3 Pattern-to-Vowel Mapping

The final stage of our system uses the predicted pattern $\mathbf{t}$ to create the stress-marked form of the word, $\bar{\mathbf{w}}$. Note the number of substrings created by our splitting method always equals the number of vowels in the word. We can thus simply map the indicator numbers in $\mathbf{t}$ to markers on their corresponding vowels to produce the stressed word.

For our example, *pronounce* → *ron-no-un-ce*, if the SVM chooses the stress pattern, 0-1-0-0, we produce the correct stress-marked word, *pronóunce*. If we instead stress the third vowel, 0-0-1-0, we produce an incorrect output, *pronoúnce*.

4 Stress Prediction Experiments

In this section, we evaluate our ranking approach to stress prediction by assigning stress to spoken and written words in three languages: English, German, and Dutch. We first describe the data and the various systems we evaluate, and then provide the results.

4.1 Data

The data is extracted from CELEX (Baayen et al., 1996). Following previous work on stress prediction, we randomly partition the data into 85% for training, 5% for development, and 10% for testing. To make results on German and Dutch comparable with English, we reduce the training, development, and testing set by 80% for each. After removing all duplicated items as well as abbreviations, phrases, and diacritics, each training set contains around 55K words.

In CELEX, stress is labeled on syllables in the phonetic form of the words. Since our objective is to assign stress markers to *vowels* (as described in Section 2) we automatically map the stress markers from the stressed syllables in the phonetic forms onto phonemes and letters representing vowels. For phonemes, the process is straightforward: we move the stress marker from the beginning of a syllable to the phoneme which constitutes the nucleus of the syllable. For letters, we map the stress from the vowel phoneme onto the orthographic forms using the ALINE algorithm (Dwyer and Kondrak, 2009). The stress marker is placed on the first letter within the syllable that represents a vowel sound.[3]

[3]Our stand-off stress annotations for English, German, and Dutch CELEX orthographic data can be downloaded at: http://www.cs.ualberta.ca/~kondrak/celex.html.

System	Eng		Ger	Dut
	P+*S*	*P*	*P*	*P*
SUBSTRING	96.2	98.0	97.1	93.1
ORACLESYL	95.4	96.4	97.1	93.2
TOPPATTERN	66.8	68.9	64.1	60.8

Table 3: Stress prediction word accuracy (%) on **phonemes** for English, German, and Dutch. *P*: predicting primary stress only. *P*+*S*: primary and secondary.

CELEX also provides secondary stress annotation for English. We therefore evaluate on both primary and secondary stress (*P*+*S*) in English and on primary stress assignment alone (*P*) for English, German, and Dutch.

4.2 Comparison Approaches

We evaluate three different systems on the letter and phoneme sequences in the experimental data:

1) SUBSTRING is the system presented in Section 3. It uses the vowel-based splitting method, followed by SVM ranking.

2) ORACLESYL splits the input word into syllables according to the CELEX gold-standard, before applying SVM ranking. The output pattern is evaluated directly against the gold-standard, without pattern-to-vowel mapping.

3) TOPPATTERN is our baseline system. It uses the vowel-based splitting method to produce a substring sequence of length N. Then it simply chooses the most common stress pattern among all the stress patterns of length N.

SUBSTRING and ORACLESYL use scores produced by an SVM ranker trained on the training data. We employ the ranking mode of the popular learning package SVM^{light} (Joachims, 1999). In each case, we learn a linear kernel ranker on the training set stress patterns and tune the parameter that trades-off training error and margin on the development set.

We evaluate the systems using *word accuracy*: the percent of words for which the output form of the word, $\bar{\mathbf{w}}$, matches the gold standard.

4.3 Results

Table 3 provides results on English, German, and Dutch phonemes. Overall, the performance of our automatic stress predictor, SUBSTRING, is excellent. It achieves 98.0% accuracy for predicting

System	Eng		Ger	Dut
	P+S	*P*	*P*	*P*
SUBSTRING	93.5	95.1	95.9	91.0
ORACLESYL	94.6	96.0	96.6	92.8
TOPPATTERN	65.5	67.6	64.1	60.8

Table 4: Stress prediction word accuracy (%) on **letters** for English, German, and Dutch. *P*: predicting primary stress only. *P+S*: primary and secondary.

primary stress in English, 97.1% in German, and 93.1% in Dutch. It also predicts both primary and secondary stress in English with high accuracy, 96.2%. Performance is much higher than our baseline accuracy, which is between 60% and 70%. ORACLESYL, with longer substrings and hence sparser data, does not generally improve performance. This indicates that perfect syllabification is unnecessary for phonetic stress assignment.

Our system is a major advance over the previous state-of-the-art in phonetic stress assignment. For predicting stressed/unstressed syllables in English, Black et al. (1998) obtained a per-syllable accuracy of 94.6%. We achieve 96.2% *per-word* accuracy for predicting both primary and secondary stress. Others report lower numbers on English phonemes. Bagshaw (1998) obtained 65%-83.3% per-syllable accuracy using Church (1985)'s rule-based system. For predicting both primary and secondary stress, Coleman (2000) and Pearson et al. (2000) report 69.8% and 81.0% word accuracy, respectively.

The performance on letters (Table 4) is also quite encouraging. SUBSTRING predicts primary stress with accuracy above 95% for English and German, and equal to 91% in Dutch. Performance is 1-3% lower on letters than on phonemes. On the other hand, the performance of ORACLESYL drops much less on letters. This indicates that most of SUBSTRING's errors are caused by the splitting method. Letter vowels may or may not represent spoken vowels. By creating a substring for every vowel letter we may produce an incorrect number of syllables. Our pattern feature is therefore less effective.

Nevertheless, SUBSTRING's accuracy on letters also represents a clear improvement over previous work. Webster (2004) reports 80.3% word accuracy on letters in English and 81.2% in German. The most comparable work is Demberg et al. (2007), which achieves 90.1% word accuracy on letters in German CELEX, assuming perfect letter syllabification. In order to reproduce their strict experimental setup, we re-partition the full set of German CELEX data to ensure that no overlap of word stems exists between the training and test sets. Using the new data sets, our system achieves a word accuracy of 92.3%, a 2.2% improvement over Demberg et al. (2007)'s result. Moreover, if we also assume perfect syllabification, the accuracy is 94.3%, a 40% reduction in error rate.

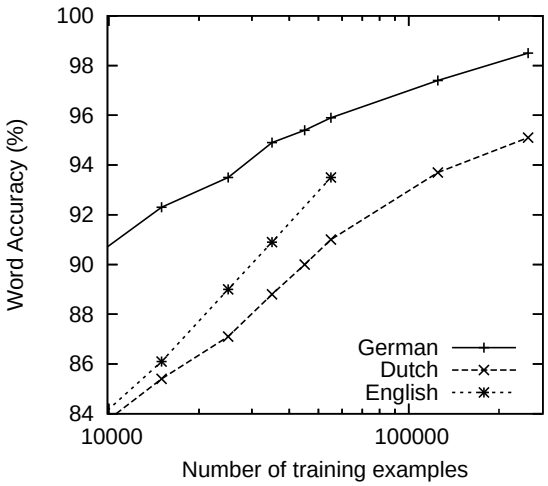


Figure 1: Stress prediction accuracy on letters.

We performed a detailed analysis to understand the strong performance of our system. First of all, note that an error could happen if a test-set stress pattern was not observed in the training data; its correct stress pattern would not be considered as an output. In fact, no more than two test errors in any test set were so caused. This strongly justifies the reduced set of outputs used in our ranking formulation.

We also tested all systems with the Stress Pattern feature removed. Results were worse in all cases. As expected, it is most valuable for predicting primary and secondary stress. On English phonemes, accuracy drops from 96.2% to 95.3% without it. On letters, it drops from 93.5% to 90.0%. The gain from this feature also validates our ranking framework, as such arbitrary features over the entire output sequence can not be used in standard search-based sequence prediction.

Finally, we examined the relationship between training data size and performance by plotting learning curves for letter stress accuracy (Figure 1). Unlike the tables above, here we use the

full set of data in Dutch and German CELEX to create the largest-possible training sets (255K examples). None of the curves are levelling off; performance grows log-linearly across the full range.

5 Lexical stress and L2P conversion

In this section, we evaluate various methods of combining stress prediction with phoneme generation. We first describe the specific system that we use for letter-to-phoneme (L2P) conversion. We then discuss the different ways stress prediction can be integrated with L2P, and define the systems used in our experiments. Finally, we provide the results.

5.1 The L2P system

We combine stress prediction with a state-of-the-art L2P system (Jiampojamarn et al., 2008). Like our stress ranker, their system is a data-driven sequence predictor that is trained with supervised learning. The score for each output sequence is a weighted combination of features. The feature weights are trained using the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003), a powerful online discriminative training framework. Like other recent L2P systems (Bisani and Ney, 2002; Marchand and Damper, 2007; Jiampojamarn et al., 2007), this approach does not generate stress, nor does it consider stress when it generates phonemes.

For L2P experiments, we use the same training, testing, and development data as was used in Section 4. For all experiments, we use the development set to determine at which iteration to stop training in the online algorithm.

5.2 Combining stress and phoneme generation

Various methods have been used for combining stress and phoneme generation. Phonemes can be generated without regard to stress, with stress assigned as a post-process (Bagshaw, 1998; Coleman, 2000). Both van den Bosch (1997) and Black et al. (1998) argue that stress should be predicted at the same time as phonemes. They expand the output set to distinguish between stressed and unstressed phonemes. Similarly, Demberg et al. (2007) produce phonemes, stress, and syllable-boundaries within a single joint n-gram model. Pearson et al. (2000) generate phonemes and stress together by jointly optimizing a decision-tree phoneme-generator and a stress predictor based on stress pattern counts. In contrast, Webster (2004) first assigns stress to letters, creating an expanded input set, and then predicts both phonemes and stress jointly. The system marks stress on letter vowels by determining the correspondence between affixes and stress in written words.

Following the above approaches, we can expand the input or output symbols of our L2P system to include stress. However, since both decision tree systems and our L2P predictor utilize only local context, they may produce invalid global output. One option, used by Demberg et al. (2007), is to add a constraint to the output generation, requiring each output sequence to have exactly one primary stress.

We enhance this constraint, based on the observation that the number of valid output sequences is fairly limited (Section 3.2). The modified system produces the highest-scoring sequence such that the output's corresponding stress pattern has been observed in our training data. We call this the **stress pattern constraint**. This is a tighter constraint than having only one primary stress.[4] Another advantage is that it provides some guidance for the assignment of secondary stress.

Inspired by the aforementioned strategies, we evaluate the following approaches:

1) JOINT: The L2P system's input sequence is letters, the output sequence is phonemes+stress.

2) JOINT+CONSTR: Same as JOINT, except it selects the highest scoring output that obeys the stress pattern constraint.

3) POSTPROCESS: The L2P system's input is letters, the output is phonemes. It then applies the SVM stress ranker (Section 3) to the phonemes to produce the full phoneme+stress output.

4) LETTERSTRESS: The L2P system's input is letters+stress, the output is phonemes+stress. It creates the stress-marked letters by applying the SVM ranker to the input letters as a preprocess.

5) ORACLESTRESS: The same input/output as LETTERSTRESS, except it uses the gold-standard stress on letters (Section 4.1).

[4] In practice, the L2P system generates a top-N list, and we take the highest-scoring output on the list that satisfies the constraint. If none satisfy the constraint, we take the top output that has only one primary stress.

System	Eng		Ger	Dut
	P+S	*P*	*P*	*P*
JOINT	78.9	80.0	86.0	81.1
JOINT+CONSTR	84.6	86.0	90.8	88.7
POSTPROCESS	86.2	87.6	90.9	88.8
LETTERSTRESS	86.5	87.2	90.1	86.6
ORACLESTRESS	91.4	91.4	92.6	94.5
Festival	61.2	62.5	71.8	65.1

Table 5: Combined phoneme *and* stress prediction word accuracy (%) for English, German, and Dutch. *P*: predicting primary stress only. *P+S*: primary and secondary.

Note that while the first approach uses only local information to make predictions (features within a context window around the current letter), systems 2 to 5 leverage global information in some manner: systems 3 and 4 use the predictions of our stress ranker, while 2 uses a global stress pattern constraint.[5]

We also generated stress and phonemes using the popular Festival Speech Synthesis System[6] (version 1.96, 2004) and report its accuracy.

5.3 Results

Word accuracy results for predicting both phonemes and stress are provided in Table 5. First of all, note that the JOINT approach, which simply expands the output set, is 4%-8% worse than all other comparison systems across the three languages. These results clearly indicate the drawbacks of predicting stress using only local information. In English, both LETTERSTRESS and POSTPROCESS perform best, while POSTPROCESS and the constrained system are highest on German and Dutch. Results using the oracle letter stress show that given perfect stress assignment on letters, phonemes and stress can be predicted very accurately, in all cases above 91%.

We also found that the phoneme prediction accuracy alone (i.e., without stress) is quite similar for all the systems. The gains over JOINT on combined stress and phoneme accuracy are almost entirely due to more accurate stress assignment. Utilizing the oracle stress on letters markedly improves phoneme prediction in English (from 88.8% to 91.4%). This can be explained by the fact that English vowels are often reduced to schwa when unstressed (Section 2).

Predicting both phonemes and stress is a challenging task, and each of our globally-informed systems represents a major improvement over previous work. The accuracy of Festival is much lower even than our JOINT approach, but the relative performance on the different languages is quite similar.

A few papers report accuracy on the combined stress and phoneme prediction task. The most directly comparable work is van den Bosch (1997), which also predicts primary and secondary stress using English CELEX data. However, the reported word accuracy is only 62.1%. Three other papers report word accuracy on phonemes and stress, using different data sets. Pearson et al. (2000) report 58.5% word accuracy for predicting phonemes and primary/secondary stress. Black et al. (1998) report 74.6% word accuracy in English, while Webster (2004) reports 68.2% on English and 82.9% in German (all primary stress only). Finally, Demberg et al. (2007) report word accuracy on predicting phonemes, stress, *and* syllabification on German CELEX data. They achieve 86.3% word accuracy.

6 Conclusion

We have presented a discriminative ranking approach to lexical stress prediction, which clearly outperforms previously developed systems. The approach is largely language-independent, applicable to both orthographic and phonetic representations, and flexible enough to handle multiple stress levels. When combined with an existing L2P system, it achieves impressive accuracy in generating pronunciations together with their stress patterns. In the future, we will investigate additional features to leverage syllabic and morphological information, when available. Kernel functions could also be used to automatically create a richer feature space; preliminary experiments have shown gains in performance using polynomial and RBF kernels with our stress ranker.

Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council of Canada, the Alberta Ingenuity Fund, and the Alberta Informatics Circle of Research Excellence.

[5] This constraint could also help the other systems. However, since they already use global information, it yields only marginal improvements.

[6] http://www.cstr.ed.ac.uk/projects/festival/

References

Joanne Arciuli and Linda Cupples. 2006. The processing of lexical stress during visual word recognition: Typicality effects and orthographic correlates. *Quarterly Journal of Experimental Psychology*, 59(5):920–948.

Harald Baayen, Richard Piepenbrock, and Leon Gulikers. 1996. The CELEX2 lexical database. LDC96L14.

Paul C. Bagshaw. 1998. Phonemic transcription by analogy in text-to-speech synthesis: Novel word pronunciation and lexicon compression. *Computer Speech and Language*, 12(2):119–142.

Susan Bartlett, Grzegorz Kondrak, and Colin Cherry. 2008. Automatic syllabification with structured SVMs for letter-to-phoneme conversion. In *ACL-08: HLT*, pages 568–576.

Maximilian Bisani and Hermann Ney. 2002. Investigations on joint-multigram models for grapheme-to-phoneme conversion. In *ICSLP*, pages 105–108.

Alan W Black, Kevin Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules. In *The 3rd ESCA Workshop on Speech Synthesis*, pages 77–80.

Noam Chomsky and Morris Halle. 1968. The sound pattern of English. *New York: Harper and Row*.

Kenneth Church. 1985. Stress assignment in letter to sound rules for speech synthesis. In *ACL*, pages 246–253.

Cynthia G. Clopper. 2002. Frequency of stress patterns in English: A computational analysis. *IULC Working Papers Online*.

John Coleman. 2000. Improved prediction of stress in out-of-vocabulary words. In *IEEE Seminar on the State of the Art in Speech Synthesis*.

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

Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8.

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

Vera Demberg, Helmut Schmid, and Gregor Möhler. 2007. Phonological constraints and morphological preprocessing for grapheme-to-phoneme conversion. In *ACL*, pages 96–103.

Qing Dou. 2009. An SVM ranking approach to stress assignment. Master's thesis, University of Alberta.

Kenneth Dwyer and Grzegorz Kondrak. 2009. Reducing the annotation effort for letter-to-phoneme conversion. In *ACL-IJCNLP*.

Erik C. Fudge. 1984. English word-stress. *London: Allen and Unwin*.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and Hidden Markov Models to letter-to-phoneme conversion. In *NAACL-HLT 2007*, pages 372–379.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. In *ACL-08: HLT*, pages 905–913.

Thorsten Joachims. 1999. Making large-scale Support Vector Machine learning practical. In B. Schölkopf and C. Burges, editors, *Advances in Kernel Methods: Support Vector Machines*, pages 169–184. MIT-Press.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

Yannick Marchand and Robert I. Damper. 2007. Can syllabification improve pronunciation by analogy of English? *Natural Language Engineering*, 13(1):1–24.

Steve Pearson, Roland Kuhn, Steven Fincke, and Nick Kibre. 2000. Automatic methods for lexical stress assignment and syllabification. In *ICSLP*, pages 423–426.

Dan Roth and Wen-tau Yih. 2005. Integer linear programming inference for conditional random fields. In *ICML*, pages 736–743.

Lara Tagliapietra and Patrizia Tabossi. 2005. Lexical stress effects in Italian spoken word recognition. In *The XXVII Annual Conference of the Cognitive Science Society*, pages 2140–2144.

Antal van den Bosch. 1997. *Learning to pronounce written words: A study in inductive language learning*. Ph.D. thesis, Universiteit Maastricht.

Gabriel Webster. 2004. Improving letter-to-pronunciation accuracy with automatic morphologically-based stress prediction. In *ICSLP*, pages 2573–2576.

Briony Williams. 1987. Word stress assignment in a text-to-speech synthesis system for British English. *Computer Speech and Language*, 2:235–272.

George Kingsley Zipf. 1929. Relative frequency as a determinant of phonetic change. *Harvard Studies in Classical Philology*, 15:1–95.

Reducing the Annotation Effort for Letter-to-Phoneme Conversion

Kenneth Dwyer and **Grzegorz Kondrak**
Department of Computing Science
University of Alberta
Edmonton, AB, Canada, T6G 2E8
{dwyer,kondrak}@cs.ualberta.ca

Abstract

Letter-to-phoneme (L2P) conversion is the process of producing a correct phoneme sequence for a word, given its letters. It is often desirable to reduce the quantity of training data — and hence human annotation — that is needed to train an L2P classifier for a new language. In this paper, we confront the challenge of building an accurate L2P classifier with a minimal amount of training data by combining several diverse techniques: context ordering, letter clustering, active learning, and phonetic L2P alignment. Experiments on six languages show up to 75% reduction in annotation effort.

1 Introduction

The task of letter-to-phoneme (L2P) conversion is to produce a correct sequence of phonemes, given the letters that comprise a word. An accurate L2P converter is an important component of a text-to-speech system. In general, a lookup table does not suffice for L2P conversion, since out-of-vocabulary words (e.g., proper names) are inevitably encountered. This motivates the need for classification techniques that can predict the phonemes for an unseen word.

Numerous studies have contributed to the development of increasingly accurate L2P systems (Black et al., 1998; Kienappel and Kneser, 2001; Bisani and Ney, 2002; Demberg et al., 2007; Jiampojamarn et al., 2008). A common assumption made in these works is that ample amounts of labelled data are available for training a classifier. Yet, in practice, this is the case for only a small number of languages. In order to train an L2P classifier for a new language, we must first annotate words in that language with their correct phoneme sequences. As annotation is expensive, we would like to minimize the amount of effort that is required to build an adequate training set. The objective of this work is not necessarily to achieve state-of-the-art performance when presented with large amounts of training data, but to outperform other approaches when training data is limited.

This paper proposes a system for training an accurate L2P classifier while requiring as few annotated words as possible. We employ decision trees as our supervised learning method because of their transparency and flexibility. We incorporate context ordering into a decision tree learner that guides its tree-growing procedure towards generating more intuitive rules. A clustering over letters serves as a back-off model in cases where individual letter counts are unreliable. An active learning technique is employed to request the phonemes (labels) for the words that are expected to be the most informative. Finally, we apply a novel L2P alignment technique based on phonetic similarity, which results in impressive gains in accuracy without relying on any training data.

Our empirical evaluation on several L2P datasets demonstrates that significant reductions in annotation effort are indeed possible in this domain. Individually, all four enhancements improve the accuracy of our decision tree learner. The combined system yields savings of up to 75% in the number of words that have to be labelled, and reductions of at least 52% are observed on all the datasets. This is achieved without any additional tuning for the various languages.

The paper is organized as follows. Section 2 explains how supervised learning for L2P conversion is carried out with decision trees, our classifier of choice. Sections 3 through 6 describe our four main contributions towards reducing the annotation effort for L2P: context ordering (Section 3), clustering letters (Section 4), active learning (Section 5), and phonetic alignment (Section 6). Our experimental setup and results are discussed in

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 127–135,
Suntec, Singapore, 2-7 August 2009.

Sections 7 and 8, respectively. Finally, Section 9 offers some concluding remarks.

2 Decision tree learning of L2P classifiers

In this work, we employ a decision tree model to learn the mapping from words to phoneme sequences. Decision tree learners are attractive because they are relatively fast to train, require little or no parameter tuning, and the resulting classifier can be interpreted by the user. A number of prior studies have applied decision trees to L2P data and have reported good generalization accuracy (Andersen et al., 1996; Black et al., 1998; Kienappel and Kneser, 2001). Also, the widely-used Festival Speech Synthesis System (Taylor et al., 1998) relies on decision trees for L2P conversion.

We adopt the standard approach of using the letter context as features. The decision tree predicts the phoneme for the focus letter based on the m letters that appear before and after it in the word (including the focus letter itself, and beginning/end of word markers, where applicable). The model predicts a phoneme independently for each letter in a given word. In order to keep our model simple and transparent, we do not explore the possibility of conditioning on adjacent (predicted) phonemes. Any improvement in accuracy resulting from the inclusion of phoneme features would also be realized by the baseline that we compare against, and thus would not materially influence our findings.

We employ *binary* decision trees because they substantially outperformed *n-ary* trees in our preliminary experiments. In L2P, there are many unique values for each attribute, namely, the letters of a given alphabet. In a *n*-ary tree each decision node partitions the data into *n* subsets, one per letter, that are potentially sparse. By contrast, a binary tree creates one branch for the nominated letter, and one branch grouping the remaining letters into a single subset. In the forthcoming experiments, we use binary decision trees exclusively.

3 Context ordering

In the L2P task, context letters that are adjacent to the focus letter tend to be more important than context letters that are further away. For example, the English letter c is usually pronounced as [s] if the following letter is e or i. The general tree-growing algorithm has no notion of the letter distance, but instead chooses the letters on the basis of their estimated *information gain* (Manning and Schütze, 1999). As a result, it will sometimes query a letter at position $+3$ (denoted l_3), for example, before examining the letters that are closer to the center of the context window.

We propose to modify the tree-growing procedure to encourage the selection of letters near the focus letter before those at greater offsets are examined. In its strictest form, which resembles the "dynamically expanding context" search strategy of Davel and Barnard (2004), l_i can only be queried after $l_0, \ldots, l_{i-1}$ have been queried. However, this approach seems overly rigid for L2P. In English, for example, l_2 can directly influence the pronunciation of a vowel regardless of the value of l_1 (c.f., the difference between *rid* and *ride*).

Instead, we adopt a less intrusive strategy, which we refer to as "context ordering," that biases the decision tree toward letters that are closer to the focus, but permits gaps when the information gain for a distant letter is relatively high. Specifically, the ordering constraint described above is still applied, but only to letters that have above-average information gain (where the average is calculated across all letters/attributes). This means that a letter with above-average gain that is eligible with respect to the ordering will take precedence over an ineligible letter that has an even higher gain. However, if all the eligible letters have below-average gain, the ineligible letter with the highest gain is selected irrespective of its position. Our only strict requirement is that the focus letter must always be queried first, unless its information gain is zero.

Kienappel and Kneser (2001) also worked on improving decision tree performance for L2P, and devised tie-breaking rules in the event that the tree-growing procedure ranked two or more questions as being equally informative. In our experience with L2P datasets, exact ties are rare; our context ordering mechanism will have more opportunities to guide the tree-growing process. We expect this change to improve accuracy, especially when the amount of training data is very limited. By biasing the decision tree learner toward questions that are intuitively of greater utility, we make it less prone to overfitting on small data samples.

4 Clustering letters

A decision tree trained on L2P data bases its phonetic predictions on the surrounding letter context.

Yet, when making predictions for unseen words, contexts will inevitably be encountered that did not appear in the training data. Instead of relying solely on the particular letters that surround the focus letter, we postulate that the learner could achieve better generalization if it had access to information about the *types* of letters that appear before and after. That is, instead of treating letters as abstract symbols, we would like to encode knowledge of the similarity between certain letters as features. One way of achieving this goal is to group the letters into classes or clusters based on their contextual similarity. Then, when a prediction has to be made for an unseen (or low probability) letter sequence, the letter classes can provide additional information.

Kienappel and Kneser (2001) report accuracy gains when applying letter clustering to the L2P task. However, their decision tree learner incorporates neighboring phoneme predictions, and employs a variety of different pruning strategies; the portion of the gains attributable to letter clustering are not evident. In addition to exploring the effect of letter clustering on a wider range of languages, we are particularly concerned with the impact that clustering has on decision tree performance when the training set is small. The addition of letter class features to the data may enable the active learner to better evaluate candidate words in the pool, and therefore make more informed selections.

To group the letters into classes, we employ a hierarchical clustering algorithm (Brown et al., 1992). One advantage of inducing a hierarchy is that we need not commit to a particular level of granularity; in other words, we are not required to specify the number of classes beforehand, as is the case with some other clustering algorithms.[1]

The clustering algorithm is initialized by placing each letter in its own class, and then proceeds in a bottom-up manner. At each step, the pair of classes is merged that leads to the smallest loss in the average *mutual information* (Manning and Schütze, 1999) between adjacent classes. The merging process repeats until a single class remains that contains all the letters in the alphabet. Recall that in our problem setting we have access to a (presumably) large pool of unannotated words. The unigram and bigram frequencies required by the clustering algorithm are calculated from these words; hence, the letters can be grouped into classes prior to annotation. The letter classes only need to be computed once for a given language. We implemented a brute-force version of the algorithm that examines all the possible merges at each step, and generates a hierarchy within a few hours. However, when dealing with a larger number of unique tokens (e.g., when clustering words instead of letters), additional optimizations are needed in order to make the procedure tractable.

Letter	Bit String	Letter	Bit String
a	01000	n	1111
b	10000000	o	01001
c	10100	p	10001
d	11000	q	1000001
e	0101	r	111010
f	100001	s	11010
g	11001	t	101010
h	10110	u	0111
i	0110	v	100110
j	10000001	w	100111
k	10111	x	111011
l	11100	y	11011
m	10010	z	101011
		#	00

Table 1: Hierarchical clustering of English letters

The resulting hierarchy takes the form of a binary tree, where the root node/cluster contains all the letters, and each leaf contains a single letter. Hence, each letter can be represented by a *bit string* that describes the path from the root to its leaf. As an illustration, the clustering in Table 1 was automatically generated from the words in the English CMU Pronouncing Dictionary (Carnegie Mellon University, 1998). It is interesting to note that the first bit distinguishes vowels from consonants, meaning that these were the last two groups that were merged by the clustering algorithm. Note also that the beginning/end of word marker (#) is included in the hierarchy, and is the last character to be absorbed into a larger cluster. This indicates that # carries more information than most letters, as is to be expected, in light of its distinct status. We also experimented with a manually-constructed letter hierarchy, but observed no significant differences in accuracy vis-à-vis the automatic clustering.

[1]This approach is inspired by the work of Miller et al. (2004), who clustered *words* for a named-entity tagging task.

5 Active learning

Whereas a passive supervised learning algorithm is provided with a collection of training examples that are typically drawn at random, an active learner has control over the labelled data that it obtains (Cohn et al., 1992). The latter attempts to select its training set intelligently by requesting the labels of only those examples that are judged to be the most useful or informative. Numerous studies have demonstrated that active learners can make more efficient use of unlabelled data than do passive learners (Abe and Mamitsuka, 1998; Miller et al., 2004; Culotta and McCallum, 2005). However, relatively few researchers have applied active learning techniques to the L2P domain. This is despite the fact that annotated data for training an L2P classifier is not available in most languages. We briefly review two relevant studies before proceeding to describe our active learning strategy.

Maskey et al. (2004) propose a bootstrapping technique that iteratively requests the labels of the n most frequent words in a corpus. A classifier is trained on the words that have been annotated thus far, and then predicts the phonemes for each of the n words being considered. Words for which the prediction confidence is above a certain threshold are immediately added to the lexicon, while the remaining words must be verified (and corrected, if necessary) by a human annotator. The main drawback of such an approach lies in the risk of adding erroneous entries to the lexicon when the classifier is overly confident in a prediction.

Kominek and Black (2006) devise a word selection strategy based on letter n-gram coverage and word length. Their method slightly outperforms random selection, thereby establishing passive learning as a strong baseline. However, only a single Italian dataset was used, and the results do not necessarily generalize to other languages.

In this paper, we propose to apply an active learning technique known as *Query-by-Bagging* (Abe and Mamitsuka, 1998). We consider a pool-based active learning setting, whereby the learner has access to a pool of unlabelled examples (words), and may obtain labels (phoneme sequences) at a cost. This is an iterative procedure in which the learner trains a classifier on the current set of labelled training data, then selects one or more new examples to label, according to the classifier's predictions on the pool data. Once labelled, these examples are added to the training set, the classifier is re-trained, and the process repeats until some stopping criterion is met (e.g., annotation resources are exhausted).

Query-by-Bagging (QBB) is an instance of the Query-by-Committee algorithm (Freund et al., 1997), which selects examples that have high classification variance. At each iteration, QBB employs the bagging procedure (Breiman, 1996) to create a committee of classifiers C. Given a training set T containing k examples (in our setting, k is the total number of letters that have been labelled), bagging creates each committee member by sampling k times from T (with replacement), and then training a classifier C_i on the resulting data. The example in the pool that maximizes the disagreement among the predictions of the committee members is selected.

A crucial question is how to calculate the disagreement among the predicted phoneme sequences for a word in the pool. In the L2P domain, we assume that a human annotator specifies the phonemes for an entire word, and that the active learner cannot query individual letters. We require a measure of confidence at the word level; yet, our classifiers make predictions at the letter level. This is analogous to the task of estimating record confidence using field confidence scores in information extraction (Culotta and McCallum, 2004).

Our solution is as follows. Let $\mathbf{w}$ be a word in the pool. Each classifier C_i predicts the phoneme for each letter $l \in \mathbf{w}$. These "votes" are aggregated to produce a vector $\mathbf{v}_l$ for letter l that indicates the distribution of the $|C|$ predictions over its possible phonemes. We then compute the *margin* for each letter: If $\{p, p'\} \in \mathbf{v}_l$ are the two highest vote totals, then the margin is $M(\mathbf{v}_l) = |p - p'|$. A small margin indicates disagreement among the constituent classifiers. We define the disagreement score for the entire word as the minimum margin:

$$score(\mathbf{w}) = \min_{l \in \mathbf{w}} \{M(\mathbf{v}_l)\} \quad (1)$$

We also experimented with maximum vote entropy and average margin/entropy, where the average is taken over all the letters in a word. The minimum margin exhibited the best performance on our development data; hence, we do not provide a detailed evaluation of the other measures.

6 L2P alignment

Before supervised learning can take place, the letters in each word need to be aligned with

phonemes. However, a lexicon typically provides just the letter and phoneme sequences for each word, without specifying the specific phoneme(s) that each letter elicits. The sub-task of L2P that pairs letters with phonemes in the training data is referred to as *alignment*. The L2P alignments that are specified in the training data can influence the accuracy of the resulting L2P classifier. In our setting, we are interested in mapping each letter to either a single phoneme or the "null" phoneme.

The standard approach to L2P alignment is described by Damper et al. (2005). It performs an Expectation-Maximization (EM) procedure that takes a (preferably large) collection of words as input and computes alignments for them simultaneously. However, since in our active learning setting the data is acquired incrementally, we cannot count on the initial availability of a substantial set of words accompanied by their phonemic transcriptions.

In this paper, we apply the ALINE algorithm to the task of L2P alignment (Kondrak, 2000; Inkpen et al., 2007). ALINE, which performs phonetically-informed alignment of two strings of phonemes, requires no training data, and so is ideal for our purposes. Since our task requires the alignment of phonemes with *letters*, we wish to replace every letter with a phoneme that is the most likely to be produced by that letter. On the other hand, we would like our approach to be language-independent. Our solution is to simply treat every letter as an IPA symbol (International Phonetic Association, 1999). The IPA is based on the Roman alphabet, but also includes a number of other symbols. The 26 IPA letter symbols tend to correspond to the usual phonetic value that the letter represents in the Latin script.[2] For example, the IPA symbol [m] denotes "voiced bilabial nasal," which is the phoneme represented by the letter m in most languages that utilize Latin script.

The alignments produced by ALINE are of high quality. The example below shows the alignment of the Italian word *scianchi* to its phonetic transcription [ʃaŋki]. ALINE correctly aligns not only identical IPA symbols (i:i), but also IPA symbols that represent similar sounds (s:ʃ, n:ŋ, c:k).

s	c	i	a	n	c	h	i
|			|	|	|		|
ʃ			a	ŋ	k		i

[2] ALINE can also be applied to non-Latin scripts by replacing every grapheme with the IPA symbol that is phonetically closest to it (Jiampojamarn et al., 2009).

7 Experimental setup

We performed experiments on six datasets, which were obtained from the PRONALSYL letter-to-phoneme conversion challenge.[3] They are: English CMUDict (Carnegie Mellon University, 1998); French BRULEX (Content et al., 1990), Dutch and German CELEX (Baayen et al., 1996), the Italian Festival dictionary (Cosi et al., 2000), and the Spanish lexicon. Duplicate words and words containing punctuation or numerals were removed, as were abbreviations and acronyms. The resulting datasets range in size from 31,491 to 111,897 words. The PRONALSYL datasets are already divided into 10 folds; we used the first fold as our test set, and the other folds were merged together to form the learning set. In our preliminary experiments, we randomly set aside 10 percent of this learning set to serve as our development set.

Since the focus of our work is on algorithmic enhancements, we simulate the annotator with an oracle and do not address the potential human interface factors. During an experiment, 100 words were drawn at random from the learning set; these constituted the data on which an initial classifier was trained. The rest of the words in the learning set formed the unlabelled pool for active learning; their phonemes were hidden, and a given word's phonemes were revealed if the word was selected for labelling. After training a classifier on the 100 annotated words, we performed 190 iterations of active learning. On each iteration, 10 words were selected according to Equation 1, labelled by an oracle, and added to the training set. In order to speed up the experiments, a random sample of 2000 words was drawn from the pool and presented to the active learner each time. Hence, QBB selected 10 words from the 2000 candidates. We set the QBB committee size $|C|$ to 10.

At each step, we measured word accuracy with respect to the holdout set as the percentage of test words that yielded no erroneous phoneme predictions. Henceforth, we use accuracy to refer to word accuracy. Note that although we query examples using a committee, we train a *single* tree on these examples in order to produce an intelligible model. Prior work has demonstrated that this configuration performs well in practice (Dwyer and Holte, 2007). Our results report the accuracy of the single tree grown on each iteration, averaged

[3] Available at http://pascallin.ecs.soton.ac.uk/Challenges/PRONALSYL/Datasets/

over 10 random draws of the initial training set.

For our decision tree learner, we utilized the J48 algorithm provided by Weka (Witten and Frank, 2005). We also experimented with Wagon (Taylor et al., 1998), an implementation of CART, but J48 performed better during preliminary trials. We ran J48 with default parameter settings, except that binary trees were grown (see Section 2), and subtree raising was disabled.[4]

Our feature template was established during development set experiments with the English CMU data; the data from the other five languages did not influence these choices. The letter context consisted of the focus letter and the 3 letters appearing before and after the focus (or beginning/end of word markers, where applicable). For letter class features, bit strings of length 1 through 6 were used for the focus letter and its immediate neighbors. Bit strings of length at most 3 were used at positions $+2$ and -2, and no such features were added at ± 3.[5] We experimented with other configurations, including using bit strings of up to length 6 at all positions, but they did not produce consistent improvements over the selected scheme.

8 Results

We first examine the contributions of the individual system components, and then compare our complete system to the baseline. The dashed curves in Figure 1 represent the baseline performance with no clustering, no context ordering, random sampling, and ALINE, unless otherwise noted. In all plots, the error bars show the 99% confidence interval for the mean. Because the average word length differs across languages, we report the number of *words* along the x-axis. We have verified that our system does not substantially alter the average number of letters per word in the training set for any of these languages. Hence, the number of words reported here is representative of the true annotation effort.

[4]Subtree raising is an expensive pruning operation that had a negligible impact on accuracy during preliminary experiments. Our pruning performs subtree *replacement* only.

[5]The idea of lowering the specificity of letter class questions as the context length increases is due to Kienappel and Kneser (2001), and is intended to avoid overfitting. However, their configuration differs from ours in that they use longer context lengths (4 for German and 5 for English) and ask letter class questions at every position. Essentially, the authors tuned the feature set in order to optimize performance on each problem, whereas we seek a more general representation that will perform well on a variety of languages.

8.1 Context ordering

Our context ordering strategy improved the accuracy of the decision tree learner on every language (see Figure 1a). Statistically significant improvements were realized on Dutch, French, and German. Our expectation was that context ordering would be particularly helpful during the early rounds of active learning, when there is a greater risk of overfitting on the small training sets. For some languages (notably, German and Spanish) this was indeed the case; yet, for Dutch, context ordering became more effective as the training set increased in size.

It should be noted that our context ordering strategy is sufficiently general that it can be implemented in other decision tree learners that grow binary trees, such as Wagon/CART (Taylor et al., 1998). An n-ary implementation is also feasible, although we have not tried this variation.

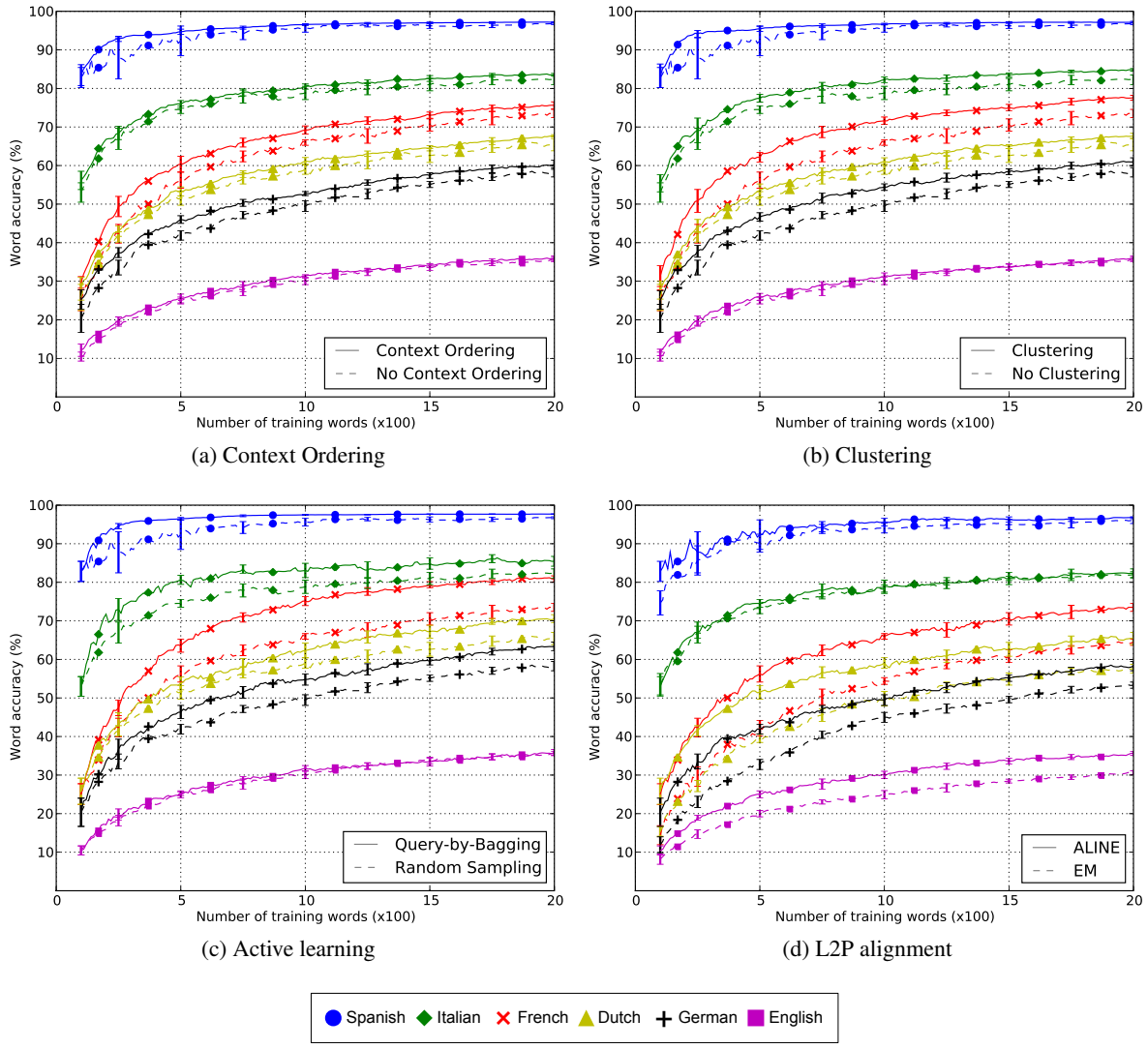
8.2 Clustering letters

As can be seen in Figure 1b, clustering letters into classes tended to produce a steady increase in accuracy. The only case where it had no statistically significant effect was on English. Another benefit of clustering is that it reduces variance. The confidence intervals are generally wider when clustering is disabled, meaning that the system's performance was less sensitive to changes in the initial training set when letter classes were used.

8.3 Active learning

On five of the six datasets, Query-by-Bagging required significantly fewer labelled examples to reach the maximum level of performance achieved by the passive learner (see Figure 1c). For instance, on the Spanish dataset, random sampling reached 97% word accuracy after 1420 words had been annotated, whereas QBB did so with only 510 words — a 64% reduction in labelling effort. Similarly, savings ranging from 30% to 63% were observed for the other languages, with the exception of English, where a statistically insignificant 4% reduction was recorded. Since English is highly irregular in comparison with the other five languages, the active learner tends to query examples that are difficult to classify, but which are unhelpful in terms of generalization.

It is important to note that empirical comparisons of different active learning techniques have shown that random sampling establishes a very



(a) Context Ordering

(b) Clustering

(c) Active learning

(d) L2P alignment

Figure 1: Performance of the individual system components

strong baseline on some datasets (Schein and Ungar, 2007; Settles and Craven, 2008). It is rarely the case that a given active learning strategy is able to unanimously outperform random sampling across a range of datasets. From this perspective, to achieve statistically significant improvements on five of six L2P datasets (without ever being beaten by random) is an excellent result for QBB.

8.4 L2P alignment

The ALINE method for L2P alignment outperformed EM on all six datasets (see Figure 1d). As was mentioned in Section 6, the EM aligner depends on all the available training data, whereas ALINE processes words individually. Only on Spanish and Italian, languages which have highly regular spelling systems, was the EM aligner competitive with ALINE. The accuracy gains on the remaining four datasets are remarkable, considering that better alignments do not necessarily translate into improved classification.

We hypothesized that EM's inferior performance was due to the limited quantities of data that were available in the early stages of active learning. In a follow-up experiment, we allowed EM to align the entire learning set in advance, and these aligned entries were revealed when requested by the learner. We compared this with the usual procedure whereby EM is applied to the labelled training data at each iteration of learning. The learning curves (not shown) were virtually indistinguishable, and there were no statistically significant differences on any of the languages. EM appears to produce poor alignments regardless of the amount of available data.

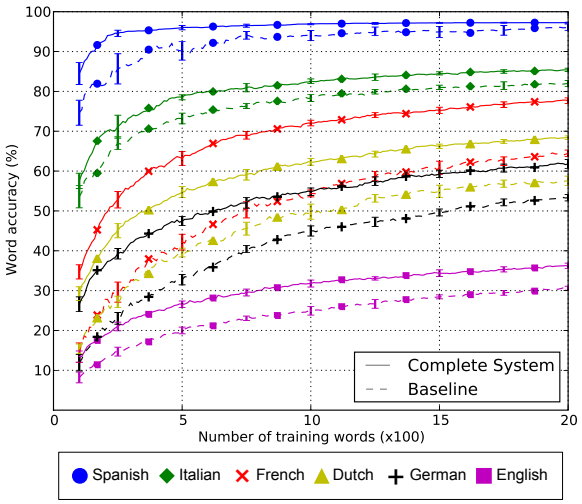


Figure 2: Performance of the complete system

8.5 Complete system

The complete system consists of context ordering, clustering, Query-by-Bagging, and ALINE; the baseline represents random sampling with EM alignment and no additional enhancements. Figure 2 plots the word accuracies for all six datasets.

Although the absolute word accuracies varied considerably across the different languages, our system significantly outperformed the baseline in every instance. On the French dataset, for example, the baseline labelled 1850 words before reaching its maximum accuracy of 64%, whereas the complete system required only 480 queries to reach 64% accuracy. This represents a reduction of 74% in the labelling effort. The savings for the other languages are: Spanish, 75%; Dutch, 68%; English, 59%; German, 59%; and Italian, 52%.[6] Interestingly, the savings are the highest on Spanish, even though the corresponding accuracy gains are the smallest. This demonstrates that our approach is also effective on languages with relatively transparent orthography.

At first glance, the performance of both systems appears to be rather poor on the English dataset. To put our results into perspective, Black et al. (1998) report 57.8% accuracy on this dataset with a similar alignment method and decision tree learner. Our baseline system achieves 57.3% accuracy when 90,000 words have been labelled. Hence, the low values in Figure 2 simply reflect the fact that many more examples are required to learn an accurate classifier for the English data.

[6]The average savings in the number of labelled words with respect to the entire learning curve are similar, ranging from 50% on Italian to 73% on Spanish.

9 Conclusions

We have presented a system for learning a letter-to-phoneme classifier that combines four distinct enhancements in order to minimize the amount of data that must be annotated. Our experiments involving datasets from several languages clearly demonstrate that unlabelled data can be used more efficiently, resulting in greater accuracy for a given training set size, without any additional tuning for the different languages. The experiments also show that a phonetically-based aligner may be preferable to the widely-used EM alignment technique, a discovery that could lead to the improvement of L2P accuracy in general.

While this work represents an important step in reducing the cost of constructing an L2P training set, we intend to explore other active learners and classification algorithms, including sequence labelling strategies (Settles and Craven, 2008). We also plan to incorporate user-centric enhancements (Davel and Barnard, 2004; Culotta and McCallum, 2005) with the aim of reducing both the effort and expertise that is required to annotate words with their phoneme sequences.

Acknowledgments

We would like to thank Sittichai Jiampojamarn for helpful discussions and for providing an implementation of the Expectation-Maximization alignment algorithm. This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Informatics Circle of Research Excellence (iCORE).

References

Naoki Abe and Hiroshi Mamitsuka. 1998. Query learning strategies using boosting and bagging. In *Proc. International Conference on Machine Learning*, pages 1–9.

Ove Andersen, Ronald Kuhn, Ariane Lazaridès, Paul Dalsgaard, Jürgen Haas, and Elmar Nöth. 1996. Comparison of two tree-structured approaches for grapheme-to-phoneme conversion. In *Proc. International Conference on Spoken Language Processing*, volume 3, pages 1700–1703.

R. Harald Baayen, Richard Piepenbrock, and Leon Gulikers, 1996. *The CELEX2 lexical database*. Linguistic Data Consortium, Univ. of Pennsylvania.

Maximilian Bisani and Hermann Ney. 2002. Investigations on joint-multigram models for grapheme-to-phoneme conversion. In *Proc. International Conference on Spoken Language Processing*, pages 105–108.

Alan W. Black, Kevin Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules. In *ESCA Workshop on Speech Synthesis*, pages 77–80.

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Carnegie Mellon University. 1998. The Carnegie Mellon pronouncing dictionary.

David A. Cohn, Les E. Atlas, and Richard E. Ladner. 1992. Improving generalization with active learning. *Machine Learning*, 15(2):201–221.

Alain Content, Phillppe Mousty, and Monique Radeau. 1990. Brulex: Une base de données lexicales informatisée pour le français écrit et parlé. *L'année Psychologique*, 90:551–566.

Piero Cosi, Roberto Gretter, and Fabio Tesser. 2000. Festival parla Italiano. In *Proc. Giornate del Gruppo di Fonetica Sperimentale*.

Aron Culotta and Andrew McCallum. 2004. Confidence estimation for information extraction. In *Proc. HLT-NAACL*, pages 109–114.

Aron Culotta and Andrew McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *Proc. National Conference on Artificial Intelligence*, pages 746–751.

Robert I. Damper, Yannick Marchand, John-David S. Marsters, and Alexander I. Bazin. 2005. Aligning text and phonemes for speech technology applications using an EM-like algorithm. *International Journal of Speech Technology*, 8(2):147–160.

Marelie Davel and Etienne Barnard. 2004. The efficient generation of pronunciation dictionaries: Human factors during bootstrapping. In *Proc. International Conference on Spoken Language Processing*, pages 2797–2800.

Vera Demberg, Helmut Schmid, and Gregor Möhler. 2007. Phonological constraints and morphological preprocessing for grapheme-to-phoneme conversion. In *Proc. ACL*, pages 96–103.

Kenneth Dwyer and Robert Holte. 2007. Decision tree instability and active learning. In *Proc. European Conference on Machine Learning*, pages 128–139.

Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. 1997. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168.

Diana Inkpen, Raphaëlle Martin, and Alain Desrochers. 2007. Graphon: un outil pour la transcription phonétique des mots français. Unpublished manuscript.

International Phonetic Association. 1999. *Handbook of the International Phonetic Association: A Guide to the Use of the International Phonetic Alphabet*. Cambridge University Press.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. In *Proc. ACL*, pages 905–913.

Sittichai Jiampojamarn, Aditya Bhargava, Qing Dou, Kenneth Dwyer, and Grzegorz Kondrak. 2009. DirecTL: a language-independent approach to transliteration. In *Named Entities Workshop (NEWS): Shared Task on Transliteration*. Submitted.

Anne K. Kienappel and Reinhard Kneser. 2001. Designing very compact decision trees for grapheme-to-phoneme transcription. In *Proc. European Conference on Speech Communication and Technology*, pages 1911–1914.

John Kominek and Alan W. Black. 2006. Learning pronunciation dictionaries: Language complexity and word selection strategies. In *Proc. HLT-NAACL*, pages 232–239.

Grzegorz Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *Proc. NAACL*, pages 288–295.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Sameer R. Maskey, Alan W. Black, and Laura M. Tomokiya. 2004. Boostrapping phonetic lexicons for new languages. In *Proc. International Conference on Spoken Language Processing*, pages 69–72.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proc. HLT-NAACL*, pages 337–342.

Andrew I. Schein and Lyle H. Ungar. 2007. Active learning for logistic regression: an evaluation. *Machine Learning*, 68(3):235–265.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proc. Conference on Empirical Methods in Natural Language Processing*, pages 1069–1078.

Paul A. Taylor, Alan Black, and Richard Caley. 1998. The architecture of the Festival Speech Synthesis System. In *ESCA Workshop on Speech Synthesis*, pages 147–151.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition.

Transliteration Alignment

Vladimir Pervouchine, Haizhou Li
Institute for Infocomm Research
A*STAR, Singapore 138632
{vpervouchine,hli}@i2r.a-star.edu.sg

Bo Lin
School of Computer Engineering
NTU, Singapore 639798
linbo@pmail.ntu.edu.sg

Abstract

This paper studies transliteration alignment, its evaluation metrics and applications. We propose a new evaluation metric, *alignment entropy*, grounded on the information theory, to evaluate the alignment quality without the need for the *gold standard* reference and compare the metric with F-score. We study the use of phonological features and affinity statistics for transliteration alignment at phoneme and grapheme levels. The experiments show that better alignment consistently leads to more accurate transliteration. In transliteration modeling application, we achieve a mean reciprocal rate (MRR) of 0.773 on Xinhua personal name corpus, a significant improvement over other reported results on the same corpus. In transliteration validation application, we achieve 4.48% equal error rate on a large LDC corpus.

1 Introduction

Transliteration is a process of rewriting a word from a source language to a target language in a different writing system using the word's phonological equivalent. The word and its transliteration form a *transliteration pair*. Many efforts have been devoted to two areas of studies where there is a need to establish the correspondence between graphemes or phonemes between a transliteration pair, also known as *transliteration alignment*.

One area is the generative transliteration modeling (Knight and Graehl, 1998), which studies how to convert a word from one language to another using statistical models. Since the models are trained on an aligned parallel corpus, the resulting statistical models can only be as good as the alignment of the corpus. Another area is the transliteration validation, which studies the ways to validate transliteration pairs. For example Knight and Graehl (1998) use the lexicon frequency, Qu and Grefenstette (2004) use the statistics in a monolingual corpus and the Web, Kuo et al. (2007) use probabilities estimated from the transliteration model to validate transliteration candidates. In this paper, we propose using the alignment distance between the a bilingual pair of words to establish the evidence of transliteration candidacy. An example of transliteration pair alignment is shown in Figure 1.

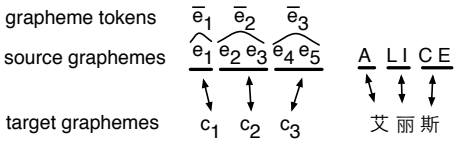


Figure 1: An example of grapheme alignment (Alice, 艾丽斯), where a Chinese grapheme, a character, is aligned to an English grapheme token.

Like the word alignment in statistical machine translation (MT), transliteration alignment becomes one of the important topics in machine transliteration, which has several unique challenges. Firstly, the grapheme sequence in a word is not delimited into grapheme tokens, resulting in an additional level of complexity. Secondly, to maintain the phonological equivalence, the alignment has to make sense at both grapheme and phoneme levels of the source and target languages. This paper reports progress in our ongoing spoken language translation project, where we are interested in the alignment problem of personal name transliteration from English to Chinese.

This paper is organized as follows. In Section 2, we discuss the prior work. In Section 3, we introduce both statistically and phonologically motivated alignment techniques and in Section 4 we advocate an evaluation metric, *alignment entropy* that measures the alignment quality. We report the experiments in Section 5. Finally, we conclude in Section 6.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 136–144,
Suntec, Singapore, 2-7 August 2009.

2 Related Work

A number of transliteration studies have touched on the alignment issue as a part of the transliteration modeling process, where alignment is needed at levels of graphemes and phonemes. In their seminal paper Knight and Graehl (1998) described a transliteration approach that transfers the grapheme representation of a word via the phonetic representation, which is known as phoneme-based transliteration technique (Virga and Khudanpur, 2003; Meng et al., 2001; Jung et al., 2000; Gao et al., 2004). Another technique is to directly transfer the grapheme, known as direct orthographic mapping, that was shown to be simple and effective (Li et al., 2004). Some other approaches that use both source graphemes and phonemes were also reported with good performance (Oh and Choi, 2002; Al-Onaizan and Knight, 2002; Bilac and Tanaka, 2004).

To align a bilingual training corpus, some take a phonological approach, in which the crafted mapping rules encode the prior linguistic knowledge about the source and target languages directly into the system (Wan and Verspoor, 1998; Meng et al., 2001; Jiang et al., 2007; Xu et al., 2006). Others adopt a statistical approach, in which the affinity between phonemes or graphemes is learned from the corpus (Gao et al., 2004; AbdulJaleel and Larkey, 2003; Virga and Khudanpur, 2003).

In the phoneme-based technique where an intermediate level of phonetic representation is used as the pivot, alignment between graphemes and phonemes of the source and target words is needed (Oh and Choi, 2005). If source and target languages have different phoneme sets, alignment between the the different phonemes is also required (Knight and Graehl, 1998). Although the direct orthographic mapping approach advocates a direct transfer of grapheme at run-time, we still need to establish the grapheme correspondence at the model training stage, when phoneme level alignment can help.

It is apparent that the quality of transliteration alignment of a training corpus has a significant impact on the resulting transliteration model and its performance. Although there are many studies of evaluation metrics of word alignment for MT (Lambert, 2008), there has been much less reported work on evaluation metrics of transliteration alignment. In MT, the quality of training corpus alignment $\mathcal{A}$ is often measured relatively to the *gold standard*, or the ground truth alignment $\mathcal{G}$, which is a manual alignment of the corpus or a part of it. Three evaluation metrics are used: *precision*, *recall*, and F*-score*, the latter being a function of the former two. They indicate how close the alignment under investigation is to the gold standard alignment (Mihalcea and Pedersen, 2003). Denoting the number of cross-lingual mappings that are common in both $\mathcal{A}$ and $\mathcal{G}$ as C_{AG}, the number of cross-lingual mappings in $\mathcal{A}$ as C_A and the number of cross-lingual mappings in $\mathcal{G}$ as C_G, precision Pr is given as C_{AG}/C_A, recall Rc as C_{AG}/C_G and F-score as $2Pr \cdot Rc/(Pr + Rc)$.

Note that these metrics hinge on the availability of the gold standard, which is often not available. In this paper we propose a novel evaluation metric for transliteration alignment grounded on the information theory. One important property of this metric is that it does not require a gold standard alignment as a reference. We will also show that how this metric is used in generative transliteration modeling and transliteration validation.

3 Transliteration alignment techniques

We assume in this paper that the source language is English and the target language is Chinese, although the technique is not restricted to English-Chinese alignment.

Let a word in the source language (English) be $\{e_i\} = \{e_1 \dots e_I\}$ and its transliteration in the target language (Chinese) be $\{c_j\} = \{c_1 \dots c_J\}$, $e_i \in E$, $c_j \in C$, and E, C being the English and Chinese sets of characters, or graphemes, respectively. Aligning $\{e_i\}$ and $\{c_j\}$ means for each target grapheme token $\bar{c}_j$ finding a source grapheme token $\bar{e}_m$, which is an English substring in $\{e_i\}$ that corresponds to c_j, as shown in the example in Figure 1. As Chinese is syllabic, we use a Chinese character c_j as the target grapheme token.

3.1 Grapheme affinity alignment

Given a *distance function* between graphemes of the source and target languages $d(e_i, c_j)$, the problem of alignment can be formulated as a dynamic programming problem with the following function to minimize:

$$\begin{aligned} D_{ij} = min(&D_{i-1,j-1} + d(e_i, c_j), \\ &D_{i,j-1} + d(*, c_j), \\ &D_{i-1,j} + d(e_i, *)) \end{aligned} \quad (1)$$

Here the asterisk * denotes a null grapheme that is introduced to facilitate the alignment between graphemes of different lengths. The minimum distance achieved is then given by

$$D = \sum_{i=1}^{I} d(e_i, c_{\theta(i)}) \quad (2)$$

where $j = \theta(i)$ is the correspondence between the source and target graphemes. The alignment can be performed via the Expectation-Maximization (EM) by starting with a random initial alignment and calculating the *affinity matrix* $count(e_i, c_j)$ over the whole parallel corpus, where element (i, j) is the number of times character e_i was aligned to c_j. From the affinity matrix conditional probabilities $P(e_i|c_j)$ can be estimated as

$$P(e_i|c_j) = count(e_i, c_j) / \sum_j count(e_i, c_j) \quad (3)$$

Alignment $j = \theta(i)$ between $\{e_i\}$ and $\{c_j\}$ that maximizes probability

$$P = \prod_i P(c_{\theta(i)}|e_i) \quad (4)$$

is also the same alignment that minimizes alignment distance D:

$$D = -\log P = -\sum_i \log P(c_{\theta(i)}|e_i) \quad (5)$$

In other words, equations (2) and (5) are the same when we have the distance function $d(e_i, c_j) = -\log P(c_j|e_i)$. Minimizing the overall distance over a training corpus, we conduct EM iterations until the convergence is achieved.

This technique solely relies on the affinity statistics derived from training corpus, thus is called grapheme affinity alignment. It is also equally applicable for alignment between a pair of symbol sequences representing either graphemes or phonemes. (Gao et al., 2004; AbdulJaleel and Larkey, 2003; Virga and Khudanpur, 2003).

3.2 Grapheme alignment via phonemes

Transliteration is about finding phonological equivalent. It is therefore a natural choice to use the phonetic representation as the pivot. It is common though that the sound inventory differs from one language to another, resulting in different phonetic representations for source and target words. Continuing with the earlier example,

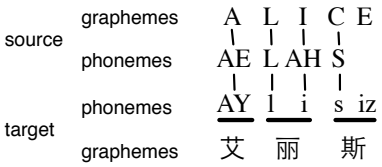


Figure 2: An example of English-Chinese transliteration alignment via phonetic representations.

Figure 2 shows the correspondence between the graphemes and phonemes of English word "Alice" and its Chinese transliteration, with CMU phoneme set used for English (Chase, 1997) and IIR phoneme set for Chinese (Li et al., 2007a).

A Chinese character is often mapped to a unique sequence of Chinese phonemes. Therefore, if we align English characters $\{e_i\}$ and Chinese phonemes $\{cp_k\}$ ($cp_k \in CP$ set of Chinese phonemes) well, we almost succeed in aligning English and Chinese grapheme tokens. Alignment between $\{e_i\}$ and $\{cp_k\}$ becomes the main task in this paper.

3.2.1 Phoneme affinity alignment

Let the phonetic transcription of English word $\{e_i\}$ be $\{ep_n\}$, $ep_n \in EP$, where EP is the set of English phonemes. Alignment between $\{e_i\}$ and $\{ep_n\}$, as well as between $\{ep_n\}$ and $\{cp_k\}$ can be performed via EM as described above. We estimate conditional probability of Chinese phoneme cp_k after observing English character e_i as

$$P(cp_k|e_i) = \sum_{\{ep_n\}} P(cp_k|ep_n) P(ep_n|e_i) \quad (6)$$

We use the distance function between English graphemes and Chinese phonemes $d(e_i, cp_k) = -\log P(cp_k|e_i)$ to perform the initial alignment between $\{e_i\}$ and $\{cp_k\}$ via dynamic programming, followed by the EM iterations until convergence. The estimates for $P(cp_k|ep_n)$ and $P(ep_n|e_i)$ are obtained from the affinity matrices: the former from the alignment of English and Chinese phonetic representations, the latter from the alignment of English words and their phonetic representations.

3.2.2 Phonological alignment

Alignment between the phonetic representations of source and target words can also be achieved using the linguistic knowledge of phonetic similarity. Oh and Choi (2002) define classes of

phonemes and assign various distances between phonemes of different classes. In contrast, we make use of phonological descriptors to define the similarity between phonemes in this paper.

Perhaps the most common way to measure the phonetic similarity is to compute the distances between phoneme features (Kessler, 2005). Such features have been introduced in many ways, such as perceptual attributes or articulatory attributes. Recently, Tao et al. (2006) and Yoon et al. (2007) have studied the use of phonological features and manually assigned phonological distance to measure the similarity of transliterated words for extracting transliterations from a comparable corpus.

We adopt the binary-valued articulatory attributes as the phonological descriptors, which are used to describe the CMU and IIR phoneme sets for English and Chinese Mandarin respectively. Withgott and Chen (1993) define a feature vector of phonological descriptors for English sounds. We extend the idea by defining a 21-element binary feature vector for each English and Chinese phoneme. Each element of the feature vector represents presence or absence of a phonological descriptor that differentiates various kinds of phonemes, e.g. vowels from consonants, front from back vowels, nasals from fricatives, etc[1].

In this way, a phoneme is described by a feature vector. We express the similarity between two phonemes by the Hamming distance, also called the phonological distance, between the two feature vectors. A difference in one descriptor between two phonemes increases their distance by 1. As the descriptors are chosen to differentiate between sounds, the distance between similar phonemes is low, while that between two very different phonemes, such as a vowel and a consonant, is high. The null phoneme, added to both English and Chinese phoneme sets, has a constant distance to any actual phonemes, which is higher than that between any two actual phonemes.

We use the phonological distance to perform the initial alignment between English and Chinese phonetic representations of words. After that we proceed with recalculation of the distances between phonemes using the affinity matrix as described in Section 3.1 and realign the corpus again. We continue the iterations until convergence is reached. Because of the use of phonological descriptors for the initial alignment, we call this technique the *phonological alignment*.

[1]The complete table of English and Chinese phonemes with their descriptors, as well as the transliteration system demo is available at http://translit.i2r.a-star.edu.sg/demos/transliteration/

4 Transliteration alignment entropy

Having aligned the graphemes between two languages, we want to measure how good the alignment is. Aligning the graphemes means aligning the English substrings, called the source grapheme tokens, to Chinese characters, the target grapheme tokens. Intuitively, the more consistent the mapping is, the better the alignment will be. We can quantify the consistency of alignment via *alignment entropy* grounded on information theory.

Given a corpus of aligned transliteration pairs, we calculate $count(c_j, \bar{e}_m)$, the number of times each Chinese grapheme token (character) c_j is mapped to each English grapheme token $\bar{e}_m$. We use the counts to estimate probabilities

$$P(\bar{e}_m, c_j) = count(c_j, \bar{e}_m) / \sum_{m,j} count(c_j, \bar{e}_m)$$

$$P(\bar{e}_m | c_j) = count(c_j, \bar{e}_m) / \sum_{m} count(c_j, \bar{e}_m)$$

The *alignment entropy* of the transliteration corpus is the weighted average of the entropy values for all Chinese tokens:

$$\begin{aligned} H &= -\sum_j P(c_j) \sum_m P(\bar{e}_m|c_j) \log P(\bar{e}_m|c_j) \\ &= -\sum_{m,j} P(\bar{e}_m, c_j) \log P(\bar{e}_m|c_j) \end{aligned} \tag{7}$$

Alignment entropy indicates the uncertainty of mapping between the English and Chinese tokens resulting from alignment. We expect and will show that this estimate is a good indicator of the alignment quality, and is as effective as the F-score, but without the need for a gold standard reference. A lower alignment entropy suggests that each Chinese token tends to be mapped to fewer distinct English tokens, reflecting better consistency. We expect a good alignment to have a sharp cross-lingual mapping with low alignment entropy.

5 Experiments

We use two transliteration corpora: Xinhua corpus (Xinhua News Agency, 1992) of 37,637 personal name pairs and LDC Chinese-English

named entity list LDC2005T34 (Linguistic Data Consortium, 2005), containing 673,390 personal name pairs. The LDC corpus is referred to as LDC05 for short hereafter. For the results to be comparable with other studies, we follow the same splitting of Xinhua corpus as that in (Li et al., 2007b) having a training and testing set of 34,777 and 2,896 names respectively. In contrast to the well edited Xinhua corpus, LDC05 contains erroneous entries. We have manually verified and corrected around 240,000 pairs to clean up the corpus. As a result, we arrive at a set of 560,768 English-Chinese (EC) pairs that follow the Chinese phonetic rules, and a set of 83,403 English-Japanese Kanji (EJ) pairs, which follow the Japanese phonetic rules, and the rest 29,219 pairs (REST) being labeled as incorrect transliterations. Next we conduct three experiments to study 1) alignment entropy vs. F-score, 2) the impact of alignment quality on transliteration accuracy, and 3) how to validate transliteration using alignment metrics.

5.1 Alignment entropy vs. F-score

As mentioned earlier, for English-Chinese grapheme alignment, the main task is to align English graphemes to Chinese phonemes. Phonetic transcription for the English names in Xinhua corpus are obtained by a grapheme-to-phoneme (G2P) converter (Lenzo, 1997), which generates phoneme sequence without providing the exact correspondence between the graphemes and phonemes. G2P converter is trained on the CMU dictionary (Lenzo, 2008).

We align English grapheme and phonetic representations $e - ep$ with the affinity alignment technique (Section 3.1) in 3 iterations. We further align the English and Chinese phonetic representations $ep - cp$ via both affinity and phonological alignment techniques, by carrying out 6 and 7 iterations respectively. The alignment methods are schematically shown in Figure 3.

To study how alignment entropy varies according to different quality of alignment, we would like to have many different alignment results. We pair the intermediate results from the $e - ep$ and $ep - cp$ alignment iterations (see Figure 3) to form $e - ep - cp$ alignments between English graphemes and Chinese phonemes and let them converge through few more iterations, as shown in Figure 4. In this way, we arrive at a total of 114 phonological and 80 affinity alignments of different quality.

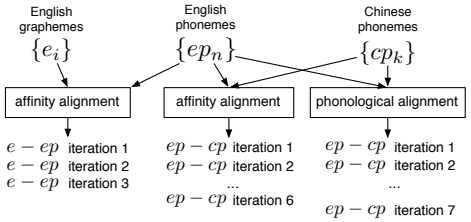


Figure 3: Aligning English graphemes to phonemes $e-ep$ and English phonemes to Chinese phonemes $ep-cp$. Intermediate $e-ep$ and $ep-cp$ alignments are used for producing $e - ep - cp$ alignments.

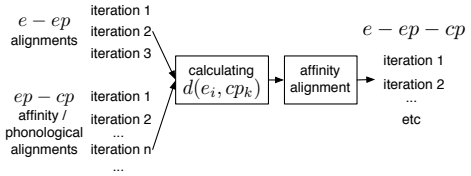
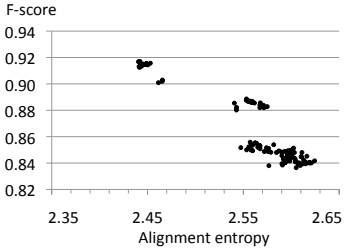


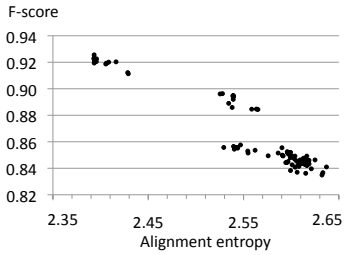
Figure 4: Example of aligning English graphemes to Chinese phonemes. Each combination of $e-ep$ and $ep-cp$ alignments is used to derive the initial distance $d(e_i, cp_k)$, resulting in several $e-ep-cp$ alignments due to the affinity alignment iterations.

We have manually aligned a random set of 3,000 transliteration pairs from the Xinhua training set to serve as the gold standard, on which we calculate the precision, recall and F-score as well as alignment entropy for each alignment. Each alignment is reflected as a data point in Figures 5a and 5b. From the figures, we can observe a clear correlation between the alignment entropy and F-score, that validates the effectiveness of alignment entropy as an evaluation metric. Note that we don't need the gold standard reference for reporting the alignment entropy.

We also notice that the data points seem to form clusters inside which the value of F-score changes insignificantly as the alignment entropy changes. Further investigation reveals that this could be due to the limited number of entries in the gold standard. The 3,000 names in the gold standard are not enough to effectively reflect the change across different alignments. F-score requires a large gold standard which is not always available. In contrast, because the alignment entropy doesn't depend on the gold standard, one can easily report the alignment performance on any unaligned parallel corpus.



(a) 80 affinity alignments



(b) 114 phonological alignments

Figure 5: Correlation between F-score and alignment entropy for Xinhua training set alignments. Results for precision and recall have similar trends .

5.2 Impact of alignment quality on transliteration accuracy

We now further study how the alignment affects the generative transliteration model in the framework of the joint source-channel model (Li et al., 2004). This model performs transliteration by maximizing the joint probability of the source and target names $P(\{e_i\}, \{c_j\})$, where the source and target names are sequences of English and Chinese grapheme tokens. The joint probability is expressed as a chain product of a series of conditional probabilities of token pairs $P(\{e_i\}, \{c_j\}) = P((\bar{e}_k, c_k)|(\bar{e}_{k-1}, c_{k-1}))$, $k = 1 \ldots N$, where we limit the history to one preceding pair, resulting in a bigram model. The conditional probabilities for token pairs are estimated from the aligned training corpus. We use this model because it was shown to be simple yet accurate (Ekbal et al., 2006; Li et al., 2007b). We train a model for each of the 114 phonological alignments and the 80 affinity alignments in Section 5.1 and conduct transliteration experiment on the Xinhua test data.

During transliteration, an input English name is first decoded into a lattice of all possible English and Chinese grapheme token pairs. Then the joint source-channel transliteration model is used to score the lattice to obtain a ranked list of m most likely Chinese transliterations (m-best list).

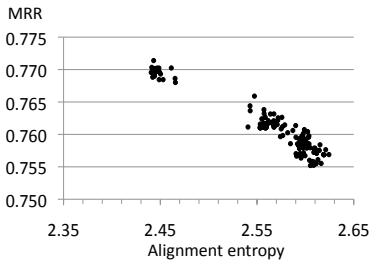
We measure transliteration accuracy as the mean reciprocal rank (MRR) (Kantor and Voorhees, 2000). If there is only one correct Chinese transliteration of the k-th English word and it is found at the r_k-th position in the m-best list, its reciprocal rank is $1/r_k$. If the list contains no correct transliterations, the reciprocal rank is 0. In case of multiple correct transliterations, we take the one that gives the highest reciprocal rank. MRR is the average of the reciprocal ranks across all words in the test set. It is commonly used as a measure of transliteration accuracy, and also allows us to make a direct comparison with other reported work (Li et al., 2007b).

We take $m = 20$ and measure MRR on Xinhua test set for each alignment of Xinhua training set as described in Section 5.1. We report MRR and the alignment entropy in Figures 6a and 7a for the affinity and phonological alignments respectively. The highest MRR we achieve is 0.771 for affinity alignments and 0.773 for phonological alignments. This is a significant improvement over the MRR of 0.708 reported in (Li et al., 2007b) on the same data. We also observe that the phonological alignment technique produces, on average, better alignments than the affinity alignment technique in terms of both the alignment entropy and MRR.

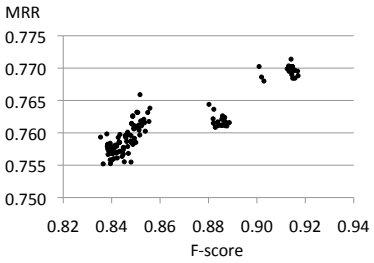
We also report the MRR and F-scores for each alignment in Figures 6b and 7b, from which we observe that alignment entropy has stronger correlation with MRR than F-score does. The Spearman's rank correlation coefficients are -0.89 and -0.88 for data in Figure 6a and 7a respectively. This once again demonstrates the desired property of alignment entropy as an evaluation metric of alignment.

To validate our findings from Xinhua corpus, we further carry out experiments on the EC set of LDC05 containing 560,768 entries. We split the set into 5 almost equal subsets for cross-validation: in each of 5 experiments one subset is used for testing and the remaining ones for training. Since LDC05 contains one-to-many English-Chinese transliteration pairs, we make sure that an English name only appears in one subset.

Note that the EC set of LDC05 contains many names of non-English, and, generally, non-European origin. This makes the G2P converter less accurate, as it is trained on an English phonetic dictionary. We therefore only apply the affinity alignment technique to align the EC set. We

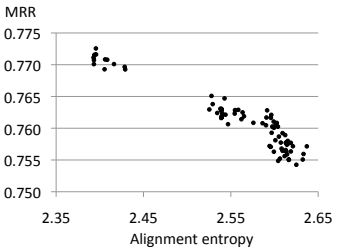


(a) 80 affinity alignments

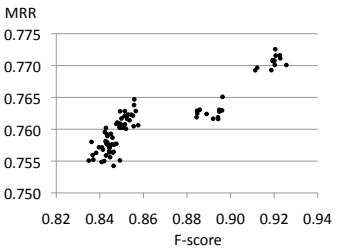


(b) 80 affinity alignments

Figure 6: Mean reciprocal ratio on Xinhua test set vs. alignment entropy and F-score for models trained with different affinity alignments.



(a) 114 phonological alignments



(b) 114 phonological alignments

Figure 7: Mean reciprocal ratio on Xinhua test set vs. alignment entropy and F-score for models trained with different phonological alignments.

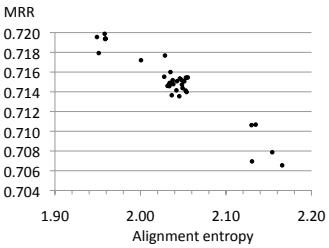


Figure 8: Mean reciprocal ratio vs. alignment entropy for alignments of EC set.

use each iteration of the alignment in the transliteration modeling and present the resulting MRR along with alignment entropy in Figure 8. The MRR results are the averages of five values produced in the five-fold cross-validations.

We observe a clear correlation between the alignment entropy and transliteration accuracy expressed by MRR on LDC05 corpus, similar to that on Xinhua corpus, with the Spearman's rank correlation coefficient of -0.77. We obtain the highest average MRR of 0.720 on the EC set.

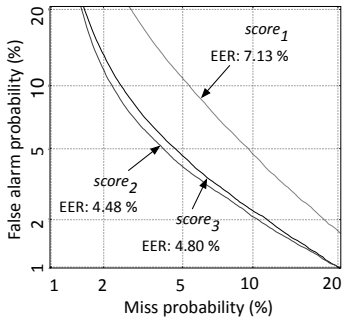
5.3 Validating transliteration using alignment measure

Transliteration validation is a hypothesis test that decides whether a given transliteration pair is genuine or not. Instead of using the lexicon frequency (Knight and Graehl, 1998) or Web statistics (Qu and Grefenstette, 2004), we propose validating transliteration pairs according to the alignment distance D between the aligned English graphemes and Chinese phonemes (see equations (2) and (5)). A distance function $d(e_i, cp_k)$ is established from each alignment on the Xinhua training set as discussed in Section 5.2.

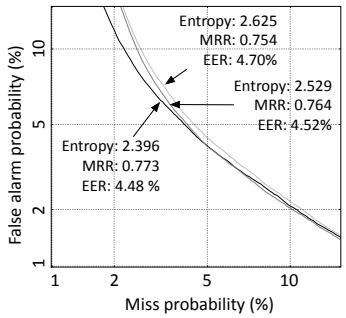
An audit of LDC05 corpus groups the corpus into three sets: an English-Chinese (EC) set of 560,768 samples, an English-Japanese (EJ) set of 83,403 samples and the REST set of 29,219 samples that are not transliteration pairs. We mark the EC name pairs as genuine and the rest 112,622 name pairs that do not follow the Chinese phonetic rules as false transliterations, thus creating the ground truth labels for an English-Chinese transliteration validation experiment. In other words, LDC05 has 560,768 genuine transliteration pairs and 112,622 false ones.

We run one iteration of alignment over LDC05 (both genuine and false) with the distance function $d(e_i, cp_k)$ derived from the affinity matrix of one aligned Xinhua training set. In this way, each transliteration pair in LDC05 provides an alignment distance. One can expect that a genuine transliteration pair typically aligns well, leading to a low distance, while a false transliteration pair will do otherwise. To remove the effect of word length, we normalize the distance by the English name length, the Chinese phonetic transcription

length, and the sum of both, producing $score_1$, $score_2$ and $score_3$ respectively.



(a) DET with $score_1$, $score_2$, $score_3$.



(b) DET results vs. three different alignment quality.

Figure 9: Detection error tradeoff (DET) curves for transliteration validation on LDC05.

We can now classify each LDC05 name pair as genuine or false by having a hypothesis test. When the test score is lower than a pre-set threshold, the name pair is accepted as genuine, otherwise false. In this way, each pre-set threshold will present two types of errors, a false alarm and a miss-detect rate. A common way to present such results is via the detection error tradeoff (DET) curves, which show all possible decision points, and the equal error rate (EER), when false alarm and miss-detect rates are equal.

Figure 9a shows three DET curves based on $score_1$, $score_2$ and $score_3$ respectively for one one alignment solution on the Xinhua training set. The horizontal axis is the probability of miss-detecting a genuine transliteration, while the vertical one is the probability of false-alarms. It is clear that out of the three, $score_2$ gives the best results.

We select the alignments of Xinhua training set that produce the highest and the lowest MRR. We also randomly select three other alignments that produce different MRR values from the pool of 114 phonological and 80 affinity alignments.

Xinhua train set alignment	Alignment entropy of Xinhua train set	MRR on Xinhua test set	LDC classification EER, %
1	2.396	0.773	4.48
2	2.529	0.764	4.52
3	2.586	0.761	4.51
4	2.621	0.757	4.71
5	2.625	0.754	4.70

Table 1: Equal error ratio of LDC transliteration pair validation for different alignments of Xinhua training set.

We use each alignment to derive distance function $d(e_i, cp_k)$. Table 1 shows the EER of LDC05 validation using $score_2$, along with the alignment entropy of the Xinhua training set that derives $d(e_i, cp_k)$, and the MRR on Xinhua test set in the generative transliteration experiment (see Section 5.2) for all 5 alignments. To avoid cluttering Figure 9b, we show the DET curves for alignments 1, 2 and 5 only. We observe that distance function derived from better aligned Xinhua corpus, as measured by both our alignment entropy metric and MRR, leads to a higher validation accuracy consistently on LDC05.

6 Conclusions

We conclude that the alignment entropy is a reliable indicator of the alignment quality, as confirmed by our experiments on both Xinhua and LDC corpora. Alignment entropy does not require the gold standard reference, it thus can be used to evaluate alignments of large transliteration corpora and is possibly to give more reliable estimate of alignment quality than the F-score metric as shown in our transliteration experiment.

The alignment quality of training corpus has a significant impact on the transliteration models. We achieve the highest MRR of 0.773 on Xinhua corpus with phonological alignment technique, which represents a significant performance gain over other reported results. Phonological alignment outperforms affinity alignment on clean database.

We propose using alignment distance to validate transliterations. A high quality alignment on a small verified corpus such as Xinhua can be effectively used to validate a large noisy corpus, such as LDC05. We believe that this property would be useful in transliteration extraction, cross-lingual information retrieval applications.

References

Nasreen AbdulJaleel and Leah S. Larkey. 2003. Statistical transliteration for English-Arabic cross language information retrieval. In *Proc. ACM CIKM*.

Yaser Al-Onaizan and Kevin Knight. 2002. Machine transliteration of names in arabic text. In *Proc. ACL Workshop: Computational Apporaches to Semitic Languages*.

Slaven Bilac and Hozumi Tanaka. 2004. A hybrid back-transliteration system for Japanese. In *Proc. COLING*, pages 597–603.

Lin L. Chase. 1997. *Error-responsive feedback mechanisms for speech recognizers*. Ph.D. thesis, CMU.

Asif Ekbal, Sudip Kumar Naskar, and Sivaji Bandyopadhyay. 2006. A modified joint source-channel model for transliteration. In *Proc. COLING/ACL*, pages 191–198

Wei Gao, Kam-Fai Wong, and Wai Lam. 2004. Phoneme-based transliteration of foreign names for OOV problem. In *Proc. IJCNLP*, pages 374–381.

Long Jiang, Ming Zhou, Lee-Feng Chien, and Cheng Niu. 2007. Named entity translation with web mining and transliteration. In *IJCAI*, pages 1629–1634.

Sung Young Jung, SungLim Hong, and Eunok Paek. 2000. An English to Korean transliteration model of extended Markov window. In *Proc. COLING*, volume 1.

Paul. B. Kantor and Ellen. M. Voorhees. 2000. The TREC-5 confusion track: comparing retrieval methods for scanned text. *Information Retrieval*, 2:165–176.

Brett Kessler. 2005. Phonetic comparison algorithms. *Transactions of the Philological Society*, 103(2):243–260.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4).

Jin-Shea Kuo, Haizhou Li, and Ying-Kuei Yang. 2007. A phonetic similarity model for automatic extraction of transliteration pairs. *ACM Trans. Asian Language Information Processing*, 6(2).

Patrik Lambert. 2008. *Exploiting lexical information and discriminative alignment training in statistical machine translation*. Ph.D. thesis, Universitat Politècnica de Catalunya, Barcelona, Spain.

Kevin Lenzo. 1997. t2p: text-to-phoneme converter builder. http://www.cs.cmu.edu/~lenzo/t2p/.

Kevin Lenzo. 2008. The CMU pronouncing dictionary. http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

Haizhou Li, Min Zhang, and Jian Su. 2004. A joint source-channel model for machine transliteration. In *Proc. ACL*, pages 159–166.

Haizhou Li, Bin Ma, and Chin-Hui Lee. 2007a. A vector space modeling approach to spoken language identification. *IEEE Trans. Acoust., Speech, Signal Process.*, 15(1):271–284.

Haizhou Li, Khe Chai Sim, Jin-Shea Kuo, and Minghui Dong. 2007b. Semantic transliteration of personal names. In *Proc. ACL*, pages 120–127.

Linguistic Data Consortium. 2005. LDC Chinese-English name entity lists LDC2005T34.

Helen M. Meng, Wai-Kit Lo, Berlin Chen, and Karen Tang. 2001. Generate phonetic cognates to handle name entities in English-Chinese cross-language spoken document retrieval. In *Proc. ASRU*.

Rada Mihalcea and Ted Pedersen. 2003. An evaluation exercise for word alignment. In *Proc. HLT-NAACL*, pages 1–10.

Jong-Hoon Oh and Key-Sun Choi. 2002. An English-Korean transliteration model using pronunciation and contextual rules. In *Proc. COLING 2002*.

Jong-Hoon Oh and Key-Sun Choi. 2005. Machine learning based english-to-korean transliteration using grapheme and phoneme information. *IEICE Trans. Information and Systems*, E88-D(7):1737–1748.

Yan Qu and Gregory Grefenstette. 2004. Finding ideographic representations of Japanese names written in Latin script via language identification and corpus validation. In *Proc. ACL*, pages 183–190.

Tao Tao, Su-Youn Yoon, Andrew Fisterd, Richard Sproat, and ChengXiang Zhai. 2006. Unsupervised named entity transliteration using temporal and phonetic correlation. In *Proc. EMNLP*, pages 250–257.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-lingual information retrieval. In *Proc. ACL MLNER*.

Stephen Wan and Cornelia Maria Verspoor. 1998. Automatic English-Chinese name transliteration for development of multilingual resources. In *Proc. COLING*, pages 1352–1356.

M. M. Withgott and F. R. Chen. 1993. *Computational models of American speech*. Centre for the study of language and information.

Xinhua News Agency. 1992. *Chinese transliteration of foreign personal names*. The Commercial Press.

LiLi Xu, Atsushi Fujii, and Tetsuya Ishikawa. 2006. Modeling impression in probabilistic transliteration into Chinese. In *Proc. EMNLP*, pages 242–249.

Su-Youn Yoon, Kyoung-Young Kim, and Richard Sproat. 2007. Multilingual transliteration using feature based phonetic method. In *Proc. ACL*, pages 112–119.

Automatic training of lemmatization rules that handle morphological changes in pre-, in- and suffixes alike

Bart Jongejan
CST-University of Copenhagen
Njalsgade 140-142 2300 København S
Denmark
bartj@hum.ku.dk

Hercules Dalianis† ‡
†DSV, KTH - Stockholm University
Forum 100, 164 40 Kista, Sweden
‡Euroling AB, SiteSeeker
Igeldammsgatan 22c
112 49 Stockholm, Sweden
hercules@dsv.su.se

Abstract

We propose a method to automatically train lemmatization rules that handle prefix, infix and suffix changes to generate the lemma from the full form of a word. We explain how the lemmatization rules are created and how the lemmatizer works. We trained this lemmatizer on Danish, Dutch, English, German, Greek, Icelandic, Norwegian, Polish, Slovene and Swedish full form-lemma pairs respectively. We obtained significant improvements of 24 percent for Polish, 2.3 percent for Dutch, 1.5 percent for English, 1.2 percent for German and 1.0 percent for Swedish compared to plain suffix lemmatization using a suffix-only lemmatizer. Icelandic deteriorated with 1.9 percent. We also made an observation regarding the number of produced lemmatization rules as a function of the number of training pairs.

1 Introduction

Lemmatizers and stemmers are valuable human language technology tools to improve precision and recall in an information retrieval setting. For example, stemming and lemmatization make it possible to match a query in one morphological form with a word in a document in another morphological form. Lemmatizers can also be used in lexicography to find new words in text material, including the words' frequency of use. Other applications are creation of index lists for book indexes as well as key word lists

Lemmatization is the process of reducing a word to its base form, normally the dictionary look-up form (lemma) of the word. A trivial way to do this is by dictionary look-up. More advanced systems use hand crafted or automatically generated transformation rules that look at the surface form of the word and attempt to produce the correct base form by replacing all or parts of the word.

Stemming conflates a word to its stem. A stem does not have to be the lemma of the word, but can be any trait that is shared between a group of words, so that even the group membership itself can be regarded as the group's stem.

The most famous stemmer is the Porter Stemmer for English (Porter 1980). This stemmer removes around 60 different suffixes, using rewriting rules in two steps.

The paper is structured as follows: section 2 discusses related work, section 3 explains what the new algorithm is supposed to do, section 4 describes some details of the new algorithm, section 5 evaluates the results, conclusions are drawn in section 6, and finally in section 7 we mention plans for further tests and improvements.

2 Related work

There have been some attempts in creating stemmers or lemmatizers automatically. Ekmekçioglu *et al*. (1996) have used *N*-gram matching for Turkish that gave slightly better results than regular rule based stemming. Theron and Cloete (1997) learned two-level rules for English, Xhosa and Afrikaans, but only single character insertions, replacements and additions were allowed. Oard *et al*. (2001) used a language independent stemming technique in a dictionary based cross language information retrieval experiment for German, French and Italian where English was the search language. A four stage backoff strategy for improving recall was intro-

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 145–153,
Suntec, Singapore, 2-7 August 2009.

duced. The system worked fine for French but not so well for Italian and German. Majumder *et al*. (2007) describe a statistical stemmer, YASS (Yet Another Suffix Stripper), mainly for Bengali and French, but they propose it also for Hindi and Gujarati. The method finds clusters of similar words in a corpus. The clusters are called stems. The method works best for languages that are basically suffix based. For Bengali precision was 39.3 percent better than without stemming, though no absolute numbers were reported for precision. The system was trained on a corpus containing 301 562 words.

Kanis & Müller (2005) used an automatic technique called OOV Words Lemmatization to train their lemmatizer on Czech, Finnish and English data. Their algorithm uses two pattern tables to handle suffixes as well as prefixes. Plisson *et al*. (2004) presented results for a system using Ripple Down Rules (RDR) to generate lemmatization rules for Slovene, achieving up to 77 percent accuracy. Matjaž *et al*. (2007) present an RDR system producing efficient suffix based lemmatizers for 14 languages, three of which (English, German and Slovene) our algorithm also has been tested with.

Stempel (Białecki 2004) is a stemmer for Polish that is trained on Polish full form – lemma pairs. When tested with inflected out-of-vocabulary (OOV) words Stempel produces 95.4 percent correct stems, of which about 81 percent also happen to be correct lemmas.

Hedlund (2001) used two different approaches to automatically find stemming rules from a corpus, for both Swedish and English. Unfortunately neither of these approaches did beat the hand crafted rules in the Porter stemmer for English (Porter 1980) or the Euroling SiteSeeker stemmer for Swedish, (Carlberger *et al*. 2001).

Jongejan & Haltrup (2005) constructed a trainable lemmatizer for the lexicographical task of finding lemmas outside the existing dictionary, bootstrapping from a training set of full form – lemma pairs extracted from the existing dictionary. This lemmatizer looks only at the suffix part of the word. Its performance was compared with a stemmer using hand crafted stemming rules, the Euroling SiteSeeker stemmer for Swedish, Danish and Norwegian, and also with a stemmer for Greek, (Dalianis & Jongejan 2006). The results showed that lemmatizer was as good as the stemmer for Swedish, slightly better for Danish and Norwegian but worse for Greek. These results are very dependent on the quality (errors, size) and complexity (diacritics, capitals) of the training data.

In the current work we have used Jongejan & Haltrup's lemmatizer as a reference, referring to it as the 'suffix lemmatizer'.

3 Delineation

3.1 Why affix rules?

German and Dutch need more advanced methods than suffix replacement since their affixing of words (inflection of words) can include both prefixing, infixing and suffixing. Therefore we created a trainable lemmatizer that handles pre- and infixes in addition to suffixes.

Here is an example to get a quick idea of what we wanted to achieve with the new training algorithm. Suppose we have the following Dutch full form – lemma pair:

`afgevraagd → afvragen`

(Translation: `wondered, to wonder`)

If this were the sole input given to the training program, it should produce a transformation rule like this:

`*ge*a*d → ***en`

The asterisks are wildcards and placeholders. The pattern on the left hand side contains three wildcards, each one corresponding to one placeholder in the replacement string on the right hand side, in the same order. The characters matched by a wildcard are inserted in the place kept free by the corresponding placeholder in the replacement expression.

With this "set" of rules a lemmatizer would be able to construct the correct lemma for some words that had not been used during the training, such as the word `verstekgezaagd` (Translation: `mitre cut`):

Word	**verstek**	**ge**	**z**	**a**	**ag**	**d**
Pattern	*	ge	*	a	*	d
Replacement	*		*		*	en
Lemma	**verstek**		**z**		**ag**	**en**

Table 1. Application of a rule to an OOV word.

For most words, however, the lemmatizer would simply fail to produce any output, because not all words do contain the literal strings `ge` and `a` and a final `d`. We remedy this by adding a one-size-fits-all rule that says "return the input as output":

`* → *`

So now our rule set consists of two rules:

```
*ge*a*d → ***en
* → *
```

The lemmatizer then finds the rule with the most specific pattern (see 4.2) that matches and applies only this rule. The last rule's pattern matches any word and so the lemmatizer cannot fail to produce output. Thus, in our toy rule set consisting of two rules, the first rule handles words like `gevraagd`, `afgezaagd`, `geklaagd`, (all three correctly) and `getalmd` (incorrectly) while the second rule handles words like `directeur` (correctly) and `zei` (incorrectly).

3.2 Inflected *vs.* agglutinated languages

A lemmatizer that only applies one rule per word is useful for inflected languages, a class of languages that includes all Indo-European languages. For these languages morphological change is not a productive process, which means that no word can be morphologically changed in an unlimited number of ways. Ideally, there are only a finite number of inflection schemes and thus a finite number of lemmatization rules should suffice to lemmatize indefinitely many words.

In agglutinated languages, on the other hand, there are classes of words that in principle have innumerous word forms. One way to lemmatize such words is to peel off all agglutinated morphemes one by one. This is an iterative process and therefore the lemmatizer discussed in this paper, which applies only one rule per word, is not an obvious choice for agglutinated languages.

3.3 Supervised training

An automatic process to create lemmatization rules is described in the following sections. By reserving a small part of the available training data for testing it is possible to quite accurately estimate the probability that the lemmatizer would produce the right lemma given any unknown word belonging to the language, even without requiring that the user masters the language (Kohavi 1995).

On the downside, letting a program construct lemmatization rules requires an extended list of full form – lemma pairs that the program can exercise on – at least tens of thousands and possibly over a million entries (Dalianis and Jongejan 2006).

3.4 Criteria for success

The main challenge for the training algorithm is that it must produce rules that accurately lemmatize OOV words. This requirement translates to two opposing tendencies during training. On the one hand we must trust rules with a wide basis of training examples more than rules with a small basis, which favours rules with patterns that fit many words. On the other hand we have the incompatible preference for cautious rules with rather specific patterns, because these must be better at avoiding erroneous rule applications than rules with generous patterns. The envisaged expressiveness of the lemmatization rules – allowing all kinds of affixes and an unlimited number of wildcards – turns the challenge into a difficult balancing act.

In the current work we wanted to get an idea of the advantages of an affix-based algorithm compared to a suffix-only based algorithm. Therefore we have made the task as hard as possible by not allowing language specific adaptations to the algorithms and by not subdividing the training words in word classes.

4 Generation of rules and look-up data structure

4.1 Building a rule set from training pairs

The training algorithm generates a data structure consisting of rules that a lemmatizer must traverse to arrive at a rule that is elected to fire.

Conceptually the training process is as follows. As the data structure is being built, the full form in each training pair is tentatively lemmatized using the data structure that has been created up to that stage. If the elected rule produces the right lemma from the full form, nothing needs to be done. Otherwise, the data structure must be expanded with a rule such that the new rule *a)* is elected instead of the erroneous rule and *b)* produces the right lemma from the full form. The training process terminates when the full forms in all pairs in the training set are transformed to their corresponding lemmas.

After training, the data structure of rules is made permanent and can be consulted by a lemmatizer. The lemmatizer must elect and fire rules in the same way as the training algorithm, so that all words from the training set are lemmatized correctly. It may however fail to produce the correct lemmas for words that were not in the training set – the OOV words.

4.2 Internal structure of rules: prime and derived rules

During training the Ratcliff/Obershelp algorithm (Ratcliff & Metzener 1988) is used to find the longest non-overlapping similar parts in a given full form – lemma pair. For example, in the pair

`afgevraagd → afvragen`

the longest common substring is `vra`, followed by `af` and `g`. These similar parts are replaced with wildcards and placeholders:

`*ge*a*d → ***en`

Now we have the prime rule for the training pair, the least specific rule necessary to lemmatize the word correctly. Rules with more specific patterns – derived rules – can be created by adding characters and by removing or adding wildcards. A rule that is derived from another rule (derived or prime) is more specific than the original rule: Any word that is successfully matched by the pattern of a derived rule is also successfully matched by the pattern of the original rule, but the converse is not the case. This establishes a partial ordering of all rules. See Figures 1 and 2, where the rules marked 'p' are prime rules and those marked 'd' are derived.

Innumerous rules can be derived from a rule with at least one wildcard in its pattern, but only a limited number can be tested in a finite time. To keep the number of candidate rules within practical limits, we used the strategy that the pattern of a candidate is minimally different from its parent's pattern: it can have one extra literal character or one wildcard less or replace one wildcard with one literal character. Alternatively, a candidate rule (such as the bottom rule in Figure 4) can arise by merging two rules. Within these constraints, the algorithm creates all possible candidate rules that transform one or more training words to their corresponding lemmas.

4.3 External structure of rules: partial ordering in a DAG and in a tree

We tried two different data structures to store new lemmatizer rules, a directed acyclic graph (DAG) and a plain tree structure with depth first, left to right traversal.

The DAG (Figure 1) expresses the complete partial ordering of the rules. There is no preferential order between the children of a rule and all paths away from the root must be regarded as equally valid. Therefore the DAG may lead to several lemmas for the same input word. For example, without the rule in the bottom part of Figure 1, the word `gelopen` would have been lemmatized to both `lopen` (correct) and `gelopen` (incorrect):

```
gelopen:
*ge* → **          lopen
*pen → *pen        gelopen
```

By adding a derived rule as a descendent of both these two rules, we make sure that lemmatization of the word `gelopen` is only handled by one rule and only results in the correct lemma:

```
gelopen:
*ge*pen → **pen    lopen
```

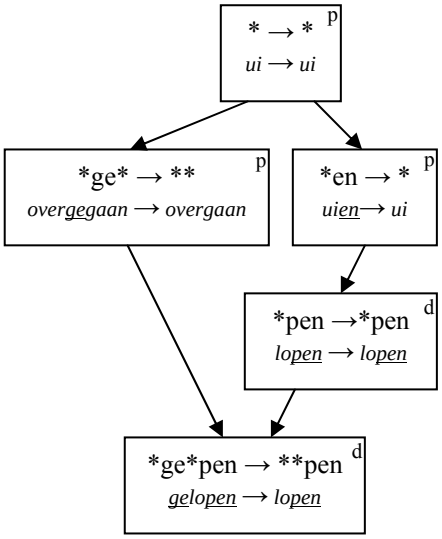


Figure 1. Five training pairs as supporters for five rules in a DAG.

The tree in Figure 2 is a simpler data structure and introduces a left to right preferential order between the children of a rule. Only one rule fires and only one lemma per word is produced. For example, because the rule `*ge* → **` precedes its sibling rule `*en → *`, whenever the former rule is applicable, the latter rule and its descendents are not even visited, irrespective of their applicability. In our example, the former rule – and only the former rule – handles the lemmatization of `gelopen`, and since it produces the correct lemma an additional rule is not necessary.

In contrast to the DAG, the tree implements negation: if the N^{th} sibling of a row of children fires, it not only means that the pattern of the N^{th} rule matches the word, it also means that the patterns of the N-1 preceding siblings do not match the word. Such implicit negation is not possible in the DAG, and this is probably the main reason why the experiments with the DAG-structure lead to huge numbers of rules, very little gener-

alization, uncontrollable training times (months, not minutes!) and very low lemmatization quality. On the other hand, the experiments with the tree structure were very successful. The building time of the rules is acceptable, taking small recursive steps during the training part. The memory use is tractable and the quality of the results is good provided good training material.

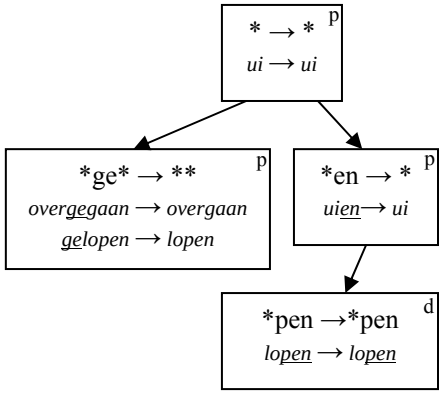


Figure 2. The same five training pairs as supporters for only four rules in a tree.

4.4 Rule selection criteria

This section pertains to the training algorithm employing a tree.

The typical situation during training is that a rule that already has been added to the tree makes lemmatization errors on some of the training words. In that case one or more corrective children have to be added to the rule[1].

If the pattern of a new child rule only matches some, but not all training words that are lemmatized incorrectly by the parent, a right sibling rule must be added. This is repeated until all training words that the parent does not lemmatize correctly are matched by the leftmost child rule or one of its siblings.

A candidate child rule is faced with training words that the parent did not lemmatize correctly and, surprisingly, also supporters of the parent, because the pattern of the candidate cannot discriminate between these two groups.

On the output side of the candidate appear the training pairs that are lemmatized correctly by the candidate, those that are lemmatized incorrectly and those that do not match the pattern of the candidate.

For each candidate rule the training algorithm creates a 2×3 table (see Table 2) that counts the number of training pairs that the candidate lemmatizes correctly or incorrectly or that the candidate does not match. The two columns count the training pairs that, respectively, were lemmatized incorrectly and correctly by the parent. These six parameters N_{xy} can be used to select the best candidate. Only four parameters are independent, because the numbers of training words that the parent lemmatized incorrectly (N_w) and correctly (N_r) are the same for all candidates. Thus, after the application of the first and most significant selection criterion, up to three more selection criteria of decreasing significance can be applied if the preceding selection ends in a tie.

Parent / Child	Incorrect	Correct (supporters)
Correct	$\boldsymbol{N_{wr}}$	N_{rr}
Incorrect	N_{ww}	$\boldsymbol{N_{rw}}$
Not matched	N_{wn}	N_{rn}
Sum	N_w	N_r

Table 2. The six parameters for rule selection among candidate rules.

A large N_{wr} and a small N_{rw} are desirable. N_{wr} is a measure for the rate at which the updated data structure has learned to correctly lemmatize those words that previously were lemmatized incorrectly. A small N_{rw} indicates that only few words that previously were lemmatized correctly are spoiled by the addition of the new rule. It is less obvious how the other numbers weigh in.

We have obtained the most success with criteria that first select for highest $N_{wr} + N_{rr} - N_{rw}$. If the competition ends in a tie, we select for lowest N_{rr} among the remaining candidates. If the competition again ends in a tie, we select for highest $N_{rn} - N_{ww}$. Due to the marginal effect of a fourth criterion we let the algorithm randomly select one of the remaining candidates instead.

The training pairs that are matched by the pattern of the winning rule become the supporters and non-supporters of that new rule and are no longer supporters or non-supporters of the parent. If the parent still has at least one non-supporter, the remaining supporters and non-supporters – the training pairs that the winning

[1] If the case of a DAG, care must be taken that the complete representation of the partial ordering of rules is maintained. Any new rule not only becomes a child of the rule that it was aimed at as a corrective child, but often also of several other rules.

candidate does not match – are used to select the right sibling of the new rule.

5 Evaluation

We trained the new lemmatizer using training material for Danish (STO), Dutch (CELEX), English (CELEX), German (CELEX), Greek (Petasis *et al.* 2003), Icelandic (IFD), Norwegian (SCARRIE), Polish (Morfologik), Slovene (Juršič *et al*. 2007) and Swedish (SUC).

The guidelines for the construction of the training material are not always known to us. In some cases, we know that the full forms have been generated automatically from the lemmas. On the other hand, we know that the Icelandic data is derived from a corpus and only contains word forms occurring in that corpus. Because of the uncertainties, the results cannot be used for a quantitative comparison of the accuracy of lemmatization between languages.

Some of the resources were already disambiguated (one lemma per full form) when we received the data. We decided to disambiguate the remaining resources as well. Handling homographs wisely is important in many lemmatization tasks, but there are many pitfalls. As we only wanted to investigate the improvement of the affix algorithm over the suffix algorithm, we decided to factor out ambiguity. We simply chose the lemma that comes first alphabetically and discarded the other lemmas from the available data.

The evaluation was carried out by dividing the available material in training data and test data in seven different ratios, setting aside between 1.54% and 98.56% as training data and the remainder as OOV test data. (See section 7). To keep the sample standard deviation *s* for the accuracy below an acceptable level we used the evaluation method *repeated random subsampling validation* that is proposed in Voorhees (2000) and Bouckaert & Frank (2000). We repeated the training and evaluation for each ratio with several randomly chosen sets, up to 17 times for the smallest and largest ratios, because these ratios lead to relatively small training sets and test sets respectively. The same procedure was followed for the suffix lemmatizer, using the same training and test sets. Table 3 shows the results for the largest training sets.

For some languages lemmatization accuracy for OOV words improved by deleting rules that are based on very few examples from the training data. This pruning was done after the training of the rule set was completed. Regarding the affix algorithm, the results for half of the languages became better with mild pruning, i.e. deleting rules with only one example. For Danish, Dutch, German, Greek and Icelandic pruning did not improve accuracy. Regarding the suffix algorithm, only English and Swedish profited from pruning.

Language	Suffix %	Affix %	Δ %	$N \times 1000$	*n*
Icelandic	73.2±1.4	71.3±1.5	-1.9	58	17
Danish	93.2±0.4	92.8±0.2	-0.4	553	5
Norwegian	87.8±0.4	87.6±0.3	-0.2	479	6
Greek	90.2±0.3	90.4±0.4	0.2	549	5
Slovene	86.0±0.6	86.7±0.3	0.7	199	9
Swedish	91.24±0.18	92.3±0.3	1.0	478	6
German	90.3±0.5	91.46±0.17	1.2	315	7
English	87.5±0.9	89.0±1.3	1.5	76	15
Dutch	88.2±0.5	90.4±0.5	2.3	302	7
Polish	69.69±0.06	93.88±0.08	24.2	3443	2

Table 3. Accuracy for the suffix and affix algorithms. The fifth column shows the size of the available data. Of these, 98.56% was used for training and 1.44% for testing. The last column shows the number *n* of performed iterations, which was inversely proportional to $\sqrt{N}$ with a minimum of two.

6 Some language specific notes

For Polish, the suffix algorithm suffers from overtraining. The accuracy tops at about 100 000 rules, which is reached when the training set comprises about 1 000 000 pairs.

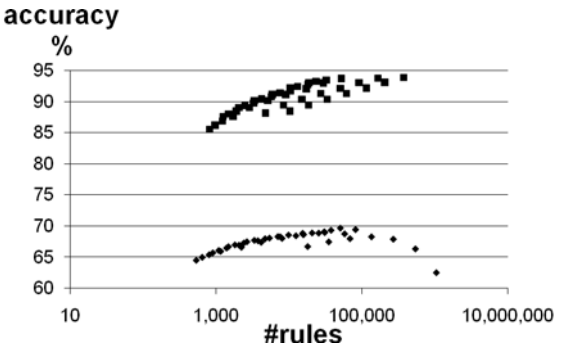


Figure 3. Accuracy vs. number of rules for Polish
Upper swarm of data points: affix algorithm.
Lower swarm of data points: suffix algorithm.
Each swarm combines results from six rule sets with varying amounts of pruning (no pruning and pruning with cut-off = 1..5).

If more training pairs are added, the number of rules grows, but the accuracy falls. The affix algorithm shows no sign of overtraining, even

though the Polish material comprised 3.4 million training pairs, more than six times the number of the second language on the list, Danish. See Figure 3.

The improvement of the accuracy for Polish was tremendous. The inflectional paradigm in Polish (as in other Slavic languages) can be left factorized, except for the superlative. However, only 3.8% of the words in the used Polish data have the superlative forming prefix `naj`, and moreover this prefix is only removed from adverbs and not from the much more numerous adjectives.

The true culprit of the discrepancy is the great number (> 23%) of words in the Polish data that have the negative prefix `nie`, which very often does not recur in the lemma. The suffix algorithm cannot handle these 23% correctly.

The improvement over the suffix lemmatizer for the case of German is unassuming. To find out why, we looked at how often rules with infix or prefix patterns fire and how well they are doing. We trained the suffix algorithm with 9/10 of the available data and tested with the remaining 1/10, about 30 000 words. Of these, 88% were lemmatized correctly (a number that indicates the smaller training set than in Table 3).

	German		Dutch	
	Acc. %	Freq %	Acc. %	Freq %
all	88.1	100.0	87.7	100.0
suffix-only	88.7	94.0	88.1	94.9
prefix	79.9	4.4	80.9	2.4
infix	83.3	2.3	77.4	3.0
ä ö ü	92.8	0.26	N/A	0.0
ge infix	68.6	0.94	77.9	2.6

Table 4. Prevalence of suffix-only rules, rules specifying a prefix, rules specifying an infix and rules specifying infixes containing either *ä*, *ö* or *ü* or the letter combination *ge*.

Almost 94% of the lemmas were created using suffix-only rules, with an accuracy of almost 89%. Less than 3% of the lemmas were created using rules that included at least one infix subpattern. Of these, about 83% were correctly lemmatized, pulling the average down. We also looked at two particular groups of infix-rules: those including the letters `ä`, `ö` or `ü` and those with the letter combination `ge`. The former group applies to many words that display umlaut, while the latter applies to past participles. The first group of rules, accounting for 11% of all words handled by infix rules, performed better than average, about 93%, while the latter group, accounting for 40% of all words handled by infix rules, performed poorly at 69% correct lemmas. Table 4 summarizes the results for German and the closely related Dutch language.

7 Self-organized criticality

Over the whole range of training set sizes the number of rules goes like $C.N^d$ with $0 < C$, and N the number of training pairs. The value of C and d not only depended on the chosen algorithm, but also on the language. Figure 4 shows how the number of generated lemmatization rules for Polish grows as a function of the number of training pairs.

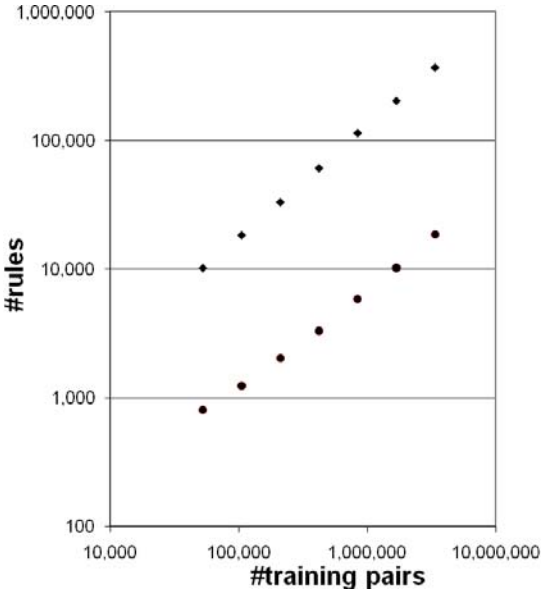


Figure 4. Number of rules vs. number of training pairs for Polish (double logarithmic scale).
Upper row: unpruned rule sets
Lower row: heavily pruned rule sets (cut-off=5)

There are two rows of data, each row containing seven data points. The rules are counted after training with 1.54 percent of the available data and then repeatedly doubling to 3.08, 6.16, 12.32, 24.64, 49.28 and 98.56 percent of the available data. The data points in the upper row designate the number of rules resulting from the training process. The data points in the lower row arise by pruning rules that are based on less than six examples from the training set.

The power law for the upper row of data points for Polish in Figure 4 is

$$N_{rules} = 0.80 N_{training}^{0.87}$$

As a comparison, for Icelandic the power law for the unpruned set of rules is

$$N_{rules} = 1.32 N_{training}^{0.90}$$

These power law expressions are derived for the affix algorithm. For the suffix algorithm the exponent in the Polish power law expression is very close to 1 (0.98), which indicates that the suffix lemmatizer is not good at all at generalizing over the Polish training data: the number of rules grows almost proportionally with the number of training words. (And, as Figure 3 shows, to no avail.) On the other hand, the suffix lemmatizer fares better than the affix algorithm for Icelandic data, because in that case the exponent in the power law expression is lower: 0.88 versus 0.90.

The power law is explained by self-organized criticality (Bak *et al.* 1987, 1988). Rule sets that originate from training sets that only differ in a single training example can be dissimilar to any degree depending on whether and where the difference is tipping the balance between competing rule candidates. Whether one or the other rule candidate wins has a very significant effect on the parts of the tree that emanate as children or as siblings from the winning node. If the difference has an effect close to the root of the tree, a large expanse of the tree is affected. If the difference plays a role closer to a leaf node, only a small patch of the tree is affected. The effect of adding a single training example can be compared with dropping a single rice corn on top of a pile of rice, which can create an avalanche of unpredictable size.

8 Conclusions

Affix rules perform better than suffix rules if the language has a heavy pre- and infix morphology and the size of the training data is big. The new algorithm worked very well with the Polish Morfologik dataset and compares well with the Stempel algorithm (Białecki 2008).

Regarding Dutch and German we have observed that the affix algorithm most often applies suffix-only rules to OOV words. We have also observed that words lemmatized this way are lemmatized better than average. The remaining words often need morphological changes in more than one position, for example both in an infix and a suffix. Although these changes are correlated by the inflectional rules of the language, the number of combinations is still large, while at the same time the number of training examples exhibiting such combinations is relatively small. Therefore the more complex rules involving infix or prefix subpatterns or combinations thereof are less well-founded than the simple suffix-only rules. The lemmatization accuracy of the complex rules will therefore in general be lower than that of the suffix-only rules. The reason why the affix algorithm is still better than the algorithm that only considers suffix rules is that the affix algorithm only generates suffix-only rules from words with suffix-only morphology. The suffix-only algorithm is not able to generalize over training examples that do not fulfil this condition and generates many rules based on very few examples. Consequently, everything else being equal, the set of suffix-only rules generated by the affix algorithm must be of higher quality than the set of rules generated by the suffix algorithm.

The new affix algorithm has fewer rules supported by only one example from the training data than the suffix algorithm. This means that the new algorithm is good at generalizing over small groups of words with exceptional morphology. On the other hand, the bulk of 'normal' training words must be bigger for the new affix based lemmatizer than for the suffix lemmatizer. This is because the new algorithm generates immense numbers of candidate rules with only marginal differences in accuracy, requiring many examples to find the best candidate.

When we began experimenting with lemmatization rules with unrestricted numbers of affixes, we could not know whether the limited amount of available training data would be sufficient to fix the enormous amount of free variables with enough certainty to obtain higher quality results than obtainable with automatically trained lemmatizers allowing only suffix transformations.

However, the results that we have obtained with the new affix algorithm are on a par with or better than those of the suffix lemmatizer. There is still room for improvements as only part of the parameter space of the new algorithm has been searched. The case of Polish shows the superiority of the new algorithm, whereas the poor results for Icelandic, a suffix inflecting language with many inflection types, were foreseeable, because we only had a small training set.

9 Future work

Work with the new affix lemmatizer has until now focused on the algorithm. To really know if the carried out theoretical work is valuable we would like to try it out in a real search setting in a search engine and see if the users appreciate the new algorithm's results.

References

Per Bak, Chao Tang and Kurt Wiesenfeld. 1987. Self-Organized Criticality: An Explanation of 1/f Noise, *Phys. Rev. Lett.*, vol. 59,. pp. 381-384, 1987

Per Bak, Chao Tang and Kurt Wiesenfeld . 1988. *Phys. Rev.* A38, (1988), pp. 364-374

Andrzej Białecki, 2004, Stempel - Algorithmic Stemmer for Polish Language http://www.getopt.org/stempel/

Remco R. Bouckaert and Eibe Frank. 2000. Evaluating the Replicability of Significance Tests for Comparing Learning Algorithms. In H. Dai, R. Srikant, & C. Zhang (Eds.), Proc. 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia, May 26-28, 2004 (pp. 3-12). Berlin: Springer.

Johan Carlberger, Hercules Dalianis, Martin Hassel, and Ola Knutsson. 2001. Improving Precision in Information Retrieval for Swedish using Stemming. In the *Proceedings of NoDaLiDa-01 - 13th Nordic Conference on Computational Linguistics*, May 21-22, Uppsala, Sweden.

Celex: http://celex.mpi.nl/

Hercules Dalianis and Bart Jongejan 2006. Handcrafted versus Machine-learned Inflectional Rules: the Euroling-SiteSeeker Stemmer and CST's Lemmatiser, in *Proceedings of the International Conference on Language Resources and Evaluation*, LREC 2006.

F. Çuna Ekmekçioglu, Mikael F. Lynch, and Peter Willett. 1996. Stemming and N-gram matching for term conflation in Turkish texts. *Information Research*, 7(1) pp 2-6.

Niklas Hedlund 2001. *Automatic construction of stemming rules*, Master Thesis, NADA-KTH, Stockholm, TRITA-NA-E0194.

IFD: Icelandic Centre for Language Technology, http://tungutaekni.is/researchsystems/rannsoknir_12en.html

Bart Jongejan and Dorte Haltrup. 2005. *The CST Lemmatiser*. Center for Sprogteknologi, University of Copenhagen version 2.7 (August, 23 2005) http://cst.dk/online/lemmatiser/cstlemma.pdf

Jakub Kanis and Ludek Müller. 2005. Automatic Lemmatizer Construction with Focus on OOV Words Lemmatization in Text, Speech and Dialogue, *Lecture Notes in Computer Science*, Berlin / Heidelberg, pp 132-139

Ron Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* 2 (12): 1137–1143, Morgan Kaufmann, San Mateo.

Prasenjit Majumder, Mandar Mitra, Swapan K. Parui, Gobinda Kole, Pabitra Mitra, and Kalyankumar Datta. 2007. YASS: Yet another suffix stripper. *ACM Transactions on Information Systems* , Volume 25 , Issue 4, October 2007.

Juršič Matjaž, Igor Mozetič, and Nada Lavrač. 2007. Learning ripple down rules for efficient lemmatization In *proceeding of the Conference on Data Mining and Data Warehouses* (SiKDD 2007), October 12, 2007, Ljubljana, Slovenia

Morfologik: Polish morphological analyzer http://mac.softpedia.com/get/Word-Processing/Morfologik.shtml

Douglas W. Oard, Gina-Anne Levow, and Clara I. Cabezas. 2001. CLEF experiments at Maryland: Statistical stemming and backoff translation. In Cross-language information retrieval and evaluation: *Proceeding of the Clef 2000 workshops* Carol Peters Ed. Springer Verlag pp. 176-187. 2001.

Georgios Petasis, Vangelis Karkaletsis , Dimitra Farmakiotou , Ion Androutsopoulos and Constantine D. Spyropoulo. 2003. A Greek Morphological Lexicon and its Exploitation by Natural Language Processing Applications. In Lecture Notes on Computer Science (LNCS), vol.2563, "Advances in Informatics - Post-proceedings of the 8th Panhellenic Conference in Informatics", Springer Verlag.

Joël Plisson, Nada Lavrač, and Dunja Mladenic. 2004, A rule based approach to word lemmatization, *Proceedings of the 7th International Multiconference Information Society*, IS-2004, Institut Jozef Stefan, Ljubljana, pp.83-6.

Martin F. Porter 1980. An algorithm for suffix stripping. *Program*, vol 14, no 3, pp 130-130.

John W. Ratcliff and David Metzener, 1988. Pattern Matching: The Gestalt Approach, *Dr. Dobb's Journal*, page 46, July 1988.

SCARRIE 2009. Scandinavian Proofreading Tools http://ling.uib.no/~desmedt/scarrie/

STO: http://cst.ku.dk/sto_ordbase/

SUC 2009. Stockholm Umeå corpus, http://www.ling.su.se/staff/sofia/suc/suc.html

Pieter Theron and Ian Cloete 1997 Automatic acquisition of two-level morphological rules, *Proceedings of the fifth conference on Applied natural language processing*, p.103-110, March 31-April 03, 1997, Washington, DC.

Ellen M. Voorhees. 2000. Variations in relevance judgments and the measurement of retrieval effectiveness, J. of Information Processing and Management 36 (2000) pp 697-716

Revisiting Pivot Language Approach for Machine Translation

Hua Wu and **Haifeng Wang**
Toshiba (China) Research and Development Center
5/F., Tower W2, Oriental Plaza, Beijing, 100738, China
{wuhua, wanghaifeng}@rdc.toshiba.com.cn

Abstract

This paper revisits the pivot language approach for machine translation. First, we investigate three different methods for pivot translation. Then we employ a hybrid method combining RBMT and SMT systems to fill up the data gap for pivot translation, where the source-pivot and pivot-target corpora are independent. Experimental results on spoken language translation show that this hybrid method significantly improves the translation quality, which outperforms the method using a source-target corpus of the same size. In addition, we propose a system combination approach to select better translations from those produced by various pivot translation methods. This method regards system combination as a translation evaluation problem and formalizes it with a regression learning model. Experimental results indicate that our method achieves consistent and significant improvement over individual translation outputs.

1 Introduction

Current statistical machine translation (SMT) systems rely on large parallel and monolingual training corpora to produce translations of relatively higher quality. Unfortunately, large quantities of parallel data are not readily available for some languages pairs, therefore limiting the potential use of current SMT systems. In particular, for speech translation, the translation task often focuses on a specific domain such as the travel domain. It is especially difficult to obtain such a domain-specific corpus for some language pairs such as Chinese to Spanish translation.

To circumvent the data bottleneck, some researchers have investigated to use a pivot language approach (Cohn and Lapata, 2007; Utiyama and Isahara, 2007; Wu and Wang 2007; Bertoldi et al., 2008). This approach introduces a third language, named the pivot language, for which there exist large source-pivot and pivot-target bilingual corpora. A pivot task was also designed for spoken language translation in the evaluation campaign of IWSLT 2008 (Paul, 2008), where English is used as a pivot language for Chinese to Spanish translation.

Three different pivot strategies have been investigated in the literature. The first is based on phrase table multiplication (Cohn and Lapata 2007; Wu and Wang, 2007). It multiples corresponding translation probabilities and lexical weights in source-pivot and pivot-target translation models to induce a new source-target phrase table. We name it the *triangulation* method. The second is the sentence translation strategy, which first translates the source sentence to the pivot sentence, and then to the target sentence (Utiyama and Isahara, 2007; Khalilov et al., 2008). We name it the *transfer* method. The third is to use existing models to build a synthetic source-target corpus, from which a source-target model can be trained (Bertoldi et al., 2008). For example, we can obtain a source-pivot corpus by translating the pivot sentence in the source-pivot corpus into the target language with pivot-target translation models. We name it the *synthetic* method.

The working condition with the pivot language approach is that the source-pivot and pivot-target parallel corpora are independent, in the sense that they are not derived from the same set of sentences, namely independently sourced corpora. Thus, some linguistic phenomena in the source-pivot corpus will lost if they do not exist in the pivot-target corpus, and vice versa. In order to fill up this data gap, we make use of rule-based machine translation (RBMT) systems to translate the pivot sentences in the source-pivot or pivot-target

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 154–162,
Suntec, Singapore, 2-7 August 2009.

corpus into target or source sentences. As a result, we can build a synthetic multilingual corpus, which can be used to improve the translation quality. The idea of using RBMT systems to improve the translation quality of SMT sysems has been explored in Hu et al. (2007). Here, we re-examine the hybrid method to fill up the data gap for pivot translation.

Although previous studies proposed several pivot translation methods, there are no studies to combine different pivot methods for translation quality improvement. In this paper, we first compare the individual pivot methods and then investigate to improve pivot translation quality by combining the outputs produced by different systems. We propose to regard system combination as a translation evaluation problem. For translations from one of the systems, this method uses the outputs from other translation systems as pseudo references. A regression learning method is used to infer a function that maps a feature vector (which measures the similarity of a translation to the pseudo references) to a score that indicates the quality of the translation. Scores are first generated independently for each translation, then the translations are ranked by their respective scores. The candidate with the highest score is selected as the final translation. This is achieved by optimizing the regression learning model's output to correlate against a set of training examples, where the source sentences are provided with several reference translations, instead of manually labeling the translations produced by various systems with quantitative assessments as described in (Albrecht and Hwa, 2007; Duh, 2008). The advantage of our method is that we do not need to manually label the translations produced by each translation system, therefore enabling our method suitable for translation selection among any systems without additional manual work.

We conducted experiments for spoken language translation on the pivot task in the IWSLT 2008 evaluation campaign, where Chinese sentences in travel domain need to be translated into Spanish, with English as the pivot language. Experimental results show that (1) the performances of the three pivot methods are comparable when only SMT systems are used. However, the triangulation method and the transfer method significantly outperform the synthetic method when RBMT systems are used to improve the translation quality; (2) The hybrid method combining SMT and RBMT system for pivot translation greatly improves the translation quality. And this translation quality is higher than that of those produced by the system trained with a real Chinese-Spanish corpus; (3) Our sentence-level translation selection method consistently and significantly improves the translation quality over individual translation outputs in all of our experiments.

Section 2 briefly introduces the three pivot translation methods. Section 3 presents the hybrid method combining SMT and RBMT systems. Section 4 describes the translation selection method. Experimental results are presented in Section 5, followed by a discussion in Section 6. The last section draws conclusions.

2 Pivot Methods for Phrase-based SMT

2.1 Triangulation Method

Following the method described in Wu and Wang (2007), we train the source-pivot and pivot-target translation models using the source-pivot and pivot-target corpora, respectively. Based on these two models, we induce a source-target translation model, in which two important elements need to be induced: phrase translation probability and lexical weight.

Phrase Translation Probability We induce the phrase translation probability by assuming the independence between the source and target phrases when given the pivot phrase.

$$\phi(\bar{s}|\bar{t}) = \sum_{\bar{p}} \phi(\bar{s}|\bar{p})\phi(\bar{p}|\bar{t}) \quad (1)$$

Where $\bar{s}$, $\bar{p}$ and $\bar{t}$ represent the phrases in the languages L_s, L_p and L_t, respectively.

Lexical Weight According to the method described in Koehn et al. (2003), there are two important elements in the lexical weight: word alignment information a in a phrase pair $(\bar{s}, \bar{t})$ and lexical translation probability $w(s|t)$.

Let a_1 and a_2 represent the word alignment information inside the phrase pairs $(\bar{s}, \bar{p})$ and $(\bar{p}, \bar{t})$ respectively, then the alignment information inside $(\bar{s}, \bar{t})$ can be obtained as shown in Eq. (2).

$$a = \{(s,t)|\exists p : (s,p) \in a_1 \ \& \ (p,t) \in a_2\} \quad (2)$$

Based on the the induced word alignment information, we estimate the co-occurring frequencies of word pairs directly from the induced phrase

pairs. Then we estimate the lexical translation probability as shown in Eq. (3).

$$w(s|t) = \frac{count(s,t)}{\sum_{s'} count(s',t)} \quad (3)$$

Where $count(s,t)$ represents the co-occurring frequency of the word pair (s,t).

2.2 Transfer Method

The transfer method first translates from the source language to the pivot language using a source-pivot model, and then from the pivot language to the target language using a pivot-target model. Given a source sentence s, we can translate it into n pivot sentences $p_1, p_2, ..., p_n$ using a source-pivot translation system. Each p_i can be translated into m target sentences $t_{i1}, t_{i2}, ..., t_{im}$. We rescore all the $n \times m$ candidates using both the source-pivot and pivot-target translation scores following the method described in Utiyama and Isahara (2007). If we use h^{fp} and h^{pt} to denote the features in the source-pivot and pivot-target systems, respectively, we get the optimal target translation according to the following formula.

$$\hat{t} = \underset{t}{\operatorname{argmax}} \sum_{k=1}^{L} (\lambda_k^{sp} h_k^{sp}(s,p) + \lambda_k^{pt} h_k^{pt}(p,t)) \quad (4)$$

Where L is the number of features used in SMT systems. λ^{sp} and λ^{pt} are feature weights set by performing minimum error rate training as described in Och (2003).

2.3 Synthetic Method

There are two possible methods to obtain a source-target corpus using the source-pivot and pivot-target corpora. One is to obtain target translations for the source sentences in the source-pivot corpus. This can be achieved by translating the pivot sentences in source-pivot corpus to target sentences with the pivot-target SMT system. The other is to obtain source translations for the target sentences in the pivot-target corpus using the pivot-source SMT system. And we can combine these two source-target corpora to produced a final synthetic corpus.

Given a pivot sentence, we can translate it into n source or target sentences. These n translations together with their source or target sentences are used to create a synthetic bilingual corpus. Then we build a source-target translation model using this corpus.

3 Using RBMT Systems for Pivot Translation

Since the source-pivot and pivot-target parallel corpora are independent, the pivot sentences in the two corpora are distinct from each other. Thus, some linguistic phenomena in the source-pivot corpus will lost if they do not exist in the pivot-target corpus, and vice versa. Here we use RBMT systems to fill up this data gap. For many source-target language pairs, the commercial pivot-source and/or pivot-target RBMT systems are available on markets. For example, for Chinese to Spanish translation, English to Chinese and English to Spanish RBMT systems are available.

With the RBMT systems, we can create a synthetic multilingual source-pivot-target corpus by translating the pivot sentences in the pivot-source or pivot-target corpus. The source-target pairs extracted from this synthetic multilingual corpus can be used to build a source-target translation model. Another way to use the synthetic multilingual corpus is to add the source-pivot or pivot-target sentence pairs in this corpus to the training data to rebuild the source-pivot or pivot-target SMT model. The rebuilt models can be applied to the triangulation method and the transfer method as described in Section 2.

Moreover, the RBMT systems can also be used to enlarge the size of bilingual training data. Since it is easy to obtain monolingual corpora than bilingual corpora, we use RBMT systems to translate the available monolingual corpora to obtain synthetic bilingual corpus, which are added to the training data to improve the performance of SMT systems. Even if no monolingual corpus is available, we can also use RBMT systems to translate the sentences in the bilingual corpus to obtain alternative translations. For example, we can use source-pivot RBMT systems to provide alternative translations for the source sentences in the source-pivot corpus.

In addition to translating training data, the source-pivot RBMT system can be used to translate the test set into the pivot language, which can be further translated into the target language with the pivot-target RBMT system. The translated test set can be added to the training data to further improve translation quality. The advantage of this method is that the RBMT system can provide translations for sentences in the test set and cover some out-of-vocabulary words in the test set

that are uncovered by the training data. It can also change the distribution of some phrase pairs and reinforce some phrase pairs relative to the test set.

4 Translation Selection

We propose a method to select the optimal translation from those produced by various translation systems. We regard sentence-level translation selection as a machine translation (MT) evaluation problem and formalize this problem with a regression learning model. For each translation, this method uses the outputs from other translation systems as pseudo references. The regression objective is to infer a function that maps a feature vector (which measures the similarity of a translation from one system to the pseudo references) to a score that indicates the quality of the translation. Scores are first generated independently for each translation, then the translations are ranked by their respective scores. The candidate with the highest score is selected.

The similar ideas have been explored in previous studies. Albrecht and Hwa (2007) proposed a method to evaluate MT outputs with pseudo references using support vector regression as the learner to evaluate translations. Duh (2008) proposed a ranking method to compare the translations proposed by several systems. These two methods require quantitative quality assessments by human judges for the translations produced by various systems in the training set. When we apply such methods to translation selection, the relative values of the scores assigned by the subject systems are important. In different data conditions, the relative values of the scores assigned by the subject systems may change. In order to train a reliable learner, we need to prepare a balanced training set, where the translations produced by different systems under different conditions are required to be manually evaluated. In extreme cases, we need to relabel the training data to obtain better performance. In this paper, we modify the method in Albrecht and Hwa (2007) to only prepare human reference translations for the training examples, and then evaluate the translations produced by the subject systems against the references using BLEU score (Papineni et al., 2002). We use smoothed sentence-level BLEU score to replace the human assessments, where we use additive smoothing to avoid zero BLEU scores when we calculate the n-gram precisions. In this case, we can easily retrain the learner under different conditions, therefore enabling our method to be applied to sentence-level translation selection from any sets of translation systems without any additional human work.

ID	Description
1-4	n-gram precisions against pseudo references ($1 \leq n \leq 4$)
5-6	PER and WER
7-8	precision, recall, fragmentation from METEOR (Lavie and Agarwal, 2007)
9-12	precisions and recalls of non-consecutive bigrams with a gap size of m ($1 \leq m \leq 2$)
13-14	longest common subsequences
15-19	n-gram precision against a target corpus ($1 \leq n \leq 5$)

Table 1: Feature sets for regression learning

In regression learning, we infer a function f that maps a multi-dimensional input vector $\mathbf{x}$ to a continuous real value y, such that the error over a set of m training examples, $(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), ..., (\mathbf{x_m}, y_m)$, is minimized according to a loss function. In the context of translation selection, y is assigned as the smoothed BLEU score. The function f represents a mathematic model of the automatic evaluation metrics. The input sentence is represented as a feature vector $\mathbf{x}$, which are extracted from the input sentence and the comparisons against the pseudo references. We use the features as shown in Table 1.

5 Experiments

5.1 Data

We performed experiments on spoken language translation for the pivot task of IWSLT 2008. This task translates Chinese to Spanish using English as the pivot language. Table 2 describes the data used for model training in this paper, including the BTEC (Basic Travel Expression Corpus) Chinese-English (CE) corpus and the BTEC English-Spanish (ES) corpus provided by IWSLT 2008 organizers, the HIT olympic CE corpus (2004-863-008)[1] and the Europarl ES corpus[2]. There are two kinds of BTEC CE corpus: BTEC CE1 and

[1] http://www.chineseldc.org/EN/purchasing.htm
[2] http://www.statmt.org/europarl/

Corpus	Size	SW	TW
BTEC CE1	20,000	164K	182K
BTEC CE2	18,972	177K	182K
HIT CE	51,791	490K	502K
BTEC ES	19,972	182K	185K
Europarl ES	400,000	8,485K	8,219K

Table 2: Training data. SW and TW represent source words and target words, respectively.

BTEC CE2. BTEC CE1 was distributed for the pivot task in IWSLT 2008 while BTEC CE2 was for the BTEC CE task, which is parallel to the BTEC ES corpus. For Chinese-English translation, we mainly used BTEC CE1 corpus. We used the BTEC CE2 corpus and the HIT Olympic corpus for comparison experiments only. We used the English parts of the BTEC CE1 corpus, the BTEC ES corpus, and the HIT Olympic corpus (if involved) to train a 5-gram English language model (LM) with interpolated Kneser-Ney smoothing. For English-Spanish translation, we selected 400k sentence pairs from the Europarl corpus that are close to the English parts of both the BTEC CE corpus and the BTEC ES corpus. Then we built a Spanish LM by interpolating an out-of-domain LM trained on the Spanish part of this selected corpus with the in-domain LM trained with the BTEC corpus.

For Chinese-English-Spanish translation, we used the development set (devset3) released for the pivot task as the test set, which contains 506 source sentences, with 7 reference translations in English and Spanish. To be capable of tuning parameters on our systems, we created a development set of 1,000 sentences taken from the training sets, with 3 reference translations in both English and Spanish. This development set is also used to train the regression learning model.

5.2 Systems and Evaluation Method

We used two commercial RBMT systems in our experiments: System A for Chinese-English bidirectional translation and System B for English-Chinese and English-Spanish translation. For phrase-based SMT translation, we used the Moses decoder (Koehn et al., 2007) and its support training scripts. We ran the decoder with its default settings and then used Moses' implementation of minimum error rate training (Och, 2003) to tune the feature weights on the development set.

To select translation among outputs produced by different pivot translation systems, we used SVM-light (Joachins, 1999) to perform support vector regression with the linear kernel.

Translation quality was evaluated using both the BLEU score proposed by Papineni et al. (2002) and also the modified BLEU (BLEU-Fix) score[3] used in the IWSLT 2008 evaluation campaign, where the brevity calculation is modified to use closest reference length instead of shortest reference length.

5.3 Results by Using SMT Systems

We conducted the pivot translation experiments using the BTEC CE1 and BTEC ES described in Section 5.1. We used the three methods described in Section 2 for pivot translation. For the transfer method, we selected the optimal translations among 10×10 candidates. For the synthetic method, we used the ES translation model to translate the English part of the CE corpus to Spanish to construct a synthetic corpus. And we also used the BTEC CE1 corpus to build a EC translation model to translate the English part of ES corpus into Chinese. Then we combined these two synthetic corpora to build a Chinese-Spanish translation model. In our experiments, only 1-best Chinese or Spanish translation was used since using n-best results did not greatly improve the translation quality. We used the method described in Section 4 to select translations from the translations produced by the three systems. For each system, we used three different alignment heuristics (grow, grow-diag, grow-diag-final[4]) to obtain the final alignment results, and then constructed three different phrase tables. Thus, for each system, we can get three different translations for each input. These different translations can serve as pseudo references for the outputs of other systems. In our case, for each sentence, we have 6 pseudo reference translations. In addition, we found out that the *grow* heuristic performed the best for all the systems. Thus, for an individual system, we used the translation results produced using the *grow* alignment heuristic.

The translation results are shown in Table 3. ASR and CRR represent different input conditions, namely the result of automatic speech recog-

[3] https://www.slc.atr.jp/Corpus/IWSLT08/eval/IWSLT08_auto_eval.tgz

[4] A description of the alignment heuristics can be found at http://www.statmt.org/jhuws/?n=FactoredTraining.TrainingParameters

Method	BLEU	BLEU-Fix
Triangulation	33.70/27.46	31.59/25.02
Transfer	33.52/28.34	31.36/26.20
Synthetic	34.35/27.21	32.00/26.07
Combination	38.14/29.32	34.76/27.39

Table 3: CRR/ASR translation results by using SMT systems

nition and correct recognition result, respectively. Here, we used the 1-best ASR result. From the translation results, it can be seen that three methods achieved comparable translation quality on both ASR and CRR inputs, with the translation results on CRR inputs are much better than those on ASR inputs because of the errors in the ASR inputs. The results also show that our translation selection method is very effective, which achieved absolute improvements of about 4 and 1 BLEU scores on CRR and ASR inputs, respectively.

5.4 Results by Using both RBMT and SMT Systems

In order to fill up the data gap as discussed in Section 3, we used the RBMT System A to translate the English sentences in the ES corpus into Chinese. As described in Section 3, this corpus can be used by the three pivot translation methods. First, the synthetic Chinese-Spanish corpus can be combined with those produced by the EC and ES SMT systems, which were used in the synthetic method. Second, the synthetic Chinese-English corpus can be added into the BTEC CE1 corpus to build the CE translation model. In this way, the intersected English phrases in the CE corpus and ES corpus becomes more, which enables the Chinese-Spanish translation model induced using the triangulation method to cover more phrase pairs. For the transfer method, the CE translation quality can be also improved, which would result in the improvement of the Spanish translation quality.

The translation results are shown in the columns under "EC RBMT" in Table 4. As compared with those in Table 3, the translation quality was greatly improved, with absolute improvements of at least 5.1 and 3.9 BLEU scores on CRR and ASR inputs for system combination results. The above results indicate that RBMT systems indeed can be used to fill up the data gap for pivot translation.

In our experiments, we also used a CE RBMT system to enlarge the size of training data by providing alternative English translations for the Chinese part of the CE corpus. The translation results are shown in the columns under "+CE RBMT" in Table 4. From the translation results, it can be seen that, enlarging the size of training data with RBMT systems can further improve the translation quality.

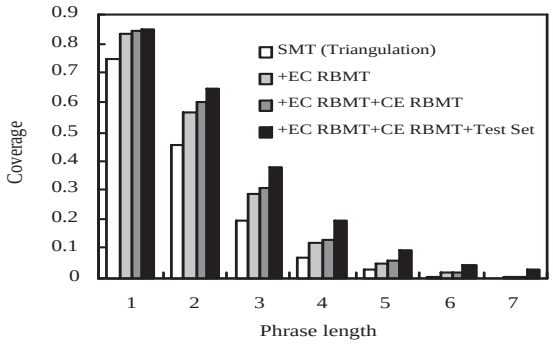


Figure 1: Coverage on test source phrases

In addition to translating the training data, the CE RBMT system can be also used to translate the test set into English, which can be further translated into Spanish with the ES RBMT system B.[5][6] The translated test set can be further added to the training data to improve translation quality. The columns under "+Test Set" in Table 4 describes the translation results. The results show that translating the test set using RBMT systems greatly improved the translation result, with further improvements of about 2 and 1.5 BLEU scores on CRR and ASR inputs, respectively.

The results also indicate that both the triangulation method and the transfer method greatly outperformed the synthetic method when we combined both RBMT and SMT systems in our experiments. Further analysis shows that the synthetic method contributed little to system combination. The selection results are almost the same as those selected from the translations produced by the triangulation and transfer methods.

In order to further analyze the translation results, we evaluated the above systems by examining the coverage of the phrase tables over the test phrases. We took the triangulation method as a case study, the results of which are shown in Fig-

[5]Although using the ES RBMT system B to translate the training data did not improve the translation quality, it improved the translation quality by translating the test set.

[6]The RBMT systems achieved a BLEU score of 24.36 on the test set.

	EC RBMT		+ CE RBMT		+ Test Set	
Method	**BLEU**	**BLEU-Fix**	**BLEU**	**BLEU-Fix**	**BLEU**	**BLEU-Fix**
Triangulation	40.69/31.02	37.99/29.15	41.59/31.43	39.39/29.95	44.71/32.60	42.37/31.14
Transfer	42.06/31.72	39.73/29.35	43.40/33.05	40.73/30.06	45.91/34.52	42.86/31.92
Synthetic	39.10/29.73	37.26/28.45	39.90/30.00	37.90/28.66	41.16/31.30	37.99/29.36
Combination	43.21/33.23	40.58/31.17	45.09/34.10	42.88/31.73	47.06/35.62	44.94/32.99

Table 4: CRR/ASR translation results by using RBMT and SMT systems

Method	**BLEU**	**BLEU-Fix**
Triangulation	45.64/33.15	42.11/31.11
Transfer	47.18/34.56	43.61/32.17
Combination	48.42/36.42	45.42/33.52

Table 5: CRR/ASR translation results by using additional monolingual corpora

Method	**Sys. A**	**Sys. B**	**Sys. A+B**
Triangulation	40.69	39.28	41.01
Transfer	42.06	39.57	43.03
Synthetic	39.10	38.24	39.26
Combination	43.21	40.59	44.27

Table 6: CRR translation results (BLEU scores) by using different RBMT systems

ure 1. It can be seen that using RBMT systems to translate the training and/or test data can cover more source phrases in the test set, which results in translation quality improvement.

5.5 Results by Using Monolingual Corpus

In addition to translating the limited bilingual corpus, we also translated additional monolingual corpus to further enlarge the size of the training data. We assume that it is easier to obtain a monolingual pivot corpus than to obtain a monolingual source or target corpus. Thus, we translated the English part of the HIT Olympic corpus into Chinese and Spanish using EC and ES RBMT systems. The generated synthetic corpus was added to the training data to train EC and ES SMT systems. Here, we used the synthetic CE Olympic corpus to train a model, which was interpolated with the CE model trained with both the BTEC CE1 corpus and the synthetic BTEC corpus to obtain an interpolated CE translation model. Similarly, we obtained an interpolated ES translation model. Table 5 describes the translation results.[7] The results indicate that translating monolingual corpus using the RBMT system further improved the translation quality as compared with those in Table 4.

6 Discussion

6.1 Effects of Different RBMT Systems

In this section, we compare the effects of two commercial RBMT systems with different translation accuracy on spoken language translation. The goals are (1) to investigate whether a RBMT system can improve pivot translation quality even if its translation accuracy is not high, and (2) to compare the effects of RBMT system with different translation accuracy on pivot translation. Besides the EC RBMT system A used in the above section, we also used the EC RBMT system B for this experiment.

We used the two systems to translate the test set from English to Chinese, and then evaluated the translation quality against Chinese references obtained from the IWSLT 2008 evaluation campaign. The BLEU scores are 43.90 and 29.77 for System A and System B, respectively. This shows that the translation quality of System B on spoken language corpus is much lower than that of System A. Then we applied these two different RBMT systems to translate the English part of the BTEC ES corpus into Chinese as described in Section 5.4. The translation results on CRR inputs are shown in Table 6.[8] We replicated some of the results in Table 4 for the convenience of comparison. The results indicate that the higher the translation accuracy of the RBMT system is, the better the pivot translation is. If we compare the results with those only using SMT systems as described in Table 3, the translation quality was greatly improved by at least 3 BLEU scores, even if the translation ac-

[7] Here we excluded the synthetic method since it greatly falls behind the other two methods.

[8] We omitted the ASR translation results since the trends are the same as those for CRR inputs. And we only showed BLEU scores since the trend for BLEU-Fix scores is similar.

Method	Multilingual	+ BTEC CE1
Triangulation	41.86/39.55	42.41/39.55
Transfer	42.46/39.09	43.84/40.34
Standard	42.21/40.23	42.21/40.23
Combination	43.75/40.34	44.68/41.14

Table 7: CRR translation results by using multilingual corpus. "/" separates the BLEU and BLEU-fix scores.

curacy of System B is not so high. Combining two RBMT systems further improved the translation quality, which indicates that the two systems complement each other.

6.2 Results by Using Multilingual Corpus

In this section, we compare the translation results by using a multilingual corpus with those by using independently sourced corpora. BTEC CE2 and BTEC ES are from the same source sentences, which can be taken as a multilingual corpus. The two corpora were employed to build CE and ES SMT models, which were used in the triangulation method and the transfer method. We also extracted the Chinese-Spanish (CS) corpus to build a standard CS translation system, which is denoted as *Standard*. The comparison results are shown in Table 7. The translation quality produced by the systems using a multilingual corpus is much higher than that produced by using independently sourced corpora as described in Table 3, with an absolute improvement of about 5.6 BLEU scores. If we used the EC RBMT system, the translation quality of those in Table 4 is comparable to that by using the multilingual corpus, which indicates that our method using RBMT systems to fill up the data gap is effective. The results also indicate that our translation selection method for pivot translation outperforms the method using only a real source-target corpus.

For comparison purpose, we added BTEC CE1 into the training data. The translation quality was improved by only 1 BLEU score. This again proves that our method to fill up the data gap is more effective than that to increase the size of the independently sourced corpus.

6.3 Comparison with Related Work

In IWSLT 2008, the best result for the pivot task is achieved by Wang et al. (2008). In order to compare the results, we added the bilingual HIT Olympic corpus into the CE training data.[9] We also compared our translation selection method with that proposed in (Wang et al., 2008) that is based on the target sentence average length (TSAL). The translation results are shown in Table 8. "Wang" represents the results in Wang et al. (2008). "TSAL" represents the translation selection method proposed in Wang et al. (2008), which is applied to our experiment. From the results, it can be seen that our method outperforms the best system in IWSLT 2008 and that our translation selection method outperforms the method based on target sentence average length.

	Ours	Wang	TSAL
BLEU	49.57	-	48.25
BLEU-Fix	46.74	45.10	45.27

Table 8: Comparison with related work

7 Conclusion

In this paper, we have compared three different pivot translation methods for spoken language translation. Experimental results indicated that the triangulation method and the transfer method generally outperform the synthetic method. Then we showed that the hybrid method combining RBMT and SMT systems can be used to fill up the data gap between the source-pivot and pivot-target corpora. By translating the pivot sentences in independent corpora, the hybrid method can produce translations whose quality is higher than those produced by the method using a source-target corpus of the same size. We also showed that even if the translation quality of the RBMT system is low, it still greatly improved the translation quality.

In addition, we proposed a system combination method to select better translations from outputs produced by different pivot methods. This method is developed through regression learning, where only a small size of training examples with reference translations are required. Experimental results indicate that this method can consistently and significantly improve translation quality over individual translation outputs. And our system outperforms the best system for the pivot task in the IWSLT 2008 evaluation campaign.

[9] We used about 70k sentence pairs for CE model training, while Wang et al. (2008) used about 100k sentence pairs, a CE translation dictionary and more monolingual corpora for model training.

References

Joshua S. Albrecht and Rebecca Hwa. 2007. Regression for Sentence-Level MT Evaluation with Pseudo References. In *Proceedings of the 45th Annual Meeting of the Accosiation of Computational Linguistics*, pages 296–303.

Nicola Bertoldi, Madalina Barbaiani, Marcello Federico, and Roldano Cattoni. 2008. Phrase-Based Statistical Machine Translation with Pivot Languages. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 143-149.

Tevor Cohn and Mirella Lapata. 2007. Machine Translation by Triangulation: Making Effective Use of Multi-Parallel Corpora. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 348–355.

Kevin Duh. 2008. Ranking vs. Regression in Machine Translation Evaluation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 191–194.

Xiaoguang Hu, Haifeng Wang, and Hua Wu. 2007. Using RBMT Systems to Produce Bilingual Corpus for SMT. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 287–295.

Thorsten Joachims. 1999. Making Large-Scale SVM Learning Practical. In Bernhard Schöelkopf, Christopher Burges, and Alexander Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

Maxim Khalilov, Marta R. Costa-Jussà, Carlos A. Henríquez, José A.R. Fonollosa, Adolfo Hernández, José B. Mariño, Rafael E. Banchs, Chen Boxing, Min Zhang, Aiti Aw, and Haizhou Li. 2008. The TALP & I2R SMT Systems for IWSLT 2008. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 116–123.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *HLT-NAACL: Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133.

Philipp Koehn, Hieu Hoang, Alexanda Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Associa-tion for Computational Linguistics, demonstration session*, pages 177–180.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In *Proceedings of Workshop on Statistical Machine Translation at the 45th Annual Meeting of the Association of Computational Linguistics*, pages 228–231.

Franz J. Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Michael Paul. 2008. Overview of the IWSLT 2008 Evaluation Campaign. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 1–17.

Masao Utiyama and Hitoshi Isahara. 2007. A Comparison of Pivot Methods for Phrase-Based Statistical Machine Translation. In *Proceedings of human language technology: the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 484–491.

Haifeng Wang, Hua Wu, Xiaoguang Hu, Zhanyi Liu, Jianfeng Li, Dengjun Ren, and Zhengyu Niu. 2008. The TCH Machine Translation System for IWSLT 2008. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 124–131.

Hua Wu and Haifeng Wang. 2007. Pivot Language Approach for Phrase-Based Statistical Machine Translation. In *Proceedings of 45th Annual Meeting of the Association for Computational Linguistics*, pages 856–863.

Efficient Minimum Error Rate Training and Minimum Bayes-Risk Decoding for Translation Hypergraphs and Lattices

Shankar Kumar[1] and **Wolfgang Macherey**[1] and **Chris Dyer**[2] and **Franz Och**[1]

[1]Google Inc.
1600 Amphitheatre Pkwy.
Mountain View, CA 94043, USA
{shankarkumar,wmach,och}@google.com

[2]Department of Linguistics
University of Maryland
College Park, MD 20742, USA
redpony@umd.edu

Abstract

Minimum Error Rate Training (MERT) and Minimum Bayes-Risk (MBR) decoding are used in most current state-of-the-art Statistical Machine Translation (SMT) systems. The algorithms were originally developed to work with N-best lists of translations, and recently extended to lattices that encode many more hypotheses than typical N-best lists. We here extend lattice-based MERT and MBR algorithms to work with hypergraphs that encode a vast number of translations produced by MT systems based on Synchronous Context Free Grammars. These algorithms are more efficient than the lattice-based versions presented earlier. We show how MERT can be employed to optimize parameters for MBR decoding. Our experiments show speedups from MERT and MBR as well as performance improvements from MBR decoding on several language pairs.

1 Introduction

Statistical Machine Translation (SMT) systems have improved considerably by directly using the error criterion in both training and decoding. By doing so, the system can be optimized for the translation task instead of a criterion such as likelihood that is unrelated to the evaluation metric. Two popular techniques that incorporate the error criterion are *Minimum Error Rate Training* (MERT) (Och, 2003) and *Minimum Bayes-Risk* (MBR) decoding (Kumar and Byrne, 2004). These two techniques were originally developed for N-best lists of translation hypotheses and recently extended to *translation lattices* (Macherey et al., 2008; Tromble et al., 2008) generated by a phrase-based SMT system (Och and Ney, 2004). Translation lattices contain a significantly higher number of translation alternatives relative to N-best lists. The extension to lattices reduces the runtimes for both MERT and MBR, and gives performance improvements from MBR decoding.

SMT systems based on *synchronous context free grammars* (SCFG) (Chiang, 2007; Zollmann and Venugopal, 2006; Galley et al., 2006) have recently been shown to give competitive performance relative to phrase-based SMT. For these systems, a *hypergraph* or *packed forest* provides a compact representation for encoding a huge number of translation hypotheses (Huang, 2008).

In this paper, we extend MERT and MBR decoding to work on hypergraphs produced by SCFG-based MT systems. We present algorithms that are more efficient relative to the lattice algorithms presented in Macherey et al. (2008; Tromble et al. (2008). Lattice MBR decoding uses a linear approximation to the BLEU score (Papineni et al., 2001); the weights in this linear loss are set heuristically by assuming that n-gram precisions decay exponentially with n. However, this may not be optimal in practice. We employ MERT to select these weights by optimizing BLEU score on a development set.

A related MBR-inspired approach for hypergraphs was developed by Zhang and Gildea (2008). In this work, hypergraphs were rescored to maximize the expected count of synchronous constituents in the translation. In contrast, our MBR algorithm directly selects the hypothesis in the hypergraph with the maximum expected approximate corpus BLEU score (Tromble et al., 2008).

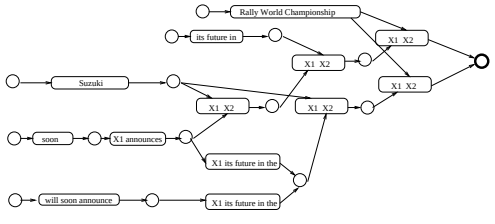


Figure 1: An example hypergraph.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 163–171,
Suntec, Singapore, 2-7 August 2009.

2 Translation Hypergraphs

A translation lattice compactly encodes a large number of hypotheses produced by a phrase-based SMT system. The corresponding representation for an SMT system based on SCFGs (e.g. Chiang (2007), Zollmann and Venugopal (2006), Mi et al. (2008)) is a *directed hypergraph* or a *packed forest* (Huang, 2008).

Formally, a hypergraph is a pair $\mathcal{H} = \langle \mathcal{V}, \mathcal{E} \rangle$ consisting of a vertex set $\mathcal{V}$ and a set of hyperedges $\mathcal{E} \subseteq \mathcal{V}^* \times \mathcal{V}$. Each hyperedge $e \in \mathcal{E}$ connects a head vertex $h(e)$ with a sequence of tail vertices $T(e) = \{v_1, ..., v_n\}$. The number of tail vertices is called the *arity* ($|e|$) of the hyperedge. If the arity of a hyperedge is zero, $h(e)$ is called a *source vertex*. The arity of a hypergraph is the maximum arity of its hyperedges. A hyperedge of arity 1 is a *regular edge*, and a hypergraph of arity 1 is a *regular graph* (lattice). Each hyperedge is labeled with a rule r_e from the SCFG. The number of nonterminals on the right-hand side of r_e corresponds with the arity of e. An example without scores is shown in Figure 1. A *path* in a translation hypergraph induces a translation hypothesis E along with its sequence of SCFG rules $D = r_1, r_2, ..., r_K$ which, if applied to the start symbol, derives E. The sequence of SCFG rules induced by a path is also called a *derivation tree for* E.

3 Minimum Error Rate Training

Given a set of source sentences F_1^S with corresponding reference translations R_1^S, the objective of MERT is to find a parameter set $\hat{\lambda}_1^M$ which minimizes an automated evaluation criterion under a linear model:

$$\hat{\lambda}_1^M = \arg\min_{\lambda_1^M} \left\{ \sum_{s=1}^{S} \mathrm{Err}\big(R_s, \hat{E}(F_s;\, \lambda_1^M)\big) \right\}$$

$$\hat{E}(F_s;\, \lambda_1^M) = \arg\max_{E} \left\{ \sum_{s=1}^{S} \lambda_m h_m(E,\, F_s) \right\}.$$

In the context of statistical machine translation, the optimization procedure was first described in Och (2003) for N-best lists and later extended to phrase-lattices in Macherey et al. (2008). The algorithm is based on the insight that, under a log-linear model, the cost function of any candidate translation can be represented as a line in the plane if the initial parameter set λ_1^M is shifted along a direction d_1^M. Let $\mathcal{C} = \{E_1, ..., E_K\}$ denote a set of candidate translations, then computing the best scoring translation hypothesis $\hat{E}$ out of $\mathcal{C}$ results in the following optimization problem:

$$\begin{aligned}
\hat{E}(F;\gamma) &= \arg\max_{E \in \mathcal{C}} \left\{ (\lambda_1^M + \gamma \cdot d_1^M)^\top \cdot h_1^M(E,F) \right\} \\
&= \arg\max_{E \in \mathcal{C}} \left\{ \underbrace{\sum_m \lambda_m h_m(E,F)}_{=a(E,F)} + \gamma \cdot \underbrace{\sum_m d_m h_m(E,F)}_{=b(E,F)} \right\} \\
&= \arg\max_{E \in \mathcal{C}} \big\{ \underbrace{a(E,F) + \gamma \cdot b(E,F)}_{(*)} \big\}
\end{aligned}$$

Hence, the total score $(*)$ for each candidate translation $E \in \mathcal{C}$ can be described as a line with γ as the independent variable. For any particular choice of γ, the decoder seeks that translation which yields the largest score and therefore corresponds to the topmost line segment. If γ is shifted from $-\infty$ to $+\infty$, other translation hypotheses may at some point constitute the topmost line segments and thus change the decision made by the decoder. The entire sequence of topmost line segments is called *upper envelope* and provides an exhaustive representation of *all* possible outcomes that the decoder may yield if γ is shifted along the chosen direction. Both the translations and their corresponding line segments can efficiently be computed without incorporating any error criterion. Once the envelope has been determined, the translation candidates of its constituent line segments are projected onto their corresponding error counts, thus yielding the *exact and unsmoothed* error surface for all candidate translations encoded in $\mathcal{C}$. The error surface can now easily be traversed in order to find that $\hat{\gamma}$ under which the new parameter set $\lambda_1^M + \hat{\gamma} \cdot d_1^M$ minimizes the global error.

In this section, we present an extension of the algorithm described in Macherey et al. (2008) that allows us to efficiently compute and represent upper envelopes over all candidate translations encoded in hypergraphs. Conceptually, the algorithm works by propagating (initially empty) envelopes from the hypergraph's source nodes bottom-up to its unique root node, thereby expanding the envelopes by applying SCFG rules to the partial candidate translations that are associated with the envelope's constituent line segments. To recombine envelopes, we need two operators: the *sum* and the *maximum* over convex polygons. To illustrate which operator is applied when, we transform $\mathcal{H} = \langle \mathcal{V}, \mathcal{E} \rangle$ into a regular graph with typed nodes by (1) marking all vertices $v \in \mathcal{V}$ with the symbol $\vee$ and (2) replacing each hyperedge $e \in \mathcal{E}$, $|e| > 1$, with a small subgraph consisting of a new vertex $v_\wedge(e)$ whose incoming and outgoing edges connect the same head and tail nodes

Algorithm 1 $\wedge$-operation (Sum)

input: associative map a: $\mathcal{V} \rightarrow \text{Env}(\mathcal{V})$, hyperarc e
output: Minkowski sum of envelopes over $T(e)$

```
for (i = 0; i < |T(e)|; ++i) {
  v = T_i(e);
  pq.enqueue(⟨v, i, 0⟩);
}

L = ∅;
D = ⟨e, ε_1 ··· ε_|e|⟩
while (!pq.empty()) {
  ⟨v, i, j⟩ = pq.dequeue();
  ℓ = A[v][j];
  D[i+1] = ℓ.D;
  if (L.empty() ∨ L.back().x < ℓ.x) {
    if (0 < j)  {
      ℓ.y += L.back().y - A[v][j-1].y;
      ℓ.m += L.back().m - A[v][j-1].m;
    }
    L.push_back(ℓ);
    L.back().D = D;
  } else {
    L.back().y += ℓ.y;
    L.back().m += ℓ.m;
    L.back().D[i+1] = ℓ.D;
    if (0 < j)  {
      L.back().y -= A[v][j-1].y;
      L.back().m -= A[v][j-1].m;
    }
  }
  if (++j < A[v].size())
    pq.enqueue(⟨v, i, j⟩);
}
return L;
```

Algorithm 2 $\vee$-operation (Max)

input: array L[0..K-1] containing line objects
output: upper envelope of L

```
Sort(L:m);
j = 0; K = size(L);
for (i = 0; i < K; ++i) {
  ℓ = L[i];
  ℓ.x = -∞;
  if (0 < j) {
    if (L[j-1].m == ℓ.m) {
      if (ℓ.y <= L[j-1].y) continue;
      --j;
    }
    while (0 < j) {
      ℓ.x = (ℓ.y - L[j-1].y)/
            (L[j-1].m - ℓ.m);
      if (L[j-1].x < ℓ.x) break;
      --j;
    }
    if (0 == j) ℓ.x = -∞;
    L[j++] = ℓ;
  } else L[j++] = ℓ;
}
L.resize(j);
return L;
```

in the transformed graph as were connected by e in the original graph. The unique outgoing edge of $v_\wedge(e)$ is associated with the rule r_e; incoming edges are not linked to any rule. Figure 2 illustrates the transformation for a hyperedge with arity 3. The graph transformation is isomorphic.

The rules associated with every hyperedge specify how line segments in the envelopes of a hyperedge's tail nodes can be combined. Suppose we have a hyperedge e with rule $r_e : X \rightarrow aX_1bX_2c$ and $T(e) = \{v_1, v_2\}$. Then we substitute X_1 and X_2 in the rule with candidate translations associated with line segments in envelopes $\mathsf{Env}(v_1)$ and $\mathsf{Env}(v_2)$ respectively.

To derive the algorithm, we consider the general case of a hyperedge e with rule $r_e : \quad X \rightarrow w_1X_1w_2...w_nX_nw_{n+1}$. Because the right-hand side of r_e has n nonterminals, the arity of e is $|e| = n$. Let $T(e) = \{v_1, ..., v_n\}$ denote the tail nodes of e. We now assume that each tail node $v_i \in T(e)$ is associated with the upper envelope over all candidate translations that are induced by derivations of the corresponding nonterminal symbol X_i. These envelopes shall be denoted by $\mathsf{Env}(v_i)$. To decompose the problem of computing and propagating the tail envelopes over the hyperedge e to its head node, we now define two operations, one for either node type, to specify how envelopes associated with the tail vertices are propagated to the head vertex.

Nodes of Type "$\wedge$": For a type $\wedge$ node, the resulting envelope is the *Minkowski sum* over the envelopes of the incoming edges (Berg et al., 2008). Since the envelopes of the incoming edges are convex hulls, the Minkowski sum provides an upper bound to the number of line segments that constitute the resulting envelope: the bound is the sum over the number of line segments in the envelopes of the incoming edges, i.e.: $\left|\mathsf{Env}(v_\wedge(e))\right| \leq \sum_{v_\vee \in T(e)} \left|\mathsf{Env}(v_\vee)\right|$.

Algorithm 1 shows the pseudo code for computing the Minkowski sum over multiple envelopes. The line objects ℓ used in this algorithm are encoded as 4-tuples, each consisting of the x-intercept with ℓ's left-adjacent line stored as $\ell.x$, the slope $\ell.m$, the y-intercept $\ell.y$, and the (partial) derivation tree $\ell.D$. At the beginning, the leftmost line segment of each envelope is inserted into a priority queue pq. The priority is defined in terms of a line's x-intercept such that lower values imply higher priority. Hence, the priority queue enumerates all line segments from left to right in ascending order of their x-intercepts, which is the order needed to compute the Minkowski sum.

Nodes of Type "$\vee$": The operation performed

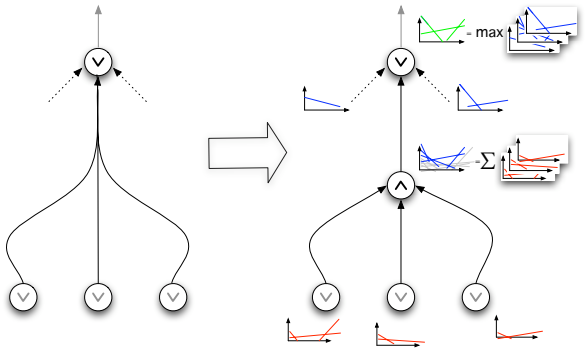


Figure 2: Transformation of a hypergraph into a factor graph and bottom-up propagation of envelopes.

at nodes of type "$\vee$" computes the convex hull over the union of the envelopes propagated over the incoming edges. This operation is a "max" operation and it is identical to the algorithm described in (Macherey et al., 2008) for phrase lattices. Algorithm 2 contains the pseudo code.

The complete algorithm then works as follows: Traversing all nodes in $\mathcal{H}$ bottom-up in topological order, we proceed for each node $v \in \mathcal{V}$ over its incoming hyperedges and combine in each such hyperedge e the envelopes associated with the tail nodes $T(e)$ by computing their sum according to Algorithm 1 ($\wedge$-operation). For each incoming hyperedge e, the resulting envelope is then expanded by applying the rule r_e to its constituent line segments. The envelopes associated with different incoming hyperedges of node v are then combined and reduced according to Algorithm 2 ($\vee$-operation). By construction, the envelope at the root node is the convex hull over the line segments of all candidate translations that can be derived from the hypergraph.

The suggested algorithm has similar properties as the algorithm presented in (Macherey et al., 2008). In particular, it has the same upper bound on the number of line segments that constitute the envelope at the root node, i.e, the size of this envelope is guaranteed to be no larger than the number of edges in the transformed hypergraph.

4 Minimum Bayes-Risk Decoding

We first review Minimum Bayes-Risk (MBR) decoding for statistical MT. An MBR decoder seeks the hypothesis with the least expected loss under a probability model (Bickel and Doksum, 1977). If we think of statistical MT as a classifier that maps a source sentence F to a target sentence E, the MBR decoder can be expressed as follows:

$$\hat{E} = \underset{E' \in \mathcal{G}}{\operatorname{argmin}} \sum_{E \in \mathcal{G}} L(E, E') P(E|F), \qquad (1)$$

where $L(E, E')$ is the loss between any two hypotheses E and E', $P(E|F)$ is the probability model, and $\mathcal{G}$ is the space of translations (N-best list, lattice, or a hypergraph).

MBR decoding for translation can be performed by reranking an N-best list of hypotheses generated by an MT system (Kumar and Byrne, 2004). This reranking can be done for any sentence-level loss function such as BLEU (Papineni et al., 2001), *Word Error Rate*, or *Position-independent Error Rate*.

Recently, Tromble et al. (2008) extended MBR decoding to translation lattices under an approximate BLEU score. They approximated log(BLEU) score by a linear function of n-gram matches and candidate length. If E and E' are the reference and the candidate translations respectively, this linear function is given by:

$$G(E, E') = \theta_0 |E'| + \sum_w \theta_{|w|} \#_w(E') \delta_w(E), \quad (2)$$

where w is an n-gram present in either E or E', and $\theta_0, \theta_1, ..., \theta_N$ are weights which are determined empirically, where N is the maximum n-gram order.

Under such a linear decomposition, the MBR decoder (Equation 1) can be written as

$$\hat{E} = \underset{E' \in \mathcal{G}}{\operatorname{argmax}} \, \theta_0 |E'| + \sum_w \theta_{|w|} \#_w(E') p(w|\mathcal{G}), \quad (3)$$

where the posterior probability of an n-gram in the lattice is given by

$$p(w|\mathcal{G}) = \sum_{E \in \mathcal{G}} 1_w(E) P(E|F). \qquad (4)$$

Tromble et al. (2008) implement the MBR decoder using Weighted Finite State Automata (WFSA) operations. First, the set of n-grams is extracted from the lattice. Next, the posterior probability of each n-gram is computed. A new automaton is then created by intersecting each n-gram with weight (from Equation 2) to an unweighted lattice. Finally, the MBR hypothesis is extracted as the best path in the automaton. We will refer to this procedure as FSAMBR.

The above steps are carried out one n-gram at a time. For a moderately large lattice, there can be several thousands of n-grams and the procedure becomes expensive. We now present an alternate approximate procedure which can avoid this

enumeration making the resulting algorithm much faster than FSAMBR.

4.1 Efficient MBR for lattices

The key idea behind this new algorithm is to rewrite the n-gram posterior probability (Equation 4) as follows:

$$p(w|\mathcal{G}) = \sum_{E \in \mathcal{G}} \sum_{e \in E} f(e, w, E) P(E|F) \quad (5)$$

where $f(e, w, E)$ is a score assigned to edge e on path E containing n-gram w:

$$f(e, w, E) = \begin{cases} 1 & w \in e, p(e|\mathcal{G}) > p(e'|\mathcal{G}), \\ & e' \text{ precedes } e \text{ on } E \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

In other words, for each path E, we count the edge that contributes n-gram w and has the highest edge posterior probability relative to its predecessors on the path E; there is exactly one such edge on each lattice path E.

We note that $f(e, w, E)$ relies on the full path E which means that it cannot be computed based on local statistics. We therefore approximate the quantity $f(e, w, E)$ with $f^*(e, w, \mathcal{G})$ that counts the edge e with n-gram w that has the highest arc posterior probability relative to predecessors in the entire lattice $\mathcal{G}$. $f^*(e, w, \mathcal{G})$ can be computed locally, and the n-gram posterior probability based on f^* can be determined as follows:

$$\begin{aligned} p(w|\mathcal{G}) &= \sum_{E \in \mathcal{G}} \sum_{e \in E} f^*(e, w, \mathcal{G}) P(E|F) \quad (7) \\ &= \sum_{e \in \mathcal{E}} 1_{w \in e} f^*(e, w, \mathcal{G}) \sum_{E \in \mathcal{G}} 1_E(e) P(E|F) \\ &= \sum_{e \in \mathcal{E}} 1_{w \in e} f^*(e, w, \mathcal{G}) P(e|\mathcal{G}), \end{aligned}$$

where $P(e|\mathcal{G})$ is the posterior probability of a lattice edge. The algorithm to perform Lattice MBR is given in Algorithm 3. For each node t in the lattice, we maintain a quantity Score(w, t) for each n-gram w that lies on a path from the source node to t. Score(w, t) is the highest posterior probability among all edges on the paths that terminate on t and contain n-gram w. The forward pass requires computing the n-grams introduced by each edge; to do this, we propagate n-grams (up to maximum order -1) terminating on each node.

4.2 Extension to Hypergraphs

We next extend the Lattice MBR decoding algorithm (Algorithm 3) to rescore hypergraphs produced by a SCFG based MT system. Algorithm 4 is an extension to the MBR decoder on lattices

Algorithm 3 MBR Decoding on Lattices

1: Sort the lattice nodes topologically.
2: Compute backward probabilities of each node.
3: Compute posterior prob. of each n-gram:
4: **for** each edge e **do**
5: Compute edge posterior probability $P(e|\mathcal{G})$.
6: Compute n-gram posterior probs. $P(w|\mathcal{G})$:
7: **for** each n-gram w introduced by e **do**
8: Propagate $n-1$ gram suffix to h_e.
9: **if** $p(e|\mathcal{G}) >$ Score$(w, T(e))$ **then**
10: Update posterior probs. and scores: $p(w|\mathcal{G})$ += $p(e|\mathcal{G})$ − Score$(w, T(e))$. Score$(w, h_e) = p(e|\mathcal{G})$.
11: **else**
12: Score$(w, h_e) =$ Score$(w, T(e))$.
13: **end if**
14: **end for**
15: **end for**
16: Assign scores to edges (given by Equation 3).
17: Find best path in the lattice (Equation 3).

(Algorithm 3). However, there are important differences when computing the n-gram posterior probabilities (Step 3). In this inside pass, we now maintain both n-gram prefixes and suffixes (up to the maximum order -1) on each hypergraph node. This is necessary because unlike a lattice, new n-grams may be created at subsequent nodes by concatenating words both to the left and the right side of the n-gram. When the arity of the edge is 2, a rule has the general form aX_1bX_2c, where X_1 and X_2 are sequences from tail nodes. As a result, we need to consider all new sequences which can be created by the cross-product of the n-grams on the two tail nodes. E.g. if $X_1 = \{c, cd, d\}$ and $X_2 = \{f, g\}$, then a total of six sequences will result. In practice, such a cross-product is not pro-

Algorithm 4 MBR Decoding on Hypergraphs

1: Sort the hypergraph nodes topologically.
2: Compute inside probabilities of each node.
3: Compute posterior prob. of each hyperedge $P(e|\mathcal{G})$.
4: Compute posterior prob. of each n-gram:
5: **for** each hyperedge e **do**
6: Merge the n-grams on the tail nodes $T(e)$. If the same n-gram is present on multiple tail nodes, keep the highest score.
7: Apply the rule on e to the n-grams on $T(e)$.
8: Propagate $n-1$ gram prefixes/suffixes to h_e.
9: **for** each n-gram w introduced by this hyperedge **do**
10: **if** $p(e|\mathcal{G}) >$ Score$(w, T(e))$ **then**
11: $p(w|\mathcal{G})$ += $p(e|\mathcal{G})$ − Score$(w, T(e))$ Score$(w, h_e) = p(e|\mathcal{G})$
12: **else**
13: Score$(w, h_e) =$ Score$(w, T(e))$
14: **end if**
15: **end for**
16: **end for**
17: Assign scores to hyperedges (Equation 3).
18: Find best path in the hypergraph (Equation 3).

hibitive when the maximum n-gram order in MBR does not exceed the order of the n-gram language model used in creating the hypergraph. In the latter case, we will have a small set of unique prefixes and suffixes on the tail nodes.

5 MERT for MBR Parameter Optimization

Lattice MBR Decoding (Equation 3) assumes a linear form for the gain function (Equation 2). This linear function contains $n+1$ parameters $\theta_0, \theta_1, ..., \theta_N$, where N is the maximum order of the n-grams involved. Tromble et al. (2008) obtained these factors as a function of n-gram precisions derived from multiple training runs. However, this does not guarantee that the resulting linear score (Equation 2) is close to the corpus BLEU. We now describe how MERT can be used to estimate these factors to achieve a better approximation to the corpus BLEU.

We recall that MERT selects weights in a linear model to optimize an error criterion (e.g. corpus BLEU) on a training set. The lattice MBR decoder (Equation 3) can be written as a linear model: $\hat{E} = \text{argmax}_{E' \in \mathcal{G}} \sum_{i=0}^{N} \theta_i g_i(E', F)$, where $g_0(E', F) = |E'|$ and $g_i(E', F) = \sum_{w:|w|=i} \#_w(E') p(w|\mathcal{G})$.

The linear approximation to BLEU may not hold in practice for unseen test sets or language-pairs. Therefore, we would like to allow the decoder to backoff to the MAP translation in such cases. To do that, we introduce an additional feature function $g_{N+1}(E, F)$ equal to the original decoder cost for this sentence. A weight assignment of 1.0 for this feature function and zeros for the other feature functions would imply that the MAP translation is chosen. We now have a total of $N+2$ feature functions which we optimize using MERT to obtain highest BLEU score on a training set.

6 Experiments

We now describe our experiments to evaluate MERT and MBR on lattices and hypergraphs, and show how MERT can be used to tune MBR parameters.

6.1 Translation Tasks

We report results on two tasks. The first one is the constrained data track of the NIST Arabic-to-English (aren) and Chinese-to-English (zhen) translation task[1]. On this task, the parallel and the monolingual data included all the allowed training sets for the constrained track. Table 1 reports statistics computed over these data sets. Our development set (*dev*) consists of the NIST 2005 eval set; we use this set for optimizing MBR parameters. We report results on NIST 2002 and NIST 2003 evaluation sets.

Dataset	# of sentences	
	aren	zhen
dev	1797	1664
nist02	1043	878
nist03	663	919

Table 1: Statistics over the NIST dev/test sets.

The second task consists of systems for 39 language-pairs with English as the target language and trained on at most $300M$ word tokens mined from the web and other published sources. The development and test sets for this task are randomly selected sentences from the web, and contain 5000 and 1000 sentences respectively.

6.2 MT System Description

Our phrase-based statistical MT system is similar to the alignment template system described in (Och and Ney, 2004; Tromble et al., 2008). Translation is performed using a standard dynamic programming beam-search decoder (Och and Ney, 2004) using two decoding passes. The first decoder pass generates either a lattice or an N-best list. MBR decoding is performed in the second pass.

We also train two SCFG-based MT systems: a hierarchical phrase-based SMT (Chiang, 2007) system and a *syntax augmented machine translation* (SAMT) system using the approach described in Zollmann and Venugopal (2006). Both systems are built on top of our phrase-based systems. In these systems, the decoder generates an initial hypergraph or an N-best list, which are then rescored using MBR decoding.

6.3 MERT Results

Table 2 shows runtime experiments for the hypergraph MERT implementation in comparison with the phrase-lattice implementation on both the aren and the zhen system. The first two columns show the average amount of time in msecs that either algorithm requires to compute the upper envelope when applied to phrase lattices. Compared to the algorithm described in (Macherey et al., 2008) which is optimized for phrase lattices, the hypergraph implementation causes a small increase in

[1] http://www.nist.gov/speech/tests/mt

	Avg. Runtime/sent [msec]			
	(Macherey 2008)		Suggested Alg.	
	aren	zhen	aren	zhen
phrase lattice	8.57	7.91	10.30	8.65
hypergraph	–	–	8.19	8.11

Table 2: Average time for computing envelopes.

running time. This increase is mainly due to the representation of line segments; while the phrase-lattice implementation stores a single backpointer, the hypergraph version stores a vector of backpointers.

The last two columns show the average amount of time that is required to compute the upper envelope on hypergraphs. For comparison, we prune hypergraphs to the same density (# of edges per edge on the best path) and achieve identical running times for computing the error surface.

6.4 MBR Results

We first compare the new lattice MBR (Algorithm 3) with MBR decoding on 1000-best lists and FSAMBR (Tromble et al., 2008) on lattices generated by the phrase-based systems; evaluation is done using both BLEU and average run-time per sentence (Table 3). Note that N-best MBR uses a sentence BLEU loss function. The new lattice MBR algorithm gives about the same performance as FSAMBR while yielding a 20X speedup.

We next report the performance of MBR on hypergraphs generated by Hiero/SAMT systems. Table 4 compares Hypergraph MBR (HGMBR) with MAP and MBR decoding on 1000 best lists. On some systems such as the Arabic-English SAMT, the gains from Hypergraph MBR over 1000-best MBR are significant. In other cases, Hypergraph MBR performs at least as well as N-best MBR. In all cases, we observe a 7X speedup in runtime. This shows the usefulness of Hypergraph MBR decoding as an efficient alternative to N-best MBR.

6.5 MBR Parameter Tuning with MERT

We now describe the results by tuning MBR n-gram parameters (Equation 2) using MERT. We first compute $N + 1$ MBR feature functions on each edge of the lattice/hypergraph. We also include the total decoder cost on the edge as as additional feature function. MERT is then performed to optimize the BLEU score on a development set; For MERT, we use 40 random initial parameters as well as parameters computed using corpus based statistics (Tromble et al., 2008).

	BLEU (%)				Avg.
	aren		zhen		time
	nist03	nist02	nist03	nist02	(ms.)
MAP	54.2	64.2	40.1	39.0	-
N-best MBR	54.3	64.5	40.2	39.2	3.7
Lattice MBR					
FSAMBR	54.9	65.2	40.6	39.5	3.7
LatMBR	54.8	65.2	40.7	39.4	0.2

Table 3: Lattice MBR for a phrase-based system.

	BLEU (%)				Avg.
	aren		zhen		time
	nist03	nist02	nist03	nist02	(ms.)
Hiero					
MAP	52.8	62.9	41.0	39.8	-
N-best MBR	53.2	63.0	41.0	40.1	3.7
HGMBR	53.3	63.1	41.0	40.2	0.5
SAMT					
MAP	53.4	63.9	41.3	40.3	-
N-best MBR	53.8	64.3	41.7	41.1	3.7
HGMBR	54.0	64.6	41.8	41.1	0.5

Table 4: Hypergraph MBR for Hiero/SAMT systems.

Table 5 shows results for NIST systems. We report results on nist03 set and present three systems for each language pair: phrase-based (pb), hierarchical (hier), and SAMT; Lattice MBR is done for the phrase-based system while HGMBR is used for the other two. We select the MBR scaling factor (Tromble et al., 2008) based on the development set; it is set to $0.1, 0.01, 0.5, 0.2, 0.5$ and 1.0 for the aren-phrase, aren-hier, aren-samt, zhen-phrase zhen-hier and zhen-samt systems respectively. For the multi-language case, we train phrase-based systems and perform lattice MBR for all language pairs. We use a scaling factor of 0.7 for all pairs. Additional gains can be obtained by tuning this factor; however, we do not explore that dimension in this paper. In all cases, we prune the lattices/hypergraphs to a density of 30 using *forward-backward pruning* (Sixtus and Ortmanns, 1999).

We consider a BLEU score difference to be a) gain if is at least 0.2 points, b) drop if it is at most -0.2 points, and c) no change otherwise. The results are shown in Table 6. In both tables, the following results are reported: Lattice/HGMBR with default parameters $(-5, 1.5, 2, 3, 4)$ computed using corpus statistics (Tromble et al., 2008), Lattice/HGMBR with parameters derived from MERT both without/with the baseline model cost feature (mert−b, mert+b). For multi-language systems, we only show the # of language-pairs with gains/no-changes/drops for each MBR variant with respect to the MAP translation.

We observed in the NIST systems that MERT resulted in short translations relative to MAP on the unseen test set. To prevent this behavior, we modify the MERT error criterion to include a sentence-level brevity scorer with parameter α: BLEU+brevity(α). This brevity scorer penalizes each candidate translation that is shorter than the average length over its reference translations, using a penalty term which is linear in the difference between either length. We tune α on the development set so that the brevity score of MBR translation is close to that of the MAP translation.

In the NIST systems, MERT yields small improvements on top of MBR with default parameters. This is the case for Arabic-English Hiero/SAMT. In all other cases, we see no change or even a slight degradation due to MERT. We hypothesize that the default MBR parameters (Tromble et al., 2008) are well tuned. Therefore there is little gain by additional tuning using MERT.

In the multi-language systems, the results show a different trend. We observe that MBR with default parameters results in gains on 18 pairs, no differences on 9 pairs, and losses on 12 pairs. When we optimize MBR features with MERT, the number of language pairs with gains/no changes/-drops is 22/5/12. Thus, MERT has a bigger impact here than in the NIST systems. We hypothesize that the default MBR parameters are sub-optimal for some language pairs and that MERT helps to find better parameter settings. In particular, MERT avoids the need for manually tuning these parameters by language pair.

Finally, when baseline model costs are added as an extra feature (mert+b), the number of pairs with gains/no changes/drops is 26/8/5. This shows that this feature can allow MBR decoding to back-off to the MAP translation. When MBR does not produce a higher BLEU score relative to MAP on the development set, MERT assigns a higher weight to this feature function. We see such an effect for 4 systems.

System	BLEU (%)			
	MAP	MBR		
		default	mert-b	mert+b
aren.pb	54.2	54.8	54.8	54.9
aren.hier	52.8	53.3	53.5	53.7
aren.samt	53.4	54.0	54.4	54.0
zhen.pb	40.1	40.7	40.7	40.9
zhen.hier	41.0	41.0	41.0	41.0
zhen.samt	41.3	41.8	41.6	41.7

Table 5: MBR Parameter Tuning on NIST systems

MBR wrt. MAP	default	mert-b	mert+b
# of gains	18	22	26
# of no-changes	9	5	8
# of drops	12	12	5

Table 6: MBR on Multi-language systems.

7 Discussion

We have presented efficient algorithms which extend previous work on lattice-based MERT (Macherey et al., 2008) and MBR decoding (Tromble et al., 2008) to work with hypergraphs. Our new MERT algorithm can work with both lattices and hypergraphs. On lattices, it achieves similar run-times as the implementation described in Macherey et al. (2008). The new Lattice MBR decoder achieves a 20X speedup relative to either FSAMBR implementation described in Tromble et al. (2008) or MBR on 1000-best lists. The algorithm gives comparable results relative to FSAMBR. On hypergraphs produced by Hierarchical and Syntax Augmented MT systems, our MBR algorithm gives a 7X speedup relative to 1000-best MBR while giving comparable or even better performance.

Lattice MBR decoding is obtained under a linear approximation to BLEU, where the weights are obtained using n-gram precisions derived from development data. This may not be optimal in practice for unseen test sets and language pairs, and the resulting linear loss may be quite different from the corpus level BLEU. In this paper, we have described how MERT can be employed to estimate the weights for the linear loss function to maximize BLEU on a development set. On an experiment with 40 language pairs, we obtain improvements on 26 pairs, no difference on 8 pairs and drops on 5 pairs. This was achieved without any need for manual tuning for each language pair. The baseline model cost feature helps the algorithm effectively back off to the MAP translation in language pairs where MBR features alone would not have helped.

MERT and MBR decoding are popular techniques for incorporating the final evaluation metric into the development of SMT systems. We believe that our efficient algorithms will make them more widely applicable in both SCFG-based and phrase-based MT systems.

References

M. Berg, O. Cheong, M. Krefeld, and M. Overmars, 2008. *Computational Geometry: Algorithms and Applications*, chapter 13, pages 290–296. Springer-Verlag, 3rd edition.

P. J. Bickel and K. A. Doksum. 1977. *Mathematical Statistics: Basic Ideas and Selected topics*. Holden-Day Inc., Oakland, CA, USA.

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

M. Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang, and I. Thayer. 2006. Scalable Inference and Training of Context-Rich Syntactic Translation Models. . In *COLING/ACL*, Sydney, Australia.

L. Huang. 2008. Advanced Dynamic Programming in Semiring and Hypergraph Frameworks. In *COLING*, Manchester, UK.

S. Kumar and W. Byrne. 2004. Minimum Bayes-Risk Decoding for Statistical Machine Translation. In *HLT-NAACL*, Boston, MA, USA.

W. Macherey, F. Och, I. Thayer, and J. Uszkoreit. 2008. Lattice-based Minimum Error Rate Training for Statistical Machine Translation. In *EMNLP*, Honolulu, Hawaii, USA.

H. Mi, L. Huang, and Q. Liu. 2008. Forest-Based Translation. In *ACL*, Columbus, OH, USA.

F. Och and H. Ney. 2004. The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics*, 30(4):417 – 449.

F. Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *ACL*, Sapporo, Japan.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2001. Bleu: a Method for Automatic Evaluation of Machine Translation. Technical Report RC22176 (W0109-022), IBM Research Division.

A. Sixtus and S. Ortmanns. 1999. High Quality Word Graphs Using Forward-Backward Pruning. In *ICASSP*, Phoenix, AZ, USA.

R. Tromble, S. Kumar, F. Och, and W. Macherey. 2008. Lattice Minimum Bayes-Risk Decoding for Statistical Machine Translation. In *EMNLP*, Honolulu, Hawaii.

H. Zhang and D. Gildea. 2008. Efficient Multi-pass Decoding for Synchronous Context Free Grammars. In *ACL*, Columbus, OH, USA.

A. Zollmann and A. Venugopal. 2006. Syntax Augmented Machine Translation via Chart Parsing. In *HLT-NAACL*, New York, NY, USA.

Forest-based Tree Sequence to String Translation Model

Hui Zhang[1, 2] **Min Zhang**[1] **Haizhou Li**[1] **Aiti Aw**[1] **Chew Lim Tan**[2]
[1]Institute for Infocomm Research [2]National University of Singapore
zhangh1982@gmail.com {mzhang, hli, aaiti}@i2r.a-star.edu.sg tancl@comp.nus.edu.sg

Abstract

This paper proposes a forest-based tree sequence to string translation model for syntax-based statistical machine translation, which automatically learns tree sequence to string translation rules from word-aligned source-side-parsed bilingual texts. The proposed model leverages on the strengths of both tree sequence-based and forest-based translation models. Therefore, it can not only utilize forest structure that compactly encodes exponential number of parse trees but also capture non-syntactic translation equivalences with linguistically structured information through tree sequence. This makes our model potentially more robust to parse errors and structure divergence. Experimental results on the NIST MT-2003 Chinese-English translation task show that our method statistically significantly outperforms the four baseline systems.

1 Introduction

Recently syntax-based statistical machine translation (SMT) methods have achieved very promising results and attracted more and more interests in the SMT research community. Fundamentally, syntax-based SMT views translation as a structural transformation process. Therefore, structure divergence and parse errors are two of the major issues that may largely compromise the performance of syntax-based SMT (Zhang et al., 2008a; Mi et al., 2008).

Many solutions have been proposed to address the above two issues. Among these advances, forest-based modeling (Mi et al., 2008; Mi and Huang, 2008) and tree sequence-based modeling (Liu et al., 2007; Zhang et al., 2008a) are two interesting modeling methods with promising results reported. Forest-based modeling aims to improve translation accuracy through digging the potential better parses from *n*-bests (i.e. forest) while tree sequence-based modeling aims to model non-syntactic translations with structured syntactic knowledge. In nature, the two methods would be complementary to each other since they manage to solve the negative impacts of monolingual parse errors and cross-lingual structure divergence on translation results from different viewpoints. Therefore, one natural way is to combine the strengths of the two modeling methods for better performance of syntax-based SMT. However, there are many challenges in combining the two methods into a single model from both theoretical and implementation engineering viewpoints. In theory, one may worry about whether the advantage of tree sequence has already been covered by forest because forest encodes implicitly a huge number of parse trees and these parse trees may generate many different phrases and structure segmentations given a source sentence. In system implementation, the exponential combinations of tree sequences with forest structures make the rule extraction and decoding tasks much more complicated than that of the two individual methods.

In this paper, we propose a forest-based tree sequence to string model, which is designed to integrate the strengths of the forest-based and the tree sequence-based modeling methods. We present our solutions that are able to extract translation rules and decode translation results for our model very efficiently. A general, configurable platform was designed for our model. With this platform, we can easily implement our method and many previous syntax-based methods by simple parameter setting. We evaluate our method on the NIST MT-2003 Chinese-English translation tasks. Experimental results show that our method significantly outperforms the two individual methods and other baseline methods. Our study shows that the proposed method is able to effectively combine the strengths of the forest-based and tree sequence-based methods, and thus having great potential to address the issues of parse errors and non-syntactic transla-

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 172–180,
Suntec, Singapore, 2-7 August 2009.

tions resulting from structure divergence. It also indicates that tree sequence and forest play different roles and make contributions to our model in different ways.

The remainder of the paper is organized as follows. Section 2 describes related work while section 3 defines our translation model. In section 4 and section 5, the key rule extraction and decoding algorithms are elaborated. Experimental results are reported in section 6 and the paper is concluded in section 7.

2 Related work

As discussed in section 1, two of the major challenges to syntax-based SMT are structure divergence and parse errors. Many techniques have been proposed to address the structure divergence issue while only fewer studies are reported in addressing the parse errors in the SMT research community.

To address structure divergence issue, many researchers (Eisner, 2003; Zhang et al., 2007) propose using the Synchronous Tree Substitution Grammar (STSG) grammar in syntax-based SMT since the STSG uses larger tree fragment as translation unit. Although promising results have been reported, STSG only uses one single sub-tree as translation unit which is still committed to the syntax strictly. Motivated by the fact that non-syntactic phrases make non-trivial contribution to phrase-based SMT, the tree sequence-based translation model is proposed (Liu et al., 2007; Zhang et al., 2008a) that uses tree sequence as the basic translation unit, rather than using single sub-tree as in the STSG. Here, a tree sequence refers to a sequence of consecutive sub-trees that are embedded in a full parse tree. For any given phrase in a sentence, there is at least one tree sequence covering it. Thus the tree sequence-based model has great potential to address the structure divergence issue by using tree sequence-based non-syntactic translation rules. Liu et al. (2007) propose the tree sequence concept and design a tree sequence to string translation model. Zhang et al. (2008a) propose a tree sequence-based tree to tree translation model and Zhang et al. (2008b) demonstrate that the tree sequence-based modelling method can well address the structure divergence issue for syntax-based SMT.

To overcome the parse errors for SMT, Mi et al. (2008) propose a forest-based translation method that uses forest instead of one best tree as translation input, where a forest is a compact representation of exponentially number of n-best parse trees. Mi and Huang (2008) propose a forest-based rule extraction algorithm, which learn tree to string rules from source forest and target string. By using forest in rule extraction and decoding, their methods are able to well address the parse error issue.

From the above discussion, we can see that traditional tree sequence-based method uses single tree as translation input while the forest-based model uses single sub-tree as the basic translation unit that can only learn tree-to-string (Galley et al. 2004; Liu et al., 2006) rules. Therefore, the two methods display different strengths, and which would be complementary to each other. To integrate their strengths, in this paper, we propose a forest-based tree sequence to string translation model.

3 Forest-based tree sequence to string model

In this section, we first explain what a packed forest is and then define the concept of the tree sequence in the context of forest followed by the discussion on our proposed model.

3.1 Packed Forest

A packed forest (forest in short) is a special kind of hyper-graph (Klein and Manning, 2001; Huang and Chiang, 2005), which is used to represent all derivations (i.e. parse trees) for a given sentence under a context free grammar (CFG). A forest F is defined as a triple $< V, E, S >$, where V is non-terminal node set, E is hyper-edge set and S is leaf node set (i.e. all sentence words). A forest F satisfies the following two conditions:

1) Each node n in V should cover a phrase, which is a continuous word sub-sequence in S.

2) Each hyper-edge e in E is defined as $v_f \Rightarrow v_1 \dots v_i \dots v_n, (v_i \in (V \cup S), v_f \in V)$, where $v_1 \dots v_i \dots v_n$ covers a sequence of continuous and non-overlap phrases, v_f is the father node of the children sequence $v_1 \dots v_i \dots v_n$. The phrase covered by v_f is just the sum of all the phrases covered by each child node v_i.

We here introduce another concept that is used in our subsequent discussions. A complete forest *CF* is a general forest with one additional condition that there is only one root node *N* in *CF*, i.e., all nodes except the root *N* in a *CF* must have at least one father node.

Fig. 1 is a complete forest while Fig. 7 is a non-complete forest due to the virtual node "VV+VV" introduced in Fig. 7. Fig. 2 is a hyper-edge (IP => NP VP) of Fig. 1, where NP covers

the phrase “Xinhuashe”, VP covers the phrase “shengming youguan guiding” and IP covers the entire sentence. In Fig.1, only root IP has no father node, so it is a complete forest. The two parse trees T1 and T2 encoded in Fig. 1 are shown separately in Fig. 3 and Fig. 4[1].

Different parse tree represents different derivations and explanations for a given sentence. For example, for the same input sentence in Fig. 1, T1 interprets it as “XNA (Xinhua News Agency) declares some regulations.” while T2 interprets it as “XNA declaration is related to some regulations.”.

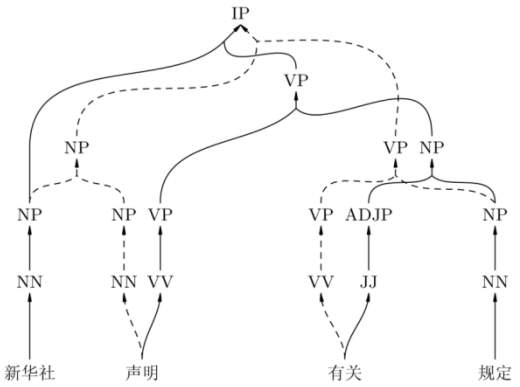


Figure 1. A packed forest for sentence “新华社 /Xinhuashe 声明/shengming 有关/youguan 规定 /guiding”

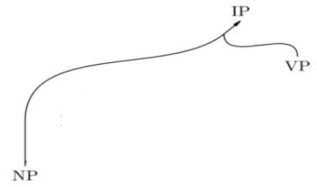


Figure 2. A hyper-edge used in Fig. 1

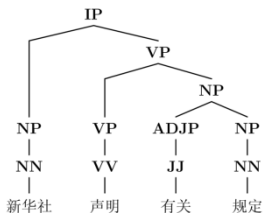


Figure 3. Tree 1 (T1)

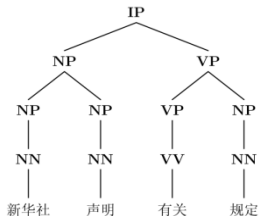


Figure 4. Tree 2 (T2)

3.2 Tree sequence in packed forest

Similar to the definition of tree sequence used in a single parse tree defined in Liu et al. (2007) and Zhang et al. (2008a), a tree sequence in a forest also refers to an ordered sub-tree sequence that covers a continuous phrase without overlapping. However, the major difference between them lies in that the sub-trees of a tree sequence in forest may belongs to different single parse trees while, in a single parse tree-based model, all the sub-trees in a tree sequence are committed to the same parse tree.

The forest-based tree sequence enables our model to have the potential of exploring additional parse trees that may be wrongly pruned out by the parser and thus are not encoded in the forest. This is because that a tree sequence in a forest allows its sub-trees coming from different parse trees, where these sub-trees may not be merged finally to form a complete parse tree in the forest. Take the forest in Fig. 1 as an example, where ((*VV shengming*) (*JJ youguan*)) is a tree sequence that all sub-trees appear in T1 while ((*VV shengming*) (*VV youguan*)) is a tree sequence whose sub-trees do not belong to any single tree in the forest. But, indeed the two sub-trees (*VV shengming*) and (*VV youguan*) can be merged together and further lead to a complete single parse tree which may offer a correct interpretation to the input sentence (as shown in Fig. 5). In addition, please note that, on the other hand, more parse trees may introduce more noisy structures. In this paper, we leave this problem to our model and let the model decide which sub-structures are noisy features.

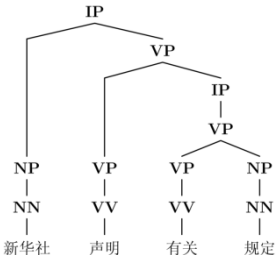


Figure 5. A parse tree that was wrongly pruned out

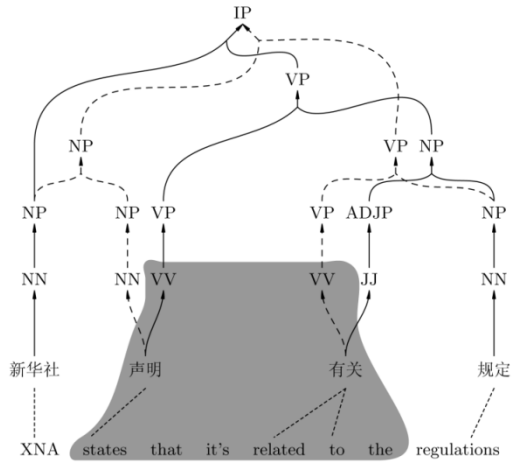


Figure 6. A tree sequence to string rule

[1] Please note that a single tree (as T1 and T2 shown in Fig. 3 and Fig. 4) is represented by edges instead of hyper-edges. A hyper-edge is a group of edges satisfying the 2nd condition as shown in the forest definition.

A tree-sequence to string translation rule in a forest is a triple <*L*, *R*, *A*>, where *L* is the tree sequence in source language, *R* is the string containing words and variables in target language, and *A* is the alignment between the leaf nodes of *L* and *R*. This definition is similar to that of (Liu et al. 2007, Zhang et al. 2008a) except our tree-sequence is defined in forest. The shaded area of Fig. 6 exemplifies a tree sequence to string translation rule in the forest.

3.3 Forest-based tree-sequence to string translation model

Given a source forest *F* and target translation T_S as well as word alignment A, our translation model is formulated as:

$$\Pr(F, T_s, A) = \sum_{\theta_i \in \Theta, C(\Theta)=(F,T_s,A)} \prod_{r_i \in \theta_i} p(r_i)$$

By the above Eq., translation becomes a tree sequence structure to string mapping issue. Given the *F*, T_S and A, there are multiple derivations that could map *F* to T_S under the constraint A. The mapping probability $\Pr(F, T_s, A)$ in our study is obtained by summing over the probabilities of all derivations Θ. The probability of each derivation θ_i is given as the product of the probabilities of all the rules $p(r_i)$ used in the derivation (here we assume that each rule is applied *independently* in a derivation).

Our model is implemented under log-linear framework (Och and Ney, 2002). We use seven basic features that are analogous to the commonly used features in phrase-based systems (Koehn, 2003): 1) bidirectional rule mapping probabilities, 2) bidirectional lexical rule translation probabilities, 3) target language model, 4) number of rules used and 5) number of target words. In addition, we define two new features: 1) number of leaf nodes in auxiliary rules (the auxiliary rule will be explained later in this paper) and 2) product of the probabilities of all hyper-edges of the tree sequences in forest.

4 Training

This section discusses how to extract our translation rules given a triple $< F, T_s, A >$. As we know, the traditional tree-to-string rules can be easily extracted from $< F, T_s, A >$ using the algorithm of Mi and Huang (2008)[2]. We would like to leverage on their algorithm in our study. Unfortunately, their algorithm is not directly applicable to our problem because tree rules have only one root while tree sequence rules have multiple roots. This makes the tree sequence rule extraction very complex due to its interaction with forest structure. To address this issue, we introduce the concepts of virtual node and virtual hyper-edge to convert a complete parse forest *F* to a non-complete forest *F* which is designed to encode all the tree sequences that we want. Therefore, by doing so, the tree sequence rules can be extracted from a forest in the following two steps:

1) Convert the complete parse forest *F* into a non-complete forest *F* in order to cover those tree sequences that cannot be covered by a single tree node.

2) Employ the forest-based tree rule extraction algorithm (Mi and Huang, 2008) to extract our rules from the non-complete forest.

To facilitate our discussion, here we introduce two notations:

- Alignable: A consecutive source phrase is an alignable phrase if and only if it can be aligned with at least one consecutive target phrase under the word-alignment constraint. The covered source span is called alignable span.
- Node sequence: a sequence of nodes (either leaf or internal nodes) in a forest covering a consecutive span.

Algorithm 1 illustrates the first step of our rule extraction algorithm, which is a CKY-style Dynamic Programming (DP) algorithm to add virtual nodes into forest. It includes the following steps:

1) We traverse the forest to visit each span in bottom-up fashion (line 1-2),
   1.1) for each span [*u*,*v*] that is covered by single tree nodes[3], we put these tree nodes into the set *NSS*(*u*,*v*) and go back to step 1 (line 4-6).
   1.2) otherwise we concatenate the tree sequences of sub-spans to generate the set of tree sequences covering the current larger span (line 8-13). Then, we prune the set of node sequences (line 14). If this span is alignable, we create virtual father nodes and corresponding virtual hyper-edges to link the node sequences with the virtual father nodes (line 15-20).

[2] Mi and Huang (2008) extend the tree-based rule extraction algorithm (Galley et al., 2004) to forest-based by introducing non-deterministic mechanism. Their algorithm consists of two steps, minimal rule extraction and composed rule generation.

[3] Note that in a forest, there would be multiple single tree nodes covering the same span as shown Fig.1.

2) Finally we obtain a forest with each alignable span covered by either original tree nodes or the newly-created tree sequence virtual nodes.

Theoretically, there is exponential number of node sequences in a forest. Take Fig. 7 as an example. The *NSS* of span [1,2] only contains "NP" since it is alignable and covered by the single tree node NP. However, span [2,3] cannot be covered by any single tree node, so we have to create the *NSS* of span[2,3] by concatenating the *NSS*s of span [2,2] and span [3,3]. Since *NSS* of span [2,2] contains 4 element {"NN", "NP", "VV", "VP"} and *NSS* of span [3, 3] also contains 4 element {"VV", "VP", "JJ", "ADJP"}, *NSS* of span [2,3] contains 16=4*4 elements. To make the *NSS* manageable, we prune it with the following thresholds:

- each node sequence should contain less than *n* nodes
- each node sequence set should contain less than *m* node sequences
- sort node sequences according to their lengths and only keep the *k* shortest ones

Each virtual node is simply labeled by the concatenation of all its children's labels as shown in Fig. 7.

Algorithm 1. *add virtual nodes into forest*
Input: *packed forest F, alignment A*
Notation:
L: length of source sentence
NSS(*u*,*v*): the set of node sequences covering span [*u*,*v*]
VN(*ns*): virtual father node for node sequence *ns*.
Output: *modified forest F with virtual nodes*

```
1.  for length := 0 to L - 1 do
2.      for start := 1 to L - length do
3.          stop := start + length
4.          if span[start, stop] covered by tree nodes then
5.              for each node n of span [start, stop] do
6.                  add n into NSS(start, stop)
7.          else
8.              for pivot := start to stop - 1
9.                  for each ns1 in NSS(start, pivot) do
10.                     for each ns2 in NSS(pivot+1, stop) do
11.                         create ns := ns1 ⊕ ns2
12.                         if ns is not in NSS(start, stop) then
13.                             add ns into NSS(start, stop)
14.             do pruning on NSS(start, stop)
15.             if the span[start, stop] is alignable then
16.                 for each ns of NSS(start, stop) do
17.                     if node VN(ns) is not in F then
18.                         add node VN(ns) into F
19.                     add a hyper-edge h into F,
20.                     let lhs(h) := VN(ns), rhs(h) := ns
```

Algorithm 1 outputs a non-complete forest *CF* with each alignable span covered by either tree nodes or virtual nodes. Then we can easily extract our rules from the *CF* using the tree rule extraction algorithm (Mi and Huang, 2008).

Finally, to calculate rule feature probabilities for our model, we need to calculate the fractional counts (it is a kind of probability defined in Mi and Huang, 2008) of each translation rule in a parse forest. In the tree case, we can use the inside-outside-based methods (Mi and Huang 2008) to do it. In the tree sequence case, since the previous method cannot be used directly, we provide another solution by making an independent assumption that each tree in a tree sequence is independent to each other. With this assumption, the fractional counts of both tree and tree sequence can be calculated as follows:

$$c(r) = \frac{\alpha\beta(lhs(r))}{\alpha\beta(TOP)}$$

$$\alpha\beta(frag) = \prod_{v \in root(frag)} \alpha(v) * \prod_{h \in frag} P(h) * \prod_{v \in leaves(frag)} \beta(v)$$

where $c(r)$ is the fractional counts to be calculated for rule r, a *frag* is either *lhs*(r) (excluding virtual nodes and virtual hyper-edges) or any tree node in a forest, *TOP* is the root of the forest, $\alpha(.)$ and $\beta(.)$ are the outside and inside probabilities of nodes, $root(.)$ returns the root nodes of a tree sequence fragment, $leaves(.)$ returns the leaf nodes of a tree sequence fragment, $p(h)$ is the hyper-edge probability.

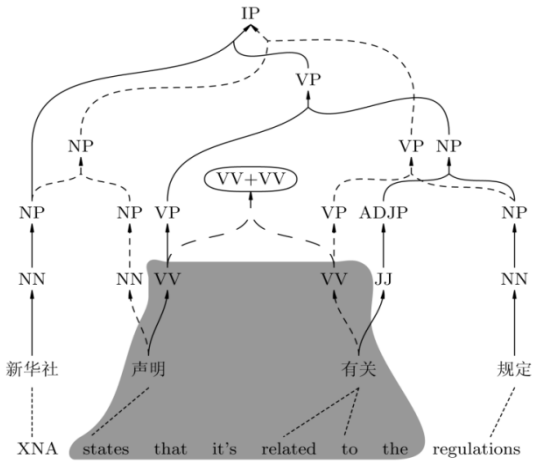


Figure 7. A virtual node in forest

5 Decoding

We benefit from the same strategy as used in our rule extraction algorithm in designing our decoding algorithm, recasting the forest-based tree sequence-to-string decoding problem as a forest-based tree-to-string decoding problem. Our decoding algorithm consists of four steps:

1) Convert the complete parse forest to a non-complete one by introducing virtual nodes.

2) Convert the non-complete parse forest into a translation forest[4] *TF* by using the translation rules and the pattern-matching algorithm presented in Mi et al. (2008).

3) Prune out redundant nodes and add auxiliary hyper-edge into the translation forest for those nodes that have either no child or no father. By this step, the translation forest *TF* becomes a complete forest.

4) Decode the translation forest using our translation model and a dynamic search algorithm.

The process of step 1 is similar to Algorithm 1 except no alignment constraint used here. This may generate a large number of additional virtual nodes; however, all redundant nodes will be filtered out in step 3. In step 2, we employ the tree-to-string pattern match algorithm (Mi et al., 2008) to convert a parse forest to a translation forest. In step 3, all those nodes not covered by any translation rules are removed. In addition, please note that the translation forest is already not a complete forest due to the virtual nodes and the pruning of rule-unmatchable nodes. We, therefore, propose Algorithm 2 to add auxiliary hyper-edges to make the translation forest complete.

In Algorithm 2, we travel the forest in bottom-up fashion (line 4-5). For each span, we do:

1) generate all the *NSS* for this span (line 7-12)

2) filter the *NSS* to a manageable size (line 13)

3) add auxiliary hyper-edges for the current span (line 15-19) if it can be covered by at least one single tree node, otherwise go to step 1 . This is the key step in our Algorithm 2. For each tree node and each node sequences covering the same span (stored in the current *NSS*), if the tree node has no children or at least one node in the node sequence has no father, we add an auxiliary hyper-edge to connect the tree node as father node with the node sequence as children. Since Algorithm 2 is DP-based and traverses the forest in a bottom-up way, all the nodes in a node sequence should already have children node after the lower level process in a small span. Finally, we re-build the *NSS* of current span for upper level *NSS* combination use (line 20-22).

In Fig. 8, the hyper-edge "IP=>NP VV+VV NP" is an auxiliary hyper-edge introduced by Algorithm 2. By Algorithm 2, we convert the translation forest into a complete translation forest. We then use a bottom-up node-based search algorithm to do decoding on the complete translation forest. We also use Cube Pruning algorithm (Huang and Chiang 2007) to speed up the translation process.

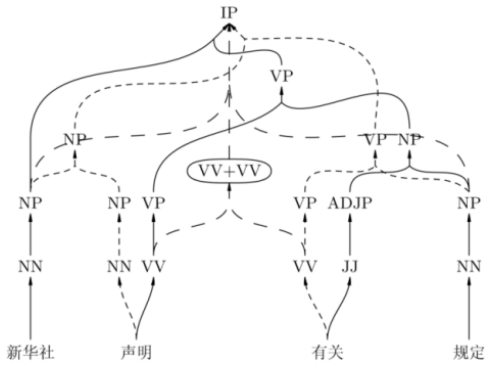


Figure 8. Auxiliary hyper-edge in a translation forest

Algorithm 2. *add auxiliary hyper-edges into mt forest F*
Input: *mt forest F*
Output: *complete forest F with auxiliary hyper-edges*

```
1.  for i := 1 to L do
2.      for each node n of span [i, i] do
3.          add n into NSS(i, i)
4.  for length := 1 to L - 1 do
5.      for start := 1 to L - length do
6.          stop := start + length
7.          for pivot := start to stop-1 do
8.              for each ns1 in NSS (start, pivot) do
9.                  for each ns2 in NSS (pivot+1,stop) do
10.                     create ns := ns1 ⊕ ns2
11.                     if ns is not in NSS(start, stop) then
12.                         add ns into NSS (start, stop)
13.         do pruning on NSS(start, stop)
14.         if there is tree node cover span [start, stop] then
15.             for each tree node n of span [start,stop] do
16.                 for each ns of NSS(start, stop) do
17.                     if node n have no children or
                            there is node in ns with no father
                            then
18.                         add auxiliary hyper-edge h into F
19.                         let lhs(h) := n, rhs(h) := ns
20.             empty NSS(start, stop)
21.             for each node n of span [start, stop] do
22.                 add n into NSS(start, stop)
```

6 Experiment

6.1 Experimental Settings

We evaluate our method on Chinese-English translation task. We use the FBIS corpus as training set, the NIST MT-2002 test set as development (dev) set and the NIST MT-2003 test set as test set. We train Charniak's parser (Charniak 2000) on CTB5 to do Chinese parsing, and modify it to output packed forest. We tune the parser on section 301-325 and test it on section 271-300. The F-measure on all sentences is 80.85%. A 3-gram language model is trained on the Xin-

[4] The concept of translation forest is proposed in Mi et al. (2008). It is a forest that consists of only the hyper-edges induced from translation rules.

hua portion of the English Gigaword3 corpus and the target side of the FBIS corpus using the SRILM Toolkits (Stolcke, 2002) with modified Kneser-Ney smoothing (Kenser and Ney, 1995). GIZA++ (Och and Ney, 2003) and the heuristics "grow-diag-final-and" are used to generate *m-to-n* word alignments. For the MER training (Och, 2003), Koehn's MER trainer (Koehn, 2007) is modified for our system. For significance test, we use Zhang et al.'s implementation (Zhang et al, 2004). Our evaluation metrics is case-sensitive BLEU-4 (Papineni et al., 2002).

For parse forest pruning (Mi et al., 2008), we utilize the Margin-based pruning algorithm presented in (Huang, 2008). Different from Mi et al. (2008) that use a static pruning threshold, our threshold is sentence-depended. For each sentence, we compute the Margin between the *n*-th best and the top 1 parse tree, then use the Margin-based pruning algorithm presented in (Huang, 2008) to do pruning. By doing so, we can guarantee to use at least all the top *n* best parse trees in the forest. However, please note that even after pruning there is still exponential number of additional trees embedded in the forest because of the sharing structure of forest. Other parameters are set as follows: maximum number of roots in a tree sequence is 3, maximum height of a translation rule is 3, maximum number of leaf nodes is 7, maximum number of node sequences on each span is 10, and maximum number of rules extracted from one node is 10000.

6.2 Experimental Results

We implement our proposed methods as a general, configurable platform for syntax-based SMT study. Based on this platform, we are able to easily implement most of the state-of-the-art syntax-based x-to-string SMT methods via simple parameter setting. For training, we set forest pruning threshold to 1 best for tree-based methods and 100 best for forest-based methods. For decoding, we set:

1) TT2S: tree-based tree-to-string model by setting the forest pruning threshold to 1 best and the number of sub-trees in a tree sequence to 1.

2) TTS2S: tree-based tree-sequence to string system by setting the forest pruning threshold to 1 best and the maximum number of sub-trees in a tree sequence to 3.

3) FT2S: forest-based tree-to-string system by setting the forest pruning threshold to 500 best, the number of sub-trees in a tree sequence to 1.

4) FTS2S: forest-based tree-sequence to string system by setting the forest pruning threshold to 500 best and the maximum number of sub-trees in a tree sequence to 3.

Model	BLEU(%)
Moses	25.68
TT2S	26.08
TTS2S	26.95
FT2S	27.66
FTS2S	**28.83**

Table 1. Performance Comparison

We use the first three syntax-based systems (TT2S, TTS2S, FT2S) and Moses (Koehn et al., 2007), the state-of-the-art phrase-based system, as our baseline systems. Table 1 compares the performance of the five methods, all of which are fine-tuned. It shows that:

1) FTS2S significantly outperforms ($p<0.05$) FT2S. This shows that tree sequence is very useful to forest-based model. Although a forest can cover much more phrases than a single tree does, there are still many non-syntactic phrases that cannot be captured by a forest due to structure divergence issue. On the other hand, tree sequence is a good solution to non-syntactic translation equivalence modeling. This is mainly because tree sequence rules are only sensitive to word alignment while tree rules, even extracted from a forest (like in FT2S), are also limited by syntax according to grammar parsing rules.

2) FTS2S shows significant performance improvement ($p<0.05$) over TTS2S due to the contribution of forest. This is mainly due to the fact that forest can offer very large number of parse trees for rule extraction and decoder.

3) Our model statistically significantly outperforms all the baselines system. This clearly demonstrates the effectiveness of our proposed model for syntax-based SMT. It also shows that the forest-based method and tree sequence-based method are complementary to each other and our proposed method is able to effectively integrate their strengths.

4) All the four syntax-based systems show better performance than Moses and three of them significantly outperforms ($p<0.05$) Moses. This suggests that syntax is very useful to SMT and translation can be viewed as a structure mapping issue as done in the four syntax-based systems.

Table 2 and Table 3 report the distribution of different kinds of translation rules in our model (training forest pruning threshold is set to 100 best) and in our decoding (decoding forest pruning threshold is set to 500 best) for one best translation generation. From the two tables, we can find that:

Rule Type	Tree to String	Tree Sequence to String
L	4,854,406	20,526,674
P	37,360,684	58,826,261
U	3,297,302	3,775,734
All	45,512,392	83,128,669

Table 2. # of rules extracted from training corpus. **L** means fully lexicalized, **P** means partially lexicalized, **U** means unlexicalized.

Rule Type	Tree to String	Tree Sequence to String
L	10,592	1,161
P	7,132	742
U	4,874	278
All	22,598	2,181

Table 3. # of rules used to generate one-best translation result in testing

1) In Table 2, the number of tree sequence rules is much larger than that of tree rules although our rule extraction algorithm only extracts those tree sequence rules over the spans that tree rules cannot cover. This suggests that the non-syntactic structure mapping is still a big challenge to syntax-based SMT.

2) Table 3 shows that the tree sequence rules is around 9% of the tree rules when generating the one-best translation. This suggests that around 9% of translation equivalences in the test set can be better modeled by tree sequence to string rules than by tree to string rules. The 9% tree sequence rules contribute 1.17 BLEU score improvement (28.83-27.66 in Table 1) to FTS2S over FT2S.

3) In Table 3, the fully-lexicalized rules are the major part (around 60%), followed by the partially-lexicalized (around 35%) and unlexicalized (around 15%). However, in Table 2, partially-lexicalized rules extracted from training corpus are the major part (more than 70%). This suggests that most partially-lexicalized rules are less effective in our model. This clearly directs our future work in model optimization.

	BLEU (%)	
N-best \ model	**FT2S**	**FTS2S**
100 Best	27.40	28.61
500 Best	27.66	28.83
2500 Best	27.66	**28.96**
5000 Best	27.79	28.89

Table 4. Impact of the forest pruning

Forest pruning is a key step for forest-based method. Table 4 reports the performance of the two forest-based models using different values of the forest pruning threshold for decoding. It shows that:

1) FTS2S significantly outperforms ($p<0.05$) FT2S consistently in all test cases. This again demonstrates the effectiveness of our proposed model. Even if in the 5000 Best case, tree sequence is still able to contribute 1.1 BLEU score improvement (28.89-27.79). It indicates the advantage of tree sequence cannot be covered by forest even if we utilize a very large forest.

2) The BLEU scores are very similar to each other when we increase the forest pruning threshold. Moreover, in one case the performance even drops. This suggests that although more parse trees in a forest can offer more structure information, they may also introduce more noise that may confuse the decoder.

7 Conclusion

In this paper, we propose a forest-based tree-sequence to string translation model to combine the strengths of forest-based methods and tree-sequence based methods. This enables our model to have the great potential to address the issues of structure divergence and parse errors for syntax-based SMT. We convert our forest-based tree sequence rule extraction and decoding issues to tree-based by introducing virtual nodes, virtual hyper-edges and auxiliary rules (hyper-edges). In our system implementation, we design a general and configurable platform for our method, based on which we can easily realize many previous syntax-based methods. Finally, we examine our methods on the FBIS corpus and the NIST MT-2003 Chinese-English translation task. Experimental results show that our model greatly outperforms the four baseline systems. Our study demonstrates that forest-based method and tree sequence-based method are complementary to each other and our proposed method is able to effectively combine the strengths of the two individual methods for syntax-based SMT.

Acknowledgement

We would like to thank Huang Yun for preparing the pictures in this paper; Run Yan for providing the java version modified MERT program and discussion on the details of MOSES; Mi Haitao for his help and discussion on re-implementing the FT2S model; Sun Jun and Xiong Deyi for their valuable suggestions.

References

Eugene Charniak. 2000. *A maximum-entropy inspired parser*. NAACL-00.

Jason Eisner. 2003. *Learning non-isomorphic tree mappings for MT.* ACL-03 (companion volume).

Michel Galley, Mark Hopkins, Kevin Knight and Daniel Marcu. 2004. *What's in a translation rule?* HLT-NAACL-04. 273-280.

Liang Huang. 2008. Forest Reranking: *Discriminative Parsing with Non-Local Features*. ACL-HLT-08. 586-594

Liang Huang and David Chiang. 2005. *Better k-best Parsing*. IWPT-05.

Liang Huang and David Chiang. 2007. *Forest rescoring: Faster decoding with integrated language models*. ACL-07. 144–151

Liang Huang, Kevin Knight and Aravind Joshi. 2006. *Statistical Syntax-Directed Translation with Extended Domain of Locality*. AMTA-06. (poster)

Reinhard Kenser and Hermann Ney. 1995. *Improved backing-off for M-gram language modeling*. ICASSP-95. 181-184

Dan Klein and Christopher D. Manning. 2001. *Parsing and Hypergraphs*. IWPT-2001.

Philipp Koehn, F. J. Och and D. Marcu. 2003. *Statistical phrase-based translation.* HLT-NAACL-03. 127-133.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin and Evan Herbst. 2007. *Moses: Open Source Toolkit for Statistical Machine Translation.* ACL-07. 177-180. (poster)

Yang Liu, Qun Liu and Shouxun Lin. 2006. *Tree-to-String Alignment Template for Statistical Machine Translation.* COLING-ACL-06. 609-616.

Yang Liu, Yun Huang, Qun Liu and Shouxun Lin. 2007. *Forest-to-String Statistical Translation Rules.* ACL-07. 704-711.

Haitao Mi, Liang Huang, and Qun Liu. 2008. *Forest-based translation*. ACL-HLT-08. 192-199.

Haitao Mi and Liang Huang. 2008. *Forest-based Translation Rule Extraction*. EMNLP-08. 206-214.

Franz J. Och and Hermann Ney. 2002. *Discriminative training and maximum entropy models for statistical machine translation.* ACL-02. 295-302.

Franz J. Och. 2003. *Minimum error rate training in statistical machine translation.* ACL-03. 160-167.

Franz Josef Och and Hermann Ney. 2003. *A Systematic Comparison of Various Statistical Alignment Models*. Computational Linguistics. 29(1) 19-51.

Kishore Papineni, Salim Roukos, ToddWard and Wei-Jing Zhu. 2002. *BLEU: a method for automatic evaluation of machine translation.* ACL-02. 311-318.

Andreas Stolcke. 2002. *SRILM - an extensible language modeling toolkit.* ICSLP-02. 901-904.

Min Zhang, Hongfei Jiang, Ai Ti Aw, Jun Sun, Sheng Li and Chew Lim Tan. 2007. *A Tree-to-Tree Alignment-based Model for Statistical Machine Translation*. MT-Summit-07. 535-542.

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, Sheng Li. 2008a. *A Tree Sequence Alignment-based Tree-to-Tree Translation Model.* ACL-HLT-08. 559-567.

Min Zhang, Hongfei Jiang, Haizhou Li, Aiti Aw, Sheng Li. 2008b. *Grammar Comparison Study for Translational Equivalence Modeling and Statistical Machine Translation.* COLING-08. 1097-1104.

Ying Zhang, Stephan Vogel, Alex Waibel. 2004. *Interpreting BLEU/NIST scores: How much improvement do we need to have a better system?* LREC-04. 2051-2054.

Active Learning for Multilingual Statistical Machine Translation*

Gholamreza Haffari and **Anoop Sarkar**
School of Computing Science, Simon Fraser University
British Columbia, Canada
{ghaffar1,anoop}@cs.sfu.ca

Abstract

Statistical machine translation (SMT) models require bilingual corpora for training, and these corpora are often multilingual with parallel text in multiple languages simultaneously. We introduce an *active learning* task of adding a new language to an existing multilingual set of parallel text and constructing high quality MT systems, from each language in the collection into this new target language. We show that adding a new language using active learning to the EuroParl corpus provides a significant improvement compared to a random sentence selection baseline. We also provide new highly effective sentence selection methods that improve AL for phrase-based SMT in the multilingual and single language pair setting.

1 Introduction

The main source of training data for statistical machine translation (SMT) models is a parallel corpus. In many cases, the same information is available in multiple languages simultaneously as a multilingual parallel corpus, e.g., European Parliament (EuroParl) and U.N. proceedings. In this paper, we consider how to use active learning (AL) in order to add a new language to such a multilingual parallel corpus and at the same time we construct an MT system from each language in the original corpus into this new target language. We introduce a novel combined measure of translation quality for multiple target language outputs (the same content from multiple source languages).

The multilingual setting provides new opportunities for AL over and above a single language pair. This setting is similar to the multi-task AL scenario (Reichart et al., 2008). In our case, the multiple tasks are individual machine translation tasks for several language pairs. The nature of the translation processes vary from any of the source languages to the new language depending on the characteristics of each source-target language pair, hence these tasks are competing for annotating the same resource. However it may be that in a single language pair, AL would pick a particular sentence for annotation, but in a multilingual setting, a different source language might be able to provide a good translation, thus saving annotation effort. In this paper, we explore how multiple MT systems can be used to effectively pick instances that are more likely to improve training quality.

Active learning is framed as an iterative learning process. In each iteration new human labeled instances (manual translations) are added to the training data based on their expected training quality. However, if we start with only a small amount of initial parallel data for the new target language, then translation quality is very poor and requires a very large injection of human labeled data to be effective. To deal with this, we use a novel framework for active learning: we assume we are given a small amount of parallel text and a large amount of monolingual source language text; using these resources, we create a large noisy parallel text which we then iteratively improve using small injections of human translations. When we build multiple MT systems from multiple source languages to the new target language, each MT system can be seen as a different 'view' on the desired output translation. Thus, we can train our multiple MT systems using either *self-training* or *co-training* (Blum and Mitchell, 1998). In self-training each MT system is re-trained using human labeled data plus its own noisy translation output on the unlabeled data. In co-training each MT system is re-trained using human labeled data plus noisy translation output from the other MT systems in the ensemble. We use consensus translations (He et al., 2008; Rosti et al., 2007; Matusov et al., 2006) as an effective method for co-training between multiple MT systems.

This paper makes the following contributions:

- We provide a new framework for multilingual MT, in which we build multiple MT systems and add a new language to an existing multilingual parallel corpus. The multilingual set-

*Thanks to James Peltier for systems support for our experiments. This research was partially supported by NSERC, Canada (RGPIN: 264905) and an IBM Faculty Award.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 181–189,
Suntec, Singapore, 2-7 August 2009.

ting allows new features for active learning which we exploit to improve translation quality while reducing annotation effort.

- We introduce new highly effective sentence selection methods that improve phrase-based SMT in the multilingual and single language pair setting.
- We describe a novel co-training based active learning framework that exploits consensus translations to effectively select only those sentences that are difficult to translate for all MT systems, thus sharing annotation cost.
- We show that using active learning to add a new language to the EuroParl corpus provides a significant improvement compared to the strong random sentence selection baseline.

2 AL-SMT: Multilingual Setting

Consider a multilingual parallel corpus, such as EuroParl, which contains parallel sentences for several languages. Our goal is to add a new language to this corpus, *and* at the same time to construct high quality MT systems from the existing languages (in the multilingual corpus) to the new language. This goal is formalized by the following objective function:

$$\mathcal{O} = \sum_{d=1}^{D} \alpha_d \times TQ(M_{F^d \rightarrow E}) \quad (1)$$

where F^d's are the source languages in the multilingual corpus (D is the total number of languages), and E is the new language. The translation quality is measured by TQ for individual systems $M_{F^d \rightarrow E}$; it can be BLEU score or WER/PER (Word error rate and position independent WER) which induces a maximization or minimization problem, respectively. The non-negative weights α_d reflect the importance of the different translation tasks and $\sum_d \alpha_d = 1$. AL-SMT formulation for *single* language pair is a special case of this formulation where only one of the α_d's in the objective function (1) is one and the rest are zero. Moreover the algorithmic framework that we introduce in Sec. 2.1 for AL in the multilingual setting includes the single language pair setting as a special case (Haffari et al., 2009).

We denote the large *unlabeled* multilingual corpus by $\mathbb{U} := \{(\mathbf{f}_j^1, .., \mathbf{f}_j^D)\}$, and the small *labeled* multilingual corpus by $\mathbb{L} := \{(\mathbf{f}_i^1, .., \mathbf{f}_i^D, \mathbf{e}_i)\}$. We overload the term *entry* to denote a tuple in $\mathbb{L}$ or in $\mathbb{U}$ (it should be clear from the context). For a single language pair we use U and L.

2.1 The Algorithmic Framework

Algorithm 1 represents our AL approach for the multilingual setting. We train our initial MT systems $\{M_{F^d \rightarrow E}\}_{d=1}^{D}$ on the multilingual corpus $\mathbb{L}$, and use them to translate *all* monolingual sentences in $\mathbb{U}$. We denote sentences in $\mathbb{U}$ together with their multiple translations by $\mathbb{U}^+$ (line 4 of Algorithm 1). Then we retrain the SMT systems on $\mathbb{L} \cup \mathbb{U}^+$ and use the resulting model to decode the test set. Afterwards, we select and remove a subset of highly informative sentences from $\mathbb{U}$, and add those sentences together with their human-provided translations to $\mathbb{L}$. This process is continued iteratively until a certain level of translation quality is met (we use the BLEU score, WER and PER) (Papineni et al., 2002). In the baseline, against which we compare our sentence selection methods, the sentences are chosen *randomly*.

When (re-)training the models, two phrase tables are learned for each SMT model: one from the labeled data $\mathbb{L}$ and the other one from *pseudo-labeled* data $\mathbb{U}^+$ (which we call the *main* and *auxiliary* phrase tables respectively). (Ueffing et al., 2007; Haffari et al., 2009) show that treating $\mathbb{U}^+$ as a source for a new feature function in a log-linear model for SMT (Och and Ney, 2004) allows us to maximally take advantage of unlabeled data by finding a weight for this feature using minimum error-rate training (MERT) (Och, 2003).

Since each entry in $\mathbb{U}^+$ has multiple translations, there are two options when building the auxiliary table for a particular language pair (F^d, E): (i) to use the corresponding translation $\mathbf{e}^d$ of the source language in a *self-training* setting, or (ii) to use the *consensus* translation among all the translation candidates $(\mathbf{e}^1, .., \mathbf{e}^D)$ in a *co-training* setting (sharing information between multiple SMT models).

A whole range of methods exist in the literature for combining the output translations of multiple MT systems for a *single* language pair, operating either at the sentence, phrase, or word level (He et al., 2008; Rosti et al., 2007; Matusov et al., 2006). The method that we use in this work operates at the sentence level, and picks a single high quality translation from the union of the n-best lists generated by multiple SMT models. Sec. 5 gives

Algorithm 1 AL-SMT-Multiple

1: Given multilingual corpora $\mathbb{L}$ and $\mathbb{U}$
2: $\{M_{F^d \to E}\}_{d=1}^{D} = \textbf{multrain}(\mathbb{L}, \emptyset)$
3: **for** $t = 1, 2, ...$ **do**
4: $\quad \mathbb{U}^+ = \textbf{multranslate}(\mathbb{U}, \{M_{F^d \to E}\}_{d=1}^{D})$
5: $\quad$ Select k sentences from $\mathbb{U}^+$, and ask a human for their *true* translations.
6: $\quad$ Remove the k sentences from $\mathbb{U}$, and add the k sentence pairs (translated by human) to $\mathbb{L}$
7: $\quad \{M_{F^d \to E}\}_{d=1}^{D} = \textbf{multrain}(\mathbb{L}, \mathbb{U}^+)$
8: $\quad$ Monitor the performance on the test set
9: **end for**

more details about features which are used in our consensus finding method, and how it is trained. Now let us address the important question of selecting highly informative sentences (step 5 in the Algorithm 1) in the following section.

3 Sentence Selection: Multiple Language Pairs

The goal is to optimize the objective function (1) with minimum human effort in providing the translations. This motivates selecting sentences which are *maximally* beneficial for *all* the MT systems. In this section, we present several protocols for sentence selection based on the combined information from multiple language pairs.

3.1 Alternating Selection

The simplest selection protocol is to choose k sentences (entries) in the first iteration of AL which improve maximally the first model $M_{F^1 \to E}$, while ignoring other models. In the second iteration, the sentences are selected with respect to the second model, and so on (Reichart et al., 2008).

3.2 Combined Ranking

Pick any AL-SMT scoring method for a *single* language pair (see Sec. 4). Using this method, we rank the entries in unlabeled data $\mathbb{U}$ for each translation task defined by language pair (F^d, E). This results in several ranking lists, each of which represents the importance of entries with respect to a particular translation task. We combine these rankings using a combined score:

$$Score\big((\mathbf{f}^1, .., \mathbf{f}^D)\big) = \sum_{d=1}^{D} \alpha_d \text{Rank}_d(\mathbf{f}^d)$$

$\text{Rank}_d(.)$ is the ranking of a sentence in the list for the d^{th} translation task (Reichart et al., 2008).

3.3 Disagreement Among the Translations

Disagreement among the candidate translations of a particular entry is evidence for the difficulty of that entry for different translation models. The reason is that disagreement increases the possibility that most of the translations are not correct. Therefore it would be beneficial to ask human for the translation of these hard entries.

Now the question is how to quantify the notion of disagreement among the candidate translations $(\mathbf{e}^1, .., \mathbf{e}^D)$. We propose two measures of disagreement which are related to the portion of shared n-grams ($n \leq 4$) among the translations:

- Let $\mathbf{e}^c$ be the consensus among all the candidate translations, then define the disagreement as $\sum_d \alpha_d \big(1 - \text{BLEU}(\mathbf{e}^c, \mathbf{e}^d)\big)$.
- Based on the disagreement of every pair of candidate translations: $\sum_d \alpha_d \sum_{d'} \big(1 - \text{BLEU}(\mathbf{e}^{d'}, \mathbf{e}^d)\big)$.

For the single language pair setting, (Haffari et al., 2009) presents and compares several sentence selection methods for statistical phrase-based machine translation. We introduce novel techniques which outperform those methods in the next section.

4 Sentence Selection: Single Language Pair

Phrases are basic units of translation in phrase-based SMT models. The phrases which may potentially be extracted from a sentence indicate its informativeness. The more new phrases a sentence can offer, the more informative it is; since it boosts the *generalization* of the model. Additionally phrase translation probabilities need to be estimated accurately, which means sentences that offer phrases whose occurrences in the corpus were rare are informative. When selecting new sentences for human translation, we need to pay attention to this tradeoff between *exploration* and *exploitation*, i.e. selecting sentences to discover new phrases v.s. estimating accurately the phrase translation probabilities. Smoothing techniques partly handle accurate estimation of translation probabilities when the events occur rarely (indeed it is the main reason for smoothing). So we mainly focus on how to expand effectively the lexicon or set of phrases of the model.

The more frequent a phrase (not a *phrase pair*) is in the *unlabeled* data, the more important it is to

know its translation; since it is more likely to see it in test data (specially when the test data is in-domain with respect to unlabeled data). The more frequent a phrase is in the *labeled* data, the more unimportant it is; since probably we have observed most of its translations.

In the labeled data L, phrases are the ones which are extracted by the SMT models; but what are the candidate phrases in the unlabeled data U? We use the currently trained SMT models to answer this question. Each translation in the n-best list of translations (generated by the SMT models) corresponds to a particular segmentation of a sentence, which breaks that sentence into several fragments (see Fig. 1). Some of these fragments are the source language part of a phrase pair available in the phrase table, which we call *regular phrases* and denote their set by X_s^{reg} for a sentence s. However, there are some fragments in the sentence which are *not* covered by the phrase table – possibly because of the OOVs (out-of-vocabulary words) or the constraints imposed by the phrase extraction algorithm – called X_s^{oov} for a sentence s. Each member of X_s^{oov} offers a set of *potential phrases* (also referred to as *OOV phrases*) which are not observed due to the *latent* segmentation of this fragment. We present two generative models for the phrases and show how to estimate and use them for sentence selection.

4.1 Model 1

In the first model, the generative story is to generate phrases for each sentence based on independent draws from a multinomial. The sample space of the multinomial consists of both regular and OOV phrases.

We build two models, i.e. two multinomials, one for labeled data and the other one for unlabeled data. Each model is trained by maximizing the log-likelihood of its corresponding data:

$$\mathcal{L}_{\mathcal{D}} := \sum_{\mathbf{s}\in\mathcal{D}} \tilde{P}(\mathbf{s}) \sum_{\mathbf{x}\in X_{\mathbf{s}}} \log P(\mathbf{x}|\boldsymbol{\theta}_{\mathcal{D}}) \qquad (2)$$

where $\mathcal{D}$ is either L or U, $\tilde{P}(\mathbf{s})$ is the *empirical* distribution of the sentences[1], and $\boldsymbol{\theta}_{\mathcal{D}}$ is the parameter vector of the corresponding probability distribution. When $\mathbf{x} \in X_{\mathbf{s}}^{oov}$, we will have

$$\begin{aligned} P(\mathbf{x}|\boldsymbol{\theta}_U) &= \sum_{h\in H_{\mathbf{x}}} P(\mathbf{x}, h|\boldsymbol{\theta}_U) \\ &= \sum_{h\in H_{\mathbf{x}}} P(h) P(\mathbf{x}|h, \boldsymbol{\theta}_U) \\ &= \frac{1}{|H_{\mathbf{x}}|} \sum_{h\in H_{\mathbf{x}}} \prod_{\mathbf{y}\in Y_{\mathbf{x}}^h} \boldsymbol{\theta}_U(\mathbf{y}) \quad (3) \end{aligned}$$

where $H_{\mathbf{x}}$ is the space of all possible segmentations for the OOV fragment $\mathbf{x}$, $Y_{\mathbf{x}}^h$ is the resulting phrases from $\mathbf{x}$ based on the segmentation h, and $\boldsymbol{\theta}_U(\mathbf{y})$ is the probability of the OOV phrase $\mathbf{y}$ in the multinomial associated with U. We let $H_{\mathbf{x}}$ to be all possible segmentations of the fragment $\mathbf{x}$ for which the resulting phrase lengths are not greater than the maximum length constraint for phrase extraction in the underlying SMT model. Since we do not know anything about the segmentations a priori, we have put a uniform distribution over such segmentations.

Maximizing (2) to find the maximum likelihood parameters for this model is an extremely difficult problem[2]. Therefore, we maximize the following lower-bound on the log-likelihood which is derived using Jensen's inequality:

$$\begin{aligned} \mathcal{L}_{\mathcal{D}} \geq \sum_{\mathbf{s}\in\mathcal{D}} \tilde{P}(\mathbf{s}) \Big[& \sum_{\mathbf{x}\in X_{\mathbf{s}}^{reg}} \log \boldsymbol{\theta}_{\mathcal{D}}(\mathbf{x}) \\ + & \sum_{\mathbf{x}\in X_{\mathbf{s}}^{oov}} \sum_{h\in H_{\mathbf{x}}} \frac{1}{|H_{\mathbf{x}}|} \sum_{\mathbf{y}\in Y_{\mathbf{x}}^h} \log \boldsymbol{\theta}_{\mathcal{D}}(\mathbf{y}) \Big] \quad (4) \end{aligned}$$

Maximizing (4) amounts to set the probability of each regular / potential phrase proportional to its count / *expected* count in the data $\mathcal{D}$.

Let $\rho_k(\mathbf{x}_{i:j})$ be the number of possible segmentations from position i to position j of an OOV fragment $\mathbf{x}$, and k is the maximum phrase length;

$$\rho_k(\mathbf{x}_{1:|\mathbf{x}|}) = \begin{cases} 0, & \text{if } |\mathbf{x}| = 0 \\ 1, & \text{if } |\mathbf{x}| = 1 \\ \sum_{i=1}^{k} \rho_k(\mathbf{x}_{i+1:|\mathbf{x}|}), & \text{otherwise} \end{cases}$$

which gives us a dynamic programming algorithm to compute the number of segmentation $|H_{\mathbf{x}}| = \rho_k(\mathbf{x}_{1:|\mathbf{x}|})$ of the OOV fragment $\mathbf{x}$. The expected count of a potential phrase $\mathbf{y}$ based on an OOV segment $\mathbf{x}$ is (see Fig. 1.c):

$$E[\mathbf{y}|\mathbf{x}] = \frac{\sum_{i\leq j} \delta_{[\mathbf{y}=\mathbf{x}_{i:j}]} \rho_k(\mathbf{x}_{1:i-1}) \rho_k(\mathbf{x}_{j+1:|\mathbf{x}|})}{\rho_k(\mathbf{x})}$$

[1] $\tilde{P}(\mathbf{s})$ is the number of times that the sentence $\mathbf{s}$ is seen in $\mathcal{D}$ divided by the number of all sentences in $\mathcal{D}$.

[2] Setting partial derivatives of the Lagrangian to zero amounts to finding the roots of a system of multivariate polynomials (a major topic in Algebraic Geometry).

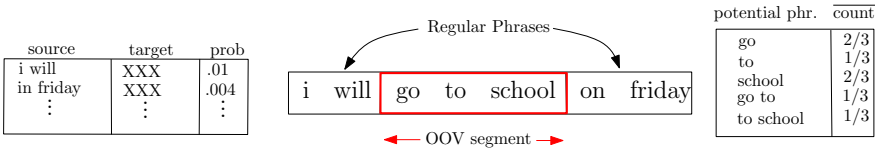


Figure 1: The given sentence in (b) is segmented, based on the source side phrases extracted from the phrase table in (a), to yield regular phrases and OOV segment. The table in (c) shows the potential phrases extracted from the OOV segment "go to school" and their expected counts (denoted by $\overline{\text{count}}$) where the maximum length for the potential phrases is set to 2. In the example, "go to school" has 3 segmentations with maximum phrase length 2: *(go)(to school), (go to)(school), (go)(to)(school).*

where $\delta_{[C]}$ is 1 if the condition C is true, and zero otherwise. We have used the fact that the number of occurrences of a phrase spanning the indices $[i, j]$ is the product of the number of segmentations of the left and the right sub-fragments, which are $\rho_k(\mathbf{x}_{1:i-1})$ and $\rho_k(\mathbf{x}_{j+1:|\mathbf{x}|})$ respectively.

4.2 Model 2

In the second model, we consider a mixture model of two multinomials responsible for generating phrases in each of the labeled and unlabeled data sets. To generate a phrase, we first toss a coin and depending on the outcome we either generate the phrase from the multinomial associated with regular phrases $\boldsymbol{\theta}_U^{reg}$ or potential phrases $\boldsymbol{\theta}_U^{oov}$:

$$P(\mathbf{x}|\boldsymbol{\theta}_U) := \beta_U \boldsymbol{\theta}_U^{reg}(\mathbf{x}) + (1-\beta_U)\boldsymbol{\theta}_U^{oov}(\mathbf{x})$$

where $\boldsymbol{\theta}_U$ includes the mixing weight β and the parameter vectors of the two multinomials. The mixture model associated with L is written similarly. The parameter estimation is based on maximizing a lower-bound on the log-likelihood which is similar to what was done for the Model 1.

4.3 Sentence Scoring

The sentence score is a linear combination of two terms: one coming from regular phrases and the other from OOV phrases:

$$\phi_1(\mathbf{s}) := \frac{\lambda}{|X_{\mathbf{s}}^{reg}|} \sum_{\mathbf{x}\in X_{\mathbf{s}}^{reg}} \log \frac{P(\mathbf{x}|\boldsymbol{\theta}_U)}{P(\mathbf{x}|\boldsymbol{\theta}_L)}$$
$$+\frac{1-\lambda}{|X_{\mathbf{s}}^{oov}|} \sum_{\mathbf{x}\in X_{\mathbf{s}}^{oov}} \sum_{h\in H_{\mathbf{x}}} \frac{1}{|H_{\mathbf{x}}|} \log \prod_{\mathbf{y}\in Y_{\mathbf{x}}^{h}} \frac{P(\mathbf{y}|\boldsymbol{\theta}_U)}{P(\mathbf{y}|\boldsymbol{\theta}_L)}$$

where we use either Model 1 or Model 2 for $P(.|\boldsymbol{\theta}_{\mathcal{D}})$. The first term is the log probability ratio of regular phrases under phrase models corresponding to unlabeled and labeled data, and the second term is the *expected* log *probability ratio* (ELPR) under the two models. Another option for the contribution of OOV phrases is to take log *of expected probability ratio* (LEPR):

$$\phi_2(\mathbf{s}) := \frac{\lambda}{|X_{\mathbf{s}}^{reg}|} \sum_{\mathbf{x}\in X_{\mathbf{s}}^{reg}} \log \frac{P(\mathbf{x}|\boldsymbol{\theta}_U)}{P(\mathbf{x}|\boldsymbol{\theta}_L)}$$
$$+\frac{1-\lambda}{|X_{\mathbf{s}}^{oov}|} \sum_{\mathbf{x}\in X_{\mathbf{s}}^{oov}} \log \sum_{h\in H_{\mathbf{x}}} \frac{1}{|H_{\mathbf{x}}|} \prod_{\mathbf{y}\in Y_{\mathbf{x}}^{h}} \frac{P(\mathbf{y}|\boldsymbol{\theta}_U)}{P(\mathbf{y}|\boldsymbol{\theta}_L)}$$

It is not difficult to prove that there is no difference between Model 1 and Model 2 when ELPR scoring is used for sentence selection. However, the situation is different for LEPR scoring: the two models produce different sentence rankings in this case.

5 Experiments

Corpora. We pre-processed the EuroParl corpus (*http://www.statmt.org/europarl*) (Koehn, 2005) and built a multilingual parallel corpus with 653,513 sentences, excluding the Q4/2000 portion of the data (2000-10 to 2000-12) which is reserved as the test set. We subsampled 5,000 sentences as the labeled data $\mathbb{L}$ and 20,000 sentences as $\mathbb{U}$ for the pool of untranslated sentences (while hiding the English part). The test set consists of 2,000 multi-language sentences and comes from the multilingual parallel corpus built from Q4/2000 portion of the data.

Consensus Finding. Let T be the union of the n-best lists of translations for a particular sentence. The consensus translation $\mathbf{t}^c$ is

$$\arg\max_{\mathbf{t}\in T} w_1 \frac{LM(\mathbf{t})}{|\mathbf{t}|} + w_2 \frac{Q_d(\mathbf{t})}{|\mathbf{t}|} + w_3 R_d(\mathbf{t}) + w_{4,d}$$

where LM($\mathbf{t}$) is the score from a 3-gram language model, $Q_d(\mathbf{t})$ is the translation score generated by the decoder for $M_{F^d\to E}$ if $\mathbf{t}$ is produced by the dth SMT model, $R_d(\mathbf{t})$ is the rank of the translation in the n-best list produced by the dth model, $w_{4,d}$ is a bias term for each translation model to make their scores comparable, and $|\mathbf{t}|$ is the length

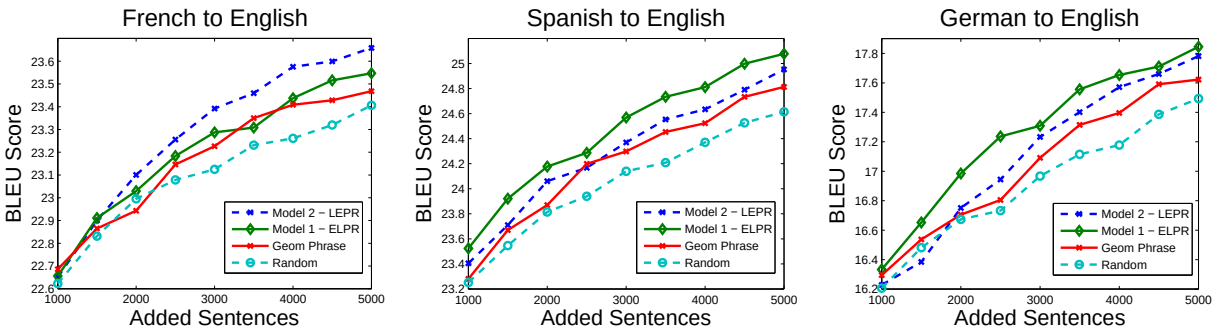


Figure 2: The performance of different sentence selection strategies as the iteration of AL loop goes on for three translation tasks. Plots show the performance of sentence selection methods for single language pair in Sec. 4 compared to the GeomPhrase (Haffari et al., 2009) and random sentence selection baseline.

of the translation sentence. The number of weights w_i is 3 plus the number of source languages, and they are trained using minimum error-rate training (MERT) to maximize the BLEU score (Och, 2003) on a development set.

Parameters. We use add-ϵ smoothing where $\epsilon = .5$ to smooth the probabilities in Sec. 4; moreover $\lambda = .4$ for ELPR and LEPR sentence scoring and maximum phrase length k is set to 4. For the multilingual experiments (which involve four source languages) we set $\alpha_d = .25$ to make the importance of individual translation tasks equal.

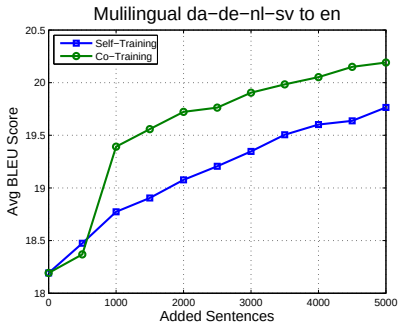


Figure 3: Random sentence selection baseline using self-training and co-training (Germanic languages to English).

5.1 Results

First we evaluate the proposed sentence selection methods in Sec. 4 for the single language pair. Then the best method from the single language pair setting is used to evaluate sentence selection methods for AL in multilingual setting. After building the initial MT system for each experiment, we select and remove 500 sentences from $\mathbb{U}$ and add them together with translations to $\mathbb{L}$ for 10 total iterations. The random sentence selection baselines are averaged over 3 independent runs.

mode	self-train		co-train	
Method	wer	per	wer	per
Combined Rank	***40.2***	***30.0***	**40.0**	***29.6***
Alternate	41.0	*30.2*	40.1	*30.1*
Disagree-Pairwise	41.9	32.0	40.5	30.9
Disagree-Center	41.8	31.8	40.6	30.7
Random Baseline	41.6	31.0	40.5	30.7

Germanic languages to English

mode	self-train		co-train	
Method	wer	per	wer	per
Combined Rank	***37.7***	***27.3***	***37.3***	***27.0***
Alternate	***37.7***	***27.3***	***37.3***	***27.0***
Random Baseline	38.6	28.1	38.1	27.6

Romance languages to English

Table 1: Comparison of multilingual selection methods with WER (word error rate), PER (position independent WER). 95% confidence interval for WER numbers is 0.7 and for PER numbers is 0.5. **Bold:** best result, *italic:* significantly better.

We use three language pairs in our single language pair experiments: French-English, German-English, and Spanish- English. In addition to random sentence selection baseline, we also compare the methods proposed in this paper to the best method reported in (Haffari et al., 2009) denoted by GeomPhrase, which differs from our models since it considers each individual OOV segment as a single OOV phrase and does not consider subsequences. The results are presented in Fig. 2. Selecting sentences based on our proposed methods outperform the random sentence selection baseline and GeomPhrase. We suspect for the situations where $\mathbb{L}$ is out-of-domain and the average phrase length is relatively small, our method will outperform GeomPhrase even more.

For the multilingual experiments, we use Germanic (German, Dutch, Danish, Swedish) and Romance (French, Spanish, Italian, Portuguese[3]) lan-

[3] A reviewer pointed out that EuroParl English-Portuguese

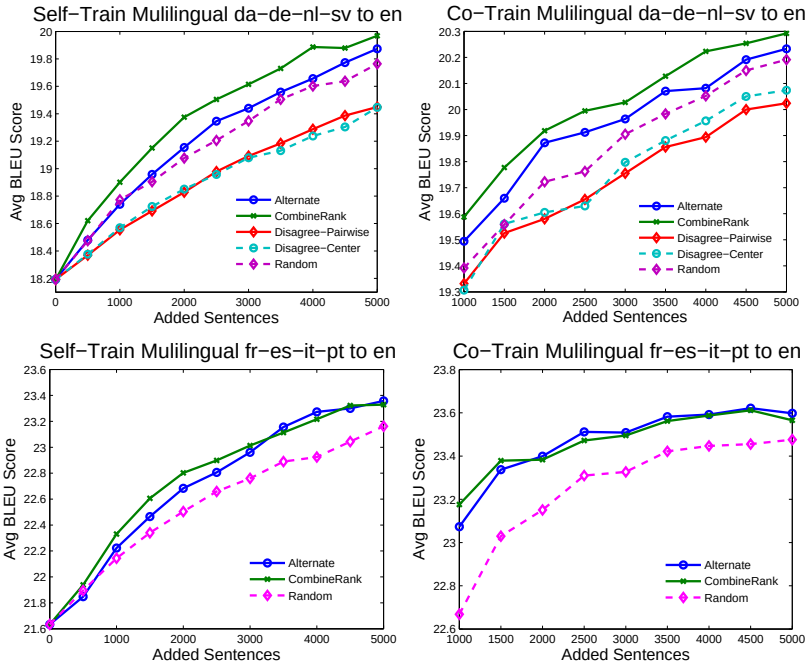


Figure 4: The left/right plot show the performance of our AL methods for multilingual setting combined with self-training/co-training. The sentence selection methods from Sec. 3 are compared with random sentence selection baseline. The top plots correspond to Danish-German-Dutch-Swedish to English, and the bottom plots correspond to French-Spanish-Italian-Portuguese to English.

guages as the source and English as the target language as two sets of experiments.[4] Fig. 3 shows the performance of random sentence selection for AL combined with self-training/co-training for the multi-source translation from the four Germanic languages to English. It shows that the co-training mode outperforms the self-training mode by almost 1 BLEU point. The results of selection strategies in the multilingual setting are presented in Fig. 4 and Tbl. 1. Having noticed that Model 1 with ELPR performs well in the single language pair setting, we use it to rank entries for individual translation tasks. Then these rankings are used by 'Alternate' and 'Combined Rank' selection strategies in the multilingual case. The 'Combined Rank' method outperforms all the other methods including the strong random selection baseline in both self-training and co-training modes. The disagreement-based selection methods underperform the baseline for translation of Germanic languages to English, so we omitted them for the Romance language experiments.

5.2 Analysis

The basis for our proposed methods has been the popularity of regular/OOV phrases in U and their unpopularity in L, which is measured by $\frac{P(\mathbf{x}|\boldsymbol{\theta}_U)}{P(\mathbf{x}|\boldsymbol{\theta}_L)}$. We need $P(\mathbf{x}|\boldsymbol{\theta}_U)$, the *estimated* distribution of phrases in U, to be as similar as possible to $P^*(\mathbf{x})$, the *true* distribution of phrases in U. We investigate this issue for regular/OOV phrases as follows:

- Using the output of the initially trained MT system on L, we extract the regular/OOV phrases as described in §4. The smoothed relative frequencies give us the regular/OOV phrasal distributions.
- Using the *true* English translation of the sentences in U, we extract the true phrases. Separating the phrases into two sets of regular and OOV phrases defined by the previous step, we use the smoothed relative frequencies and form the *true* OOV/regular phrasal distributions.

We use the KL-divergence to see how dissimilar are a pair of given probability distributions. As Tbl. 2 shows, the KL-divergence between the true and estimated distributions are less than that

	De2En	Fr2En	Es2En
$KL(P^*_{reg} \parallel P_{reg})$	4.37	4.17	4.38
$KL(P^*_{reg} \parallel unif)$	5.37	5.21	5.80
$KL(P^*_{oov} \parallel P_{oov})$	3.04	4.58	4.73
$KL(P^*_{oov} \parallel unif)$	3.41	4.75	4.99

Table 2: For regular/OOV phrases, the KL-divergence between the true distribution (P^*) and the estimated (P) or uniform ($unif$) distributions are shown, where: $KL(P^* \parallel P) := \sum_x P^*(x) \log \frac{P^*(x)}{P(x)}$.

data is very noisy and future work should omit this pair.

[4]Choice of Germanic and Romance for our experimental setting is inspired by results in (Cohn and Lapata, 2007)

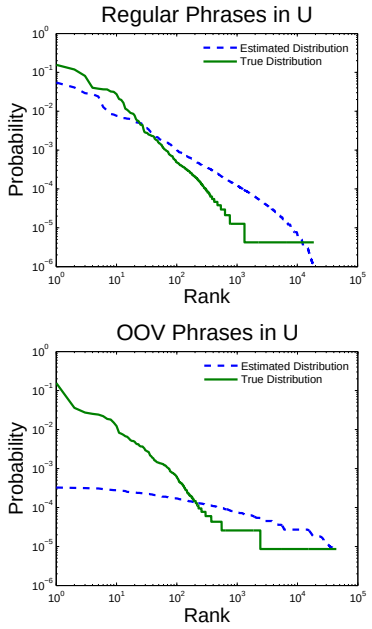


Figure 5: The log-log Zipf plots representing the true and estimated probabilities of a (source) *phrase* vs the rank of that phrase in the German to English translation task. The plots for the Spanish to English and French to English tasks are also similar to the above plots, and confirm a power law behavior in the true phrasal distributions.

between the true and uniform distributions, in all three language pairs. Since uniform distribution conveys no information, this is evidence that there is some information encoded in the estimated distribution about the true distribution. However we noticed that the true distributions of regular/OOV phrases exhibit Zipfian (power law) behavior[5] which is not well captured by the estimated distributions (see Fig. 5). Enhancing the estimated distributions to capture this power law behavior would improve the quality of the proposed sentence selection methods.

6 Related Work

(Haffari et al., 2009) provides results for active learning for MT using a single language pair. Our work generalizes to the use of multilingual corpora using new methods that are not possible with a single language pair. In this paper, we also introduce new selection methods that outperform the methods in (Haffari et al., 2009) even for MT with a single language pair. In addition in this paper by considering multilingual parallel corpora we were able to introduce co-training for AL, while (Haffari et al., 2009) only use self-training since they are using a single language pair.

[5]This observation is at the *phrase* level and not at the *word* (Zipf, 1932) or even n-gram level (Ha et al., 2002).

(Reichart et al., 2008) introduces multi-task active learning where unlabeled data require annotations for multiple tasks, e.g. they consider named-entities and parse trees, and showed that multiple tasks helps selection compared to individual tasks. Our setting is different in that the target language is the same across multiple MT tasks, which we exploit to use consensus translations and co-training to improve active learning performance.

(Callison-Burch and Osborne, 2003b; Callison-Burch and Osborne, 2003a) provide a co-training approach to MT, where one language pair creates data for another language pair. In contrast, our co-training approach uses consensus translations and our setting for active learning is very different from their semi-supervised setting. A Ph.D. proposal by Chris Callison-Burch (Callison-burch, 2003) lays out the promise of AL for SMT and proposes some algorithms. However, the lack of experimental results means that performance and feasibility of those methods cannot be compared to ours.

While we use consensus translations (He et al., 2008; Rosti et al., 2007; Matusov et al., 2006) as an effective method for co-training in this paper, unlike consensus for system combination, the source languages for each of our MT systems are different, which rules out a set of popular methods for obtaining consensus translations which assume translation for a single language pair. Finally, we briefly note that triangulation (see (Cohn and Lapata, 2007)) is orthogonal to the use of co-training in our work, since it only enhances each MT system in our ensemble by exploiting the multilingual data. In future work, we plan to incorporate triangulation into our active learning approach.

7 Conclusion

This paper introduced the novel active learning task of adding a new language to an existing multilingual set of parallel text. We construct SMT systems from each language in the collection into the new target language. We show that we can take advantage of multilingual corpora to decrease annotation effort thanks to the highly effective sentence selection methods we devised for active learning in the single language-pair setting which we then applied to the multilingual sentence selection protocols. In the multilingual setting, a novel co-training method for active learning in SMT is proposed using consensus translations which outperforms AL-SMT with self-training.

References

Avrim Blum and Tom Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-Training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory (COLT 1998)*, Madison, Wisconsin, USA, July 24-26. ACM.

Chris Callison-Burch and Miles Osborne. 2003a. Bootstrapping parallel corpora. In *NAACL workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*.

Chris Callison-Burch and Miles Osborne. 2003b. Co-training for statistical machine translation. In *Proceedings of the 6th Annual CLUK Research Colloquium*.

Chris Callison-burch. 2003. Active learning for statistical machine translation. In *PhD Proposal, Edinburgh University*.

Trevor Cohn and Mirella Lapata. 2007. Machine translation by triangulation: Making effective use of multi-parallel corpora. In *ACL*.

Le Quan Ha, E. I. Sicilia-Garcia, Ji Ming, and F.J. Smith. 2002. Extension of zipf's law to words and phrases. In *Proceedings of the 19th international conference on Computational linguistics*.

Gholamreza Haffari, Maxim Roy, and Anoop Sarkar. 2009. Active learning for statistical phrase-based machine translation. In *NAACL*.

Xiaodong He, Mei Yang, Jianfeng Gao, Patrick Nguyen, and Robert Moore. 2008. Indirect-hmm-based hypothesis alignment for combining outputs from machine translation systems. In *EMNLP*.

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

Evgeny Matusov, Nicola Ueffing, and Hermann Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *EACL*.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *ACL '02: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*.

Roi Reichart, Katrin Tomanek, Udo Hahn, and Ari Rappoport. 2008. Multi-task active learning for linguistic annotations. In *ACL*.

Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard M. Schwartz, and Bonnie Jean Dorr. 2007. Combining outputs from multiple machine translation systems. In *NAACL*.

Nicola Ueffing, Gholamreza Haffari, and Anoop Sarkar. 2007. Transductive learning for statistical machine translation. In *ACL*.

George Zipf. 1932. *Selective Studies and the Principle of Relative Frequency in Language*. Harvard University Press.

DEPEVAL(summ): Dependency-based Evaluation for Automatic Summaries

Karolina Owczarzak
Information Access Division
National Institute of Standards and Technology
Gaithersburg, MD 20899
karolina.owczarzak@nist.gov

Abstract

This paper presents DEPEVAL(summ), a dependency-based metric for automatic evaluation of summaries. Using a reranking parser and a Lexical-Functional Grammar (LFG) annotation, we produce a set of dependency triples for each summary. The dependency set for each candidate summary is then automatically compared against dependencies generated from model summaries. We examine a number of variations of the method, including the addition of WordNet, partial matching, or removing relation labels from the dependencies. In a test on TAC 2008 and DUC 2007 data, DEPEVAL(summ) achieves comparable or higher correlations with human judgments than the popular evaluation metrics ROUGE and Basic Elements (BE).

1 Introduction

Evaluation is a crucial component in the area of automatic summarization; it is used both to rank multiple participant systems in shared summarization tasks, such as the Summarization track at Text Analysis Conference (TAC) 2008 and its Document Understanding Conference (DUC) predecessors, and to provide feedback to developers whose goal is to improve their summarization systems. However, manual evaluation of a large number of documents necessary for a relatively unbiased view is often unfeasible, especially in the contexts where repeated evaluations are needed. Therefore, there is a great need for reliable automatic metrics that can perform evaluation in a fast and consistent manner.

In this paper, we explore one such evaluation metric, DEPEVAL(summ), based on the comparison of Lexical-Functional Grammar (LFG) dependencies between a candidate summary and one or more model (reference) summaries. The method is similar in nature to Basic Elements (Hovy et al., 2005), in that it extends beyond a simple string comparison of word sequences, reaching instead to a deeper linguistic analysis of the text. Both methods use hand-written extraction rules to derive dependencies from constituent parses produced by widely available Penn II Treebank parsers. The difference between DEPEVAL(summ) and BE is that in DEPEVAL(summ) the dependency extraction is accomplished through an LFG annotation of Cahill et al. (2004) applied to the output of the reranking parser of Charniak and Johnson (2005), whereas in BE (in the version presented here) dependencies are generated by the Minipar parser (Lin, 1995). Despite relying on a the same concept, our approach outperforms BE in most comparisons, and it often achieves higher correlations with human judgments than the string-matching metric ROUGE (Lin, 2004).

A more detailed description of BE and ROUGE is presented in Section 2, which also gives an account of manual evaluation methods employed at TAC 2008. Section 3 gives a short introduction to the LFG annotation. Section 4 describes in more detail DEPEVAL(summ) and its variants. Section 5 presents the experiment in which we compared the perfomance of all three metrics on the TAC 2008 data (consisting of 5,952 100-words summaries) and on the DUC 2007 data (1,620 250-word summaries) and discusses the correlations these metrics achieve. Finally, Section 6 presents conclusions and some directions for future work.

2 Current practice in summary evaluation

In the first Text Analysis Conference (TAC 2008), as well as its predecessor, the Document Understanding Conference (DUC) series, the evaluation

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 190–198,
Suntec, Singapore, 2-7 August 2009.

of summarization tasks was conducted using both manual and automatic methods. Since manual evaluation is still the undisputed gold standard, both at TAC and DUC there was much effort to evaluate manually as much data as possible.

2.1 Manual evaluation

Manual assessment, performed by human judges, usually centers around two main aspects of summary quality: content and form. Similarly to Machine Translation, where these two aspects are represented by the categories of Accuracy and Fluency, in automatic summarization evaluation performed at TAC and DUC they surface as (Content) Responsiveness and Readability. In TAC 2008 (Dang and Owczarzak, 2008), however, Content Responsiveness was replaced by Overall Responsiveness, conflating these two dimensions and reflecting the overall quality of the summary: the degree to which a summary was responding to the information need contained in the topic statement, as well as its linguistic quality. A separate Readability score was still provided, assessing the fluency and structure independently of content, based on such aspects as grammaticality, non-redundancy, referential clarity, focus, structure, and coherence. Both Overall Responsiveness and Readability were evaluated according to a five-point scale, ranging from "Very Poor" to "Very Good".

Content was evaluated manually by NIST assessors using the Pyramid framework (Passonneau et al., 2005). In the Pyramid evaluation, assessors first extract all possible "information nuggets", or Summary Content Units (SCUs) from the four human-crafted model summaries on a given topic. Each SCU is assigned a weight in proportion to the number of model summaries in which it appears, on the assumption that information which appears in most or all human-produced model summaries is more essential to the topic. Once all SCUs are harvested from the model summaries, assessors determine how many of these SCUs are present in each of the automatic peer summaries. The final score for an automatic summary is its total SCU weight divided by the maximum SCU weight available to a summary of average length (where the average length is determined by the mean SCU count of the model summaries for this topic).

All types of manual assessment are expensive and time-consuming, which is why it can be rarely provided for all submitted runs in shared tasks such as the TAC Summarization track. It is also not a viable tool for system developers who ideally would like a fast, reliable, and above all *automatic* evaluation method that can be used to improve their systems. The creation and testing of automatic evaluation methods is, therefore, an important research venue, and the goal is to produce automatic metrics that will correlate with manual assessment as closely as possible.

2.2 Automatic evaluation

Automatic metrics, because of their relative speed, can be applied more widely than manual evaluation. In TAC 2008 Summarization track, all submitted runs were scored with the ROUGE (Lin, 2004) and Basic Elements (BE) metrics (Hovy et al., 2005).

ROUGE is a collection of string-comparison techniques, based on matching *n*-grams between a candidate string and a reference string. The string in question might be a single sentence (as in the case of translation), or a set of sentences (as in the case of summaries). The variations of ROUGE range from matching unigrams (i.e. single words) to matching four-grams, with or without lemmatization and stopwords, with the options of using different weights or skip-*n*-grams (i.e. matching *n*-grams despite intervening words). The two versions used in TAC 2008 evaluations were ROUGE-2 and ROUGE-SU4, where ROUGE-2 calculates the proportion of matching bigrams between the candidate summary and the reference summaries, and ROUGE-SU4 is a combination of unigram match and skip-bigram match with skip distance of 4 words.

BE, on the other hand, employs a certain degree of linguistic analysis in the assessment process, as it rests on comparing the "Basic Elements" between the candidate and the reference. Basic Elements are syntactic in nature, and comprise the heads of major syntactic constituents in the text (noun, verb, adjective, etc.) and their modifiers in a dependency relation, expressed as a triple (head, modifier, relation type). First, the input text is parsed with a syntactic parser, then Basic Elements are extracted from the resulting parse, and the candidate BEs are matched against the reference BEs. In TAC 2008 and DUC 2008 evaluations the BEs were extracted with Minipar (Lin, 1995). Since BE, contrary to ROUGE, does not

rely solely on the surface sequence of words to determine similarity between summaries, but delves into what could be called a shallow semantic structure, comprising thematic roles such as subject and object, it is likely to notice identity of meaning where such identity is obscured by variations in word order. In fact, when it comes to evaluation of automatic summaries, BE shows higher correlations with human judgments than ROUGE, although the difference is not large enough to be statistically significant. In the TAC 2008 evaluations, BE-HM (a version of BE where the words are stemmed and the relation type is ignored) obtained a correlation of 0.911 with human assessment of overall responsiveness and 0.949 with the Pyramid score, whereas ROUGE-2 showed correlations of 0.894 and 0.946, respectively.

While using dependency information is an important step towards integrating linguistic knowledge into the evaluation process, there are many ways in which this could be approached. Since this type of evaluation processes information in stages (constituent parser, dependency extraction, and the method of dependency matching between a candidate and a reference), there is potential for variance in performance among dependency-based evaluation metrics that use different components. Therefore, it is interesting to compare our method, which relies on the Charniak-Johnson parser and the LFG annotation, with BE, which uses Minipar to parse the input and produce dependencies.

3 Lexical-Functional Grammar and the LFG parser

The method discussed in this paper rests on the assumptions of Lexical-Functional Grammar (Kaplan and Bresnan, 1982; Bresnan, 2001) (LFG). In LFG sentence structure is represented in terms of c(onstituent)-structure and f(unctional)-structure. C-structure represents the word order of the surface string and the hierarchical organisation of phrases in terms of trees. F-structures are recursive feature structures, representing abstract grammatical relations such as subject, object, oblique, adjunct, etc., approximating to predicate-argument structure or simple logical forms. C-structure and f-structure are related by means of functional annotations in c-structure trees, which describe f-structures.

While c-structure is sensitive to surface rearrangement of constituents, f-structure abstracts away from (some of) the particulars of surface realization. The sentences *John resigned yesterday* and *Yesterday, John resigned* will receive different tree representations, but identical f-structures. The f-structure can also be described in terms of a flat set of triples, or dependencies. In triples format, the f-structure for these two sentences is represented in 1.

(1)
subject(resign,john)
person(john,3)
number(john,sg)
tense(resign,past)
adjunct(resign,yesterday)
person(yesterday,3)
number(yesterday,sg)

Cahill et al. (2004), in their presentation of LFG parsing resources, distinguish 32 types of dependencies, divided into two major groups: a group of predicate-only dependencies and non-predicate dependencies. Predicate-only dependencies are those whose path ends in a predicate-value pair, describing grammatical relations. For instance, in the sentence *John resigned yesterday*, predicate-only dependencies would include: *subject*(resign, john) and *adjunct*(resign, yesterday), while non-predicate dependencies are *person*(john,3), *number*(john,sg), *tense*(resign,past), *person*(yesterday,3), *num*(yesterday,sg). Other predicate-only dependencies include: *apposition*, *complement*, *open complement*, *coordination*, *determiner*, *object*, *second object*, *oblique*, *second oblique*, *oblique agent*, *possessive*, *quantifier*, *relative clause*, *topic*, and *relative clause pronoun*. The remaining non-predicate dependencies are: *adjectival degree*, *coordination surface form*, *focus*, complementizer forms: *if*, *whether*, and *that*, *modal*, *verbal particle*, *participle*, *passive*, *pronoun surface form*, and *infinitival clause*.

These 32 dependencies, produced by LFG annotation, and the overlap between the set of dependencies derived from the candidate summary and the reference summaries, form the basis of our evaluation method, which we present in Section 4.

First, a summary is parsed with the Charniak-Johnson reranking parser (Charniak and Johnson, 2005) to obtain the phrase-structure tree. Then, a sequence of scripts annotates the output, translating the relative phrase position into f-structural dependencies. The treebank-based LFG annotation used in this paper and developed by Cahill et al. (2004) obtains high precision and recall rates. As reported in Cahill et al. (2008), the version of

the LFG parser which applies the LFG annotation algorithm to the earlier Charniak's parser (Charniak, 2000) obtains an f-score of 86.97 on the Wall Street Journal Section 23 test set. The LFG parser is robust as well, with coverage levels exceeding 99.9%, measured in terms of complete spanning parse.

4 Dependency-based evaluation

Our dependency-based evaluation method, similarly to BE, compares two unordered sets of dependencies: one bag contains dependencies harvested from the candidate summary and the other contains dependencies from one or more reference summaries. Overlap between the candidate bag and the reference bag is calculated in the form of precision, recall, and the f-measure (with precision and recall equally weighted). Since for ROUGE and BE the only reported score is recall, we present recall results here as well, calculated as in 2:

$$(2) \quad \text{DEPEVAL(summ) Recall} = \frac{|D_{cand}| \cap |D_{ref}|}{|D_{ref}|}$$

where D_{cand} are the candidate dependencies and D_{ref} are the reference dependencies.

The dependency-based method using LFG annotation has been successfully employed in the evaluation of Machine Translation (MT). In Owczarzak (2008), the method achieves equal or higher correlations with human judgments than METEOR (Banerjee and Lavie, 2005), one of the best-performing automatic MT evaluation metrics. However, it is not clear that the method can be applied without change to the task of assessing automatic summaries; after all, the two tasks - of summarization and translation - produce outputs that are different in nature. In MT, the unit of text is a sentence; text is translated, and the translation evaluated, sentence by sentence. In automatic summarization, the output unit is a summary with length varying depending on task, but which most often consists of at least several sentences. This has bearing on the matching process: with several sentences on the candidate and reference side each, there is increased possibility of trivial matches, such as dependencies containing function words, which might inflate the summary score even in the absence of important content. This is particularly likely if we were to employ partial matching for dependencies. Partial matching (indicated in the result tables with the tag **pm**) "splits" each predicate dependency into two, replacing one or the other element with a variable, e.g. for the dependency *subject*(resign, John) we would obtain two partial dependencies *subject*(resign, x) and *subject*(x, John). This process helps circumvent some of the syntactic and lexical variation between a candidate and a reference, and it proved very useful in MT evaluation (Owczarzak, 2008). In summary evaluation, as will be shown in Section 5, it leads to higher correlations with human judgments only in the case of human-produced model summaries, because almost any variation between two model summaries is "legal", i.e. either a paraphrase or another, but equally relevant, piece of information. For automatic summaries, which are of relatively poor quality, partial matching lowers our method's ability to reflect human judgment, because it results in overly generous matching in situations where the examined information is neither a paraphrase nor relevant.

Similarly, evaluating a summary against the union of all references, as we do in the baseline version of our method, increases the pool of possible matches, but may also produce score inflation through matching repetitive information across models. To deal with this, we produce a version of the score (marked in the result tables with the tag **one**) that counts only one "hit" for every dependency match, independent of how many instances of a given dependency are present in the comparison.

The use of WordNet[1] module (Rennie, 2000) did not provide a great advantage (see results tagged with **wn**), and sometimes even lowered our correlations, especially in evaluation of automatic systems. This makes sense if we take into consideration that WordNet lists all possible synonyms for all possible senses of a word, and so, given a great number of cross-sentence comparisons in multi-sentence summaries, there is an increased risk of spurious matches between words which, despite being potentially synonymous in certain contexts, are not equivalent in the text.

Another area of concern was the potential noise introduced by the parser and the annotation process. Due to parsing errors, two otherwise equivalent expressions might be encoded as differing sets of dependencies. In MT evaluation, the dependency-based method can alleviate parser

[1] http://wordnet.princeton.edu/

noise by comparing *n*-best parses for the candidate and the reference (Owczarzak et al., 2007), but this is not an efficient solution for comparing multi-sentence summaries. We have therefore attempted to at least partially counteract this issue by removing relation labels from the dependencies (i.e. producing dependencies of the form (resign, John) instead of *subject*(resign, John)), which did provide some improvement (see results tagged with **norel**).

Finally, we experimented with a predicate-only version of the evaluation, where only the predicate dependencies participate in the comparison, excluding dependencies that provide purely grammatical information such as person, tense, or number (tagged in the results table as **pred**). This move proved beneficial only in the case of system summaries, perhaps by decreasing the number of trivial matches, but decreased the method's correlation for model summaries, where such detailed information might be necessary to assess the degree of similarity between two human summaries.

5 Experimental results

The first question we have to ask is: which of the manual evaluation categories do we want our metric to imitate? It is unlikely that a single automatic measure will be able to correctly reflect both Readability and Content Responsiveness, as form and content are separate qualities and need different measures. Content seems to be the more important aspect, especially given that Readability can be partially derived from Responsiveness (a summary high in content cannot be very low in readability, although some very readable summaries can have little relevant content). Content Responsiveness was provided in DUC 2007 data, but not in TAC 2008, where the extrinsic Pyramid measure was used to evaluate content. It is, in fact, preferable to compare our metric against the Pyramid score rather than Content Responsiveness, because both the Pyramid and our method aim to measure the degree of similarity between a candidate and a model, whereas Content Responsiveness is a direct assessment of whether the summary's content is adequate given a topic and a source text. The Pyramid is, at the same time, a costly manual evaluation method, so an automatic metric that successfully emulates it would be a useful replacement.

Another question is whether we focus on system-level or summary-level evaluation. The correlation values at the summary-level are generally much lower than on the system-level, which means the metrics are better at evaluating system performance than the quality of individual summaries. System-level evaluations are essential to shared summarization tasks; summary-level assessment might be useful to developers who want to test the effect of particular improvements in their system. Of course, the ideal evaluation metric would show high correlations with human judgment on both levels.

We used the data from the TAC 2008 and DUC 2007 Summarization tracks. The first set comprised 58 system submissions and 4 human-produced model summaries for each of the 96 subtopics (there were 48 topics, each of which required two summaries: a main and an update summary), as well as human-produced Overall Responsiveness and Pyramid scores for each summary. The second set included 32 system submissions and 4 human models for each of the 45 topics. For fair comparison of models and systems, we used jackknifing: while each model was evaluated against the remaining three models, each system summary was evaluated four times, each time against a different set of three models, and the four scores were averaged.

5.1 System-level correlations

Table 1 presents system-level Pearson's correlations between the scores provided by our dependency-based metric DEPEVAL(summ), as well as the automatic metrics ROUGE-2, ROUGE-SU4, and BE-HM used in the TAC evaluation, and the manual Pyramid scores, which measured the content quality of the systems. It also includes correlations with the manual Overall Responsiveness score, which reflected both content and linguistic quality. Table 3 shows the correlations with Content Responsiveness for DUC 2007 data for ROUGE, BE, and those few select versions of DEPEVAL(summ) which achieve optimal results on TAC 2008 data (for a more detailed discussion of the selection see Section 6).

The correlations are listed for the following versions of our method: **pm** - partial matching for dependencies; **wn** - WordNet; **pred** - matching predicate-only dependencies; **norel** - ignoring dependency relation label; **one** - counting a match only once irrespective of how many instances of

TAC 2008	Pyramid		Overall Responsiveness	
Metric	models	systems	models	systems
DEPEVAL(summ): Variations				
base	0.653	0.931	0.883	0.862
pm	0.690	0.811	0.943	0.740
wn	0.687	0.929	0.888	0.860
pred	0.415	0.946	0.706	0.909
norel	0.676	0.929	0.880	0.861
one	0.585	0.958*	0.858	0.900
DEPEVAL(summ): Combinations				
pm wn	0.694	0.903	0.952*	0.839
pm pred	0.534	0.880	0.898	0.831
pm norel	0.722	0.907	0.936	0.835
pm one	0.611	0.950	0.876	0.895
wn pred	0.374	0.946	0.716	0.912
wn norel	0.405	0.941	0.752	0.905
wn one	0.611	0.952	0.856	0.897
pred norel	0.415	0.945	0.735	0.905
pred one	0.415	0.953	0.721	0.921*
norel one	0.600	0.958*	0.863	0.900
pm wn pred	0.527	0.870	0.905	0.821
pm wn norel	0.738	0.897	0.931	0.826
pm wn one	0.634	0.936	0.887	0.881
pm pred norel	0.642	0.876	0.946	0.815
pm pred one	0.504	0.948	0.817	0.907
pm norel one	0.725	0.941	0.905	0.880
wn pred norel	0.433	0.944	0.764	0.906
wn pred one	0.385	0.950	0.722	0.919
wn norel one	0.632	0.954	0.872	0.896
pred norel one	0.452	0.955	0.756	0.919
pm wn pred norel	0.643	0.861	0.940	0.800
pm wn pred one	0.486	0.932	0.809	0.890
pm pred norel one	0.711	0.939	0.881	0.891
pm wn norel one	0.743*	0.930	0.902	0.870
wn pred norel one	0.467	0.950	0.767	0.918
pm wn pred norel one	0.712	0.927	0.887	0.880
Other metrics				
ROUGE-2	0.277	0.946	0.725	0.894
ROUGE-SU4	0.457	0.928	0.866	0.874
BE-HM	0.423	0.949	0.656	0.911

Table 1: System-level Pearson's correlation between automatic and manual evaluation metrics for TAC 2008 data.

a particular dependency are present in the candidate and reference. For each of the metrics, including ROUGE and BE, we present the correlations for recall. The highest result in each category is marked by an asterisk. The background gradient indicates whether DEPEVAL(summ) correlation is higher than all three competitors ROUGE-2, ROUGE-SU4, and BE (darkest grey), two of the three (medium grey), one of the three (light grey), or none (white). The 95% confidence intervals are not included here for reasons of space, but their comparison suggests that none of the system-level differences in correlation levels are large enough to be significant. This is because the intervals themselves are very wide, due to relatively small number of summarizers (58 automatic and 8 human for TAC; 32 automatic and 10 human for DUC) involved in the comparison.

5.2 Summary-level correlations

Tables 2 and 4 present the same correlations, but this time on the level of individual summaries. As before, the highest level in each category is marked by an asterisk. Contrary to system-level, here some correlations obtained by DEPEVAL(summ) are significantly higher than those achieved by the three competing metrics, ROUGE-2, ROUGE-SU4, and BE-HM, as determined by the confidence intervals. The letters in parenthesis indicate that a given DEPEVAL(summ) variant is significantly better at correlating with human judgment than ROUGE-2 (= R2), ROUGE-SU4 (= R4), or BE-HM (= B).

6 Discussion and future work

It is obvious that none of the versions performs best across the board; their different characteristics might render them better suited either for models or for automatic systems, but not for both at the same time. This can be explained if we understand that evaluating human gold standard summaries and automatically generated summaries of poor-to-medium quality is, in a way, not the same task. Given that human models are by default well-formed and relevant, relaxing any restraints on matching between them (i.e. allowing partial dependencies, removing the relation label, or adding synonyms) serves, in effect, to accept as correct either (1) the same conceptual information expressed in different ways (where the difference might be real or introduced by faulty parsing), or (2) other information, yet still relevant to the topic. Accepting information of the former type as correct will ratchet up the score for the summary and the correlation with the summary's Pyramid score, which measures identity of information across summaries. Accepting the first *and* second type of information will raise the score and the correlation with Responsiveness, which measures relevance of information to the particular topic. However, in evaluating system summaries such relaxation of matching constraints will result in accepting irrelevant and ungrammatical information as correct, driving up the DEPEVAL(summ) score, but lowering its correlation with both Pyramid and Responsiveness. In simple words, it is okay to give a model summary "the benefit of doubt", and accept its content as correct even if it is not matching other model summaries exactly, but the same strategy applied to a system summary might cause mass over-estimation of the summary's quality.

This substantial difference in the nature of human-generated models and system-produced summaries has impact on all automatic means of evaluation, as long as we are limited to methods that operate on more shallow levels than a full

TAC 2008	Pyramid		Overall Responsiveness	
Metric	models	systems	models	systems
DEPEVAL(summ): Variations				
base	0.436 (B)	0.595 (R2,R4,B)	0.186	0.373 (R2,B)
pm	0.467 (B)	0.584 (R2,B)	0.183	0.368 (B)
wn	0.448 (B)	0.592 (R2,B)	0.192	0.376 (R2,R4,B)
pred	0.344	0.543 (B)	0.170	0.327
norel	0.437 (B)	0.596* (R2,R4,B)	0.186	0.373 (R2,B)
one	0.396	0.587 (R2,B)	0.171	0.376 (R2,R4,B)
DEPEVAL(summ): Combinations				
pm wn	0.474 (B)	0.577 (R2,B)	0.194*	0.371 (R2,B)
pm pred	0.407	0.537 (B)	0.153	0.337
pm norel	0.483 (R2,B)	0.584 (R2,B)	0.168	0.362
pm one	0.402	0.577 (R2,B)	0.167	0.384 (R2,R4,B)
wn pred	0.352	0.537 (B)	0.182	0.328
wn norel	0.364	0.541 (B)	0.187	0.329
wn one	0.411	0.581 (R2,B)	0.182	0.384 (R2,R4,B)
pred norel	0.351	0.547 (B)	0.169	0.327
pred one	0.325	0.542 (B)	0.171	0.347
norel one	0.403	0.589 (R2,B)	0.176	0.377 (R2,R4,B)
pm wn pred	0.415	0.526 (B)	0.167	0.337
pm wn norel	0.488* (R2,R4,B)	0.576 (R2,B)	0.168	0.366 (B)
pm wn one	0.417	0.563 (B)	0.179	0.389* (R2,R4.B)
pm pred norel	0.433 (B)	0.538 (B)	0.124	0.333
pm pred one	0.357	0.545 (B)	0.151	0.381 (R2,R4,B)
pm norel one	0.437 (B)	0.567 (R2,B)	0.174	0.369 (B)
wn pred norel	0.353	0.541 (B)	0.180	0.324
wn pred one	0.328	0.535 (B)	0.179	0.346
wn norel one	0.416	0.584 (R2,B)	0.185	0.385 (R2,R4,B)
pred norel one	0.336	0.549 (B)	0.169	0.351
pm wn pred norel	0.428 (B)	0.524 (B)	0.120	0.334
pm wn pred one	0.363	0.525 (B)	0.164	0.380 (R2,R4,B)
pm pred norel one	0.420 (B)	0.533 (B)	0.154	0.375 (R2,R4,B)
pm wn norel one	0.452 (B)	0.558 (B)	0.179	0.376 (R2,R4,B)
wn pred norel one	0.338	0.544 (B)	0.178	0.349
pm wn pred norel one	0.427 (B)	0.522 (B)	0.153	0.379 (R2,R4,B)
Other metrics				
ROUGE-2	0.307	0.527	0.098	0.323
ROUGE-SU4	0.318	0.557	0.153	0.327
BE-HM	0.239	0.456	0.135	0.317

Table 2: Summary-level Pearson's correlation between automatic and manual evaluation metrics for TAC 2008 data.

DUC 2007	Content Responsiveness	
Metric	models	systems
DEPEVAL(summ)	0.7341	0.8429
DEPEVAL(summ) wn	0.7355	0.8354
DEPEVAL(summ) norel	0.7394	0.8277
DEPEVAL(summ) one	0.7507	0.8634
ROUGE-2	0.4077	0.8772
ROUGE-SU4	0.2533	0.8297
BE-HM	0.5471	0.8608

Table 3: System-level Pearson's correlation between automatic metrics and Content Responsiveness for DUC 2007 data. For model summaries, only DEPEVAL correlations are significant (the 95% confidence interval does not include zero). None of the differences between metrics are significant at the 95% level.

DUC 2007	Content Responsiveness	
Metric	models	systems
DEPEVAL(summ)	0.2059	0.4150
DEPEVAL(summ) wn	0.2081	0.4178
DEPEVAL(summ) norel	0.2119	0.4185
DEPEVAL(summ) one	0.1999	0.4101
ROUGE-2	0.1501	0.3875
ROUGE-SU4	0.1397	0.4264
BE-HM	0.1330	0.3722

Table 4: Summary-level Pearson's correlation between automatic metrics and Content Responsiveness for DUC 2007 data. ROUGE-SU4 and BE correlations for model summaries are not statistically significant. None of the differences between metrics are significant at the 95% level.

semantic and pragmatic analysis against human-level world knowledge. The problem is twofold: first, our automatic metrics measure *identity* rather than *quality*. Similarity of content between a candidate summary and one or more references is acting as a proxy measure for the quality of the candidate summary; yet, we cannot forget that the relation between these two features is not purely linear. A candidate highly similar to the reference will be, necessarily, of good quality, but a candidate which is dissimilar from a reference is not necessarily of low quality (*vide* the case of parallel model summaries, which almost always contain some non-overlapping information).

The second problem is the extent to which our metrics are able to distinguish content through the veil of differing forms. Synonyms, paraphrases, or pragmatic features such as the choice of topic and focus render simple string-matching techniques ineffective, especially in the area of summarization where the evaluation happens on a supra-sentential level. As a result, then, a lot of effort was put into developing metrics that can identify similar content despite non-similar form, which naturally led to the application of linguistically-oriented approaches that look beyond surface word order.

Essentially, though, we are using imperfect measures of similarity as an imperfect stand-in for quality, and the accumulated noise often causes a divergence in our metrics' performance with model and system summaries. Much like the inverse relation of precision and recall, changes and additions that improve a metric's correlation with human scores for model summaries often weaken the correlation for system summaries, and vice versa. Admittedly, we could just ignore this problem and focus on increasing correlations for automatic summaries only; after all, the whole point of creating evaluation metrics is to score and rank the output of systems. Such a perspective can be rather short-sighted, though, given that we expect continuous improvement from the summarization systems to, ideally, human levels, so the same issues which now prevent high correlations for models will start surfacing in evaluation of system-produced summaries as well. Using metrics that only perform reliably for low-quality summaries might prevent us from noticing when those summaries become better. Our goal should be, therefore, to develop a metric which obtains high correlations in both categories, with the assumption that such a metric will be more reliable in evaluating summaries of varying quality.

Since there is no single winner among all 32 variants of DEPEVAL(summ) on TAC 2008 data, we must decide which of the categories is most important to a successful automatic evaluation metric. Correlations with Overall Responsiveness are in general lower than those with the Pyramid score (except in the case of system-level models). This makes sense, if we rememeber that Overall Responsiveness judges content as well as linguistic quality, which are two different dimensions and so a single automatic metric is unlikely to reflect it well, and that it judges content in terms of its relevance to topic, which is also beyond the reach of contemporary metrics which can at most judge content similarity to a model. This means that the Pyramid score makes for a more relevant metric to emulate.

The last dilemma is whether we choose to focus on system- or summary-level correlations. This ties in with the purpose which the evaluation metric should serve. In comparisons of multiple systems, such as in TAC 2008, the value is placed in the correct ordering of these systems; while summary-level assessment can give us important feedback and insight during the system development stage.

The final choice among all DEPEVAL(summ) versions hinges on all of these factors: we should prefer a variant which correlates highly with the Pyramid score rather than with Responsiveness, which minimizes the gap between model and automatic peer correlations while retaining relatively high values for both, and which fulfills these requirements similarly well on both summary- and system-levels. Three such variants are the baseline DEPEVAL(summ), the WordNet version DEPEVAL(summ) **wn**, and the version with removed relation labels DEPEVAL(summ) **norel**. Both the baseline and **norel** versions achieve significant improvement over ROUGE and BE in correlations with the Pyramid score for automatic summaries, and over BE for models, on the summary level. In fact, almost in all categories they achieve higher correlations than ROUGE and BE. The only exceptions are the correlations with Pyramid for systems at the system-level, but there the results are close and none of the differences in that category are significant. To balance this exception, DEPEVAL(summ) achieves much higher correlations with the Pyramid scores for model summaries than either ROUGE or BE on the system level.

In order to see whether the DEPEVAL(summ) advantage holds for other data, we examined the most optimal versions (baseline, **wn**, **norel**, as well as **one**, which is the closest counterpart to label-free BE-HM) on data from DUC 2007. Because only a portion of the DUC 2007 data was evaluated with Pyramid, we chose to look rather at the Content Responsiveness scores. As can be seen in Tables 3 and 4, the same patterns hold: decided advantage over ROUGE/BE when it comes to model summaries (especially at system-level), comparable results for automatic summaries. Since DUC 2007 data consisted of fewer summaries (1,620 vs 5,952 at TAC) and fewer submissions (32 vs 57 at TAC), some results did not reach statistical significance. In Table 3, in the models category, only DEPEVAL(summ) correlations are significant. In Table 4, in the model category, only DEPEVAL(summ) and ROUGE-2 correlations are significant. Note also that these correlations with Content Responsiveness are generally lower than those with Pyramid in previous tables, but in the case of summary-level comparison higher than the correlations with Overall Responsiveness. This is to be expected given our earlier discussion of the differences in what these metrics measure.

As mentioned before, the dependency-based evaluation can be approached from different angles, leading to differences in performance. This is exemplified in our experiment, where DEPEVAL(summ) outperforms BE, even though both these metrics rest on the same general idea. The new implementation of BE presented at the TAC 2008 workshop (Tratz and Hovy, 2008) introduces transformations for dependencies in order to increase the number of matches among elements that are semantically similar yet differ in terms of syntactic structure and/or lexical choices, and adds WordNet for synonym matching. Its core modules were updated as well: Minipar was replaced with the Charniak-Johnson reranking parser (Charniak and Johnson, 2005), Named Entity identification was added, and the BE extraction is conducted using a set of Tregex rules (Levy and Andrew, 2006). Since our method, presented in this paper, also uses the reranking parser, as well as WordNet, it would be interesting to compare both methods directly in terms of the performance of the dependency extraction procedure.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*, pages 65–73, Ann Arbor, MI, USA.

Joan Bresnan. 2001. *Lexical-Functional Syntax*. Blackwell, Oxford.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In *Proceedings of the 42th Annual Meeting of the Association for Computational Linguistics*, pages 320–327, Barcelona, Spain.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Stefan Riezler, Josef van Genabith, and Andy Way. 2008. Wide-coverage deep statistical parsing using automatic dependency structure annotation. *Comput. Linguist.*, 34(1):81–124.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *ACL 2005: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 173–180, Morristown, NJ, USA. Association for Computational Linguistics.

Eugene Charniak. 2000. A maximum entropy inspired parser. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139, Seattle, WA, USA.

Hoa Trang Dang and Karolina Owczarzak. 2008. Overview of the tac 2008 summarization track: Update task. In *to appear in: Proceedings of the 1st Text Analysis Conference (TAC)*.

Eduard Hovy, Chin-Yew Lin, and Liang Zhou. 2005. Evaluating DUC 2005 using Basic Elements. In *Proceedings of the 5th Document Understanding Conference (DUC)*.

Ronald M. Kaplan and Joan Bresnan, 1982. *The Mental Representation of Grammatical Relations*, chapter Lexical-functional Grammar: A Formal System for Grammatical Representation. MIT Press, Cambridge, MA, USA.

Roger Levy and Galen Andrew. 2006. Tregex and tsurgeon: Tools for querying and manipulating tree data structures. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*.

Dekang Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1420–1427.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of the ACL 2004 Workshop: Text Summarization Branches Out*, pages 74–81.

Karolina Owczarzak, Josef van Genabith, and Andy Way. 2007. Evaluating Machine Translation with LFG dependencies. *Machine Translation*, 21(2):95–119.

Karolina Owczarzak. 2008. *A novel dependency-based evaluation metric for Machine Translation*. Ph.D. thesis, Dublin City University.

Rebecca J. Passonneau, Ani Nenkova, Kathleen McKeown, and Sergey Sigelman. 2005. Applying the Pyramid method in DUC 2005. In *Proceedings of the 5th Document Understanding Conference (DUC)*.

Jason Rennie. 2000. Wordnet::querydata: a Perl module for accessing the WordNet database. http://people.csail.mit.edu/ jrennie/WordNet.

Stephen Tratz and Eduard Hovy. 2008. Summarization evaluation using transformed Basic Elements. In *Proceedings of the 1st Text Analysis Conference (TAC)*.

Summarizing Definition from Wikipedia

Shiren Ye and **Tat-Seng Chua** and **Jie Lu**
Lab of Media Search
National University of Singapore
{yesr|chuats|luj}@comp.nus.edu.sg

Abstract

Wikipedia provides a wealth of *knowledge*, where the first sentence, infobox (and relevant sentences), and even the entire document of a wiki article could be considered as diverse versions of summaries (definitions) of the target topic. We explore how to generate a series of summaries with various lengths based on them. To obtain more reliable associations between sentences, we introduce wiki concepts according to the internal links in Wikipedia. In addition, we develop an extended document concept lattice model to combine wiki concepts and non-textual features such as the outline and infobox. The model can concatenate representative sentences from non-overlapping salient local topics for summary generation. We test our model based on our annotated wiki articles which topics come from TREC-QA 2004-2006 evaluations. The results show that the model is effective in summarization and definition QA.

1 Introduction

Nowadays, 'ask Wikipedia' has become as popular as 'Google it' during Internet surfing, as Wikipedia is able to provide reliable information about the concept (entity) that the users want. As the largest online encyclopedia, Wikipedia assembles immense human knowledge from thousands of volunteer editors, and exhibits significant contributions to NLP problems such as semantic relatedness, word sense disambiguation and question answering (QA).

For a given definition query, many search engines (e.g., specified by '*define:*' in Google) often place the first sentence of the corresponding wiki[1] article at the top of the returned list. The use of one-sentence snippets provides a brief and concise description of the query. However, users often need more information beyond such a one-sentence definition, while feeling that the corresponding wiki article is too long. Thus, there is a strong demand to summarize wiki articles as definitions with various lengths to suite different user needs.

The initial motivation of this investigation is to find better definition answer for TREC-QA task using Wikipedia (Kor and Chua, 2007). According to past results on TREC-QA (Voorhees, 2004; Voorhees and Dang, 2005), definition queries are usually recognized as being more difficult than factoid and list queries. Wikipedia could help to improve the quality of answer finding and even provide the answers directly. Its results are better than other external resources such as WordNet, Gazetteers and Google's *define* operator, especially for definition QA (Lita et al., 2004).

Different from the *free* text used in QA and summarization, a wiki article usually contains valuable information like infobox and wiki link. **Infobox** tabulates the key properties about the target, such as birth place/date and spouse for a person as well as type, founder and products for a company. Infobox, as a form of thumbnail biography, can be considered as a mini version of a wiki article's summary. In addition, the relevant concepts existing in a wiki article usually refer to other wiki pages by wiki internal links, which will form a close set of reference relations. The current Wikipedia recursively defines over 2 million concepts (in English) via **wiki links**. Most of these concepts are multi-word terms, whereas WordNet has only 50,000 plus multi-word terms. Any term could appear in the definition of a concept if necessary, while the total vocabulary existing in WordNet's glossary definition is less than 2000. Wikipedia addresses explicit semantics for numerous concepts. These special *knowledge* representations will provide additional information for analysis and summarization. We thus need to extend existing summarization technologies to take advantage of the *knowledge* representations in Wikipedia.

[1] For readability, we follow the upper/lower case rule on *web* (say, 'web pages' and 'on the Web'), and utilize 'wiki(pedia) articles' and 'on (the) Wikipedia', the latter referring to the entire Wikipedia.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 199–207,
Suntec, Singapore, 2-7 August 2009.

The goal of this investigation is to explore summaries with different lengths in Wikipedia. Our main contribution lies in developing a summarization method that can (i) explore more reliable associations between passages (sentences) in huge feature space represented by wiki concepts; and (ii) effectively combine textual and non-textual features such as infobox and outline in Wikipedia to generate summaries as definition.

The rest of this paper is organized as follows: In the next section, we discuss the background of summarization using both textual and structural features. Section 3 presents the extended document concept lattice model for summarizing wiki articles. Section 4 describes corpus construction and experiments are described; while Section 5 concludes the paper.

2 Background

Besides some heuristic rules such as sentence position and cue words, typical summarization systems measure the associations (links) between sentences by term repetitions (e.g., LexRank (Erkan and Radev, 2004)). However, sophisticated authors usually utilize synonyms and paraphrases in various forms rather than simple term repetitions. Furnas et al. (1987) reported that two people choose the same main key word for a single well-known object less than 20% of the time. A case study by Ye et al. (2007) showed that 61 different words existing in 8 relevant sentences could be mapped into 16 distinctive concepts by means of grouping terms with close semantic (such as *[British, Britain, UK]* and *[war, fought, conflict, military]*). However, most existing summarization systems only consider the repeated words between sentences, where latent associations in terms of inter-word synonyms and paraphrases are ignored. The incomplete data likely lead to unreliable sentence ranking and selection for summary generation.

To recover the hidden associations between sentences, Ye et al. (2007) compute the semantic similarity using WordNet. The term pairs with semantic similarity higher than a predefined threshold will be grouped together. They demonstrated that collecting more links between sentences will lead to better summarization as measured by ROUGE scores, and such systems were rated among the top systems in DUC (document understanding conference) in 2005 and 2006. This WordNet-based approach has several shortcomings due to the problems of data deficiency and word sense ambiguity, etc.

Wikipedia already defined millions of multi-word concepts in separate articles. Its definition is much larger than that of WordNet. For instance, more than 20 kinds of songs and movies called Butterfly , such as *Butterfly_(Kumi_Koda_song)*, *Butterfly_(1999_film)* and *Butterfly_(2004_film)*, are listed in Wikipedia. When people say something about butterfly in Wikipedia, usually, a link is assigned to refer to a particular butterfly. Following this link, we can acquire its explicit and exact semantic (Gabrilovich and Markovitch, 2007), especially for multi-word concepts. Phrases are more important than individual words for document retrieval (Liu et al., 2004). We hope that the wiki concepts are appropriate text representation for summarization.

Generally, wiki articles have little redundancy in their contents as they utilize encyclopedia style. Their authors tend to use wiki links and '*See Also*' links to refer to the involved concepts rather than expand these concepts. In general, the guideline for composing wiki articles is to avoid overlong and over-complicated styles. Thus, the strategy of '*split it*' into a series of articles is recommended; so wiki articles are usually not too long and contain limited number of sentences. These factors lead to fewer *links* between sentences within a wiki article, as compared to normal documents. However, the principle of typical extractive summarization approaches is that the sentences whose contents are repeatedly emphasized by the authors are most important and should be included (Silber and McCoy, 2002). Therefore, it is challenging to summarize wiki articles due to low redundancy (and links) between sentences. To overcome this problem, we seek (i) more reliable links between passages, (ii) appropriate weighting metric to emphasize the salient concepts about the topic, and (iii) additional guideline on utilizing non-textual features such as outline and infobox. Thus, we develop wiki concepts to replace 'bag-of-words' approach for better link measurements between sentences, and extend an existing summarization model on free text to integrate structural information.

By analyzing rhetorical discourse structure of aim, background, solution, etc. or citation context, we can obtain appropriate abstracts and the most influential contents from scientific articles (Teufel and Moens, 2002; Mei and Zhai, 2008). Similarly, we believe that the structural information such as infobox and outline is able to improve summarization as well. The outline of a wiki article using inner links will render the structure of its definition. In addition, infobox could be considered as topic signature (Lin and Hovy, 2000) or keywords about the topic. Since keywords and summary of a document can be mutually boosted (Wan et al., 2007), infobox is capable of summarization instruction.

When Ahn (2004) and Kor (2007) utilize Wikipedia for TREC-QA definition, they treat the Wikipedia as the Web and perform normal search on it. High-frequency terms in the query snippets returned from wiki index are used to extend query and rank (re-rank) passages. These snippets usually

come from multiple wiki articles. Here the useful information may be beyond these snippets but existing terms are possibly irrelevant to the topic. On the contrary, our approach concentrates on the wiki article having the exact topic only. We assume that every sentence in the article is used to define the query topic, no matter whether it contains the term(s) of the topic or not. In order to extract some salient sentences from the article as definition summaries, we will build a summarization model that describes the relations between the sentences, where both textual and structural features are considered.

3 Our Approach

3.1 Wiki Concepts

In this subsection, we address how to find reasonable and reliable links between sentences using wiki concepts.

Consider a sentence: *'After graduating from Boston University in 1988, she went to work at a Calvin Klein store in Boston.'* from a wiki article *'Carolyn Bessette Kennedy'*[2], we can find 11 distinctive terms, such as *after, graduate, Boston, University,1988, go, work, Calvin, Klein, store, Boston*, if stop words are ignored.

However, multi-word terms such as *Boston University* and *Calvin Klein* are linked to the corresponding wiki articles, where their definitions are given. Clearly, considering the anchor texts as two ***wiki concepts*** rather than four words is more reasonable. Their granularity are closer to semantic content units in a summarization evaluation method Pyramid (Nenkova et al., 2007) and nuggets in TREC-QA . When the text is represented by wiki concepts, whose granularity is similar to the evaluation units, it is possibly easy to detect the matching output using a model. Here,

- Two separate words, *Calvin* and *Klein*, are meaningless and should be discarded; otherwise, spurious links between sentences are likely to occur.
- *Boston University* and *Boston* are processed separately, as they are different named entities. No link between them is appropriate[3].
- Terms such as *'John F. Kennedy, Jr.'* and *'John F. Kennedy'* will be considered as two diverse wiki concepts, but we do not account on how many repeated words there are.
- Different anchor texts, such as *U.S.A.* and *United States of America*, are recognized as an identical concept since they refer to the same wiki article.
- Two concepts, such as *money* and *cash*, will be merged into an identical concept when their semantics are similar.

In wiki articles, the first occurrence of a wiki concept is tagged by a wiki link, but there is no such a link to its subsequent occurrences in the remaining parts of the text in most cases. To alleviate this problem, a set of heuristic rules is proposed to unify the subsequent occurrences of concepts in normal text with previous wiki concepts in the anchor text. These heuristic rules include: (i) edit distance between linked wiki concept and candidates in normal text is larger than a predefined threshold; and (ii) partially overlapping words beginning with capital letter, etc.

After filtering out wiki concepts, the words remaining in wiki articles could be grouped into two sets: close-class terms like pronouns and prepositions as well as open-class terms like nouns and verbs. For example, in the sentence *'She died at age 33, along with her husband and sister'*, the open-class terms include *die*, *age*, *33*, *husband* and *sister*. Even though most open-class terms are defined in Wikipedia as well, the authors of the article do not consider it necessary to present their references using wiki links. Hence, we need to extend wiki concepts by concatenating them with these open-class terms to form an extended vector. In addition, we ignore all close-class terms, since we cannot find efficient method to infer reliable links across them. As a result, texts are represented as a vector of wiki concepts.

Once we introduce wiki concepts to replace typical 'bag-of-words' approach, the dimensions of concept space will reach six order of magnitudes. We cannot ignore the data spareness issue and computation cost when the concept space is so huge. Actually, for a wiki article and a set of relevant articles, the involved concepts are limited, and we need to explore them in a small sub-space. For instance, 59 articles about *Kennedy family* in Wikipedia have 10,399 distinctive wiki concepts only, where 5,157 wiki concepts exist twice and more. Computing the overlapping among them is feasible.

Furthermore, we need to merge the wiki concepts with identical or close semantic (namely, building links between these synonyms and paraphrases). We measure the semantic similarity between two concepts by using cosine distance between their wiki articles, which are represented as the vectors of wiki concepts as well. For computation efficiency, we calculate semantic similarities between all promising concept pairs beforehand, and then retrieve the value in a Hash table directly. We spent CPU time of about 12.5 days preprocessing the se-

[2] All sample sentences in this paper come from this article if not specified.

[3] Consider new pseudo sentence: *'After graduating from Stanford in 1988, she went to work ... in Boston.'* We do not need assign link between *Stanford* and *Boston* as well.

mantic calculation. Details are available at our technical report (Lu et al., 2008).

Following the principle of TFIDF, we define the weighing metric for the vector represented by wiki concepts using the entire Wikipedia as the observation collection. We define the CFIDF weight of wiki concept i in article j as:

$$w_{i,j} = cf_{i,j} \cdot idf_i = \frac{n_{i,j}}{\sum_k n_{k,j}} \cdot \log \frac{|D|}{|d_j : t_i \in d_j|}, \quad (1)$$

where $cf_{i,j}$ is the frequency of concept i in article j; idf_i is the inverse frequency of concept i in Wikipedia; and D is the number of articles in Wikipedia. Here, sparse wiki concepts will have more contribution.

In brief, we represent articles in terms of wiki concepts using the steps below.

1. Extract the wiki concepts marked by wiki links in context.
2. Detect the remaining open-class terms as wiki concepts as well.
3. Merge concepts whose semantic similarity is larger than predefined threshold (0.35 in our experiments) into the one with largest idf.
4. Weight all concepts according to Eqn (1).

3.2 Document Concept Lattice Model

Next, we build the document concept lattice (DCL) for articles represented by wiki concepts. For illustration on how DCL is built, we consider 8 sentences from DUC 2005 Cluster *d324e* (Ye et al., 2007) as case study. 8 sentences, represented by 16 distinctive concepts A-P, are considered as the base nodes 1-8 as shown in Figure 1. Once we group nodes by means of the maximal common concepts among base nodes hierarchically, we can obtain the derived nodes 11-41, which form a DCL. A derived node will annotate a local topic through a set of shared concepts, and define a sub concept space that contains the covered base nodes under proper projection. The derived node, accompanied with its base nodes, is apt to interpret a particular argument (or statement) about the involved concepts. Furthermore, one base node among them, coupled with the corresponding sentence, is capable of this interpretation and could represent the other base nodes to some degree.

In order to Extract a set of sentences to cover key distinctive local topics (arguments) as much as possible, we need to select a set of *important* non-overlapping derived nodes. We measure the importance of node N in DCL of article j in term of ***representative power (RP)*** as:

$$RP(N) = \sum_{c_i \in N} (|c_i| \cdot w_{i,j}) / \log(|N|), \quad (2)$$

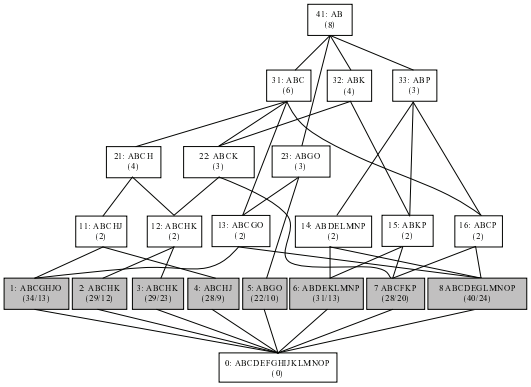


Figure 1: A sample of concept lattice

where concept c_i in node N is weighted by $w_{i,j}$ according to Eqn (1), and $|N|$ denotes the concept number in N (if N is a base node) or the number of distinct concepts in $|N|$ (if N is a derived node), respectively. Here, $|c_i|$ represents the c's frequency in N, and $\log(|N|)$ reflects N's cost if N is selected (namely, how many concepts are used in N). For example, 7 concepts in sentence 1 lead to the total $|c|$ of 34 if their weights are set to 1 equally. Its RP is $RP(1) = 34/log(7) = 40.23$. Similarly, $RP(31) = 6 * 3/log(3) = 37.73$.

By selecting a set of non-overlapping derived nodes with maximal RP, we are able to obtain a set of local topics with highest representativeness and diversity. Next, a representative sentence with maximal RP in each of such derived nodes is chosen to represent the local topics in observation. When the length of the required summary changes, the number of the local topics needed will also be modified. Consequently, we are able to select the sets of appropriate derived nodes in diverse generalization levels, and obtain various versions of summaries containing the local topics with appropriate granularities.

In the DCL example shown in Figure 1, if we expect to have a summary with two sentences, we will select the derived nodes 31 and 32 with highest RP. Nodes 31 and 32 will infer sentences 4 and 2, and they will be concatenated to form a summary. If the summary is increased to three sentences, then three derived nodes 31, 23 and 33 with maximal RP will render representative sentences 4, 5 and 6. Hence, the different number of actual sentences (4+5+6 vs. 4+2) will be selected depending on the length of the required summary. The uniqueness of DCL is that the sentences used in a shorter summary may not appear in a longer summary for the same source text. According to the distinctive derived nodes in diverse levels, the sentences with different generalization abilities are chosen to generate various summaries.

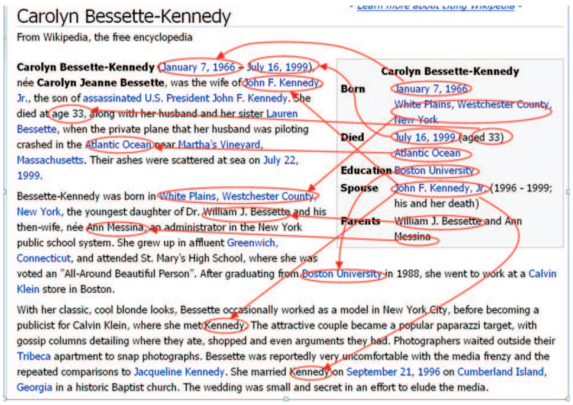


Figure 2: Properties in infobox and their support sentences

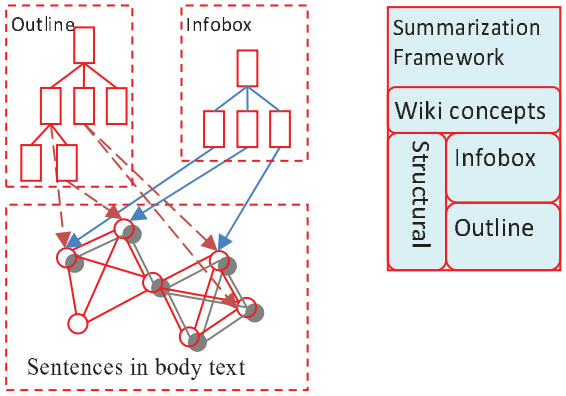


Figure 3: Extend document concept lattice by outline and infobox in Wikipedia

3.3 Model of Extended Document Concept Lattice (EDCL)

Different from free text and general web documents, wiki articles contain structural features, such as infoboxes and outlines, which correlate strongly to nuggets in definition TREC-QA. By integrating these structural features, we will generate better RP measures in derived topics which facilitates better priority assignment in local topics.

3.3.1 Outline: Wiki Macro Structure

A long wiki article usually has a hierarchical **outline** using inner links to organize its contents. For example, wiki article *Cat* consists of a set of hierarchical sections under the outline of *mouth, legs, Metabolism, genetics*, etc. This outline provides a hierarchical clustering of sub-topics assigned by its author(s), which implies that selecting sentences from diverse sections of outline is apt to obtain a balanced summary. Actually, DCL could be considered as the composite of many kinds of clusterings (Ye et al., 2007). Importing the clustering from outline into DCL will be helpful for the generation of a balanced summary. We thus incorporate the structure of outline into DCL as follows: (i) treat section titles as concepts in the pseudo derived nodes; (ii) link these pseudo nodes and the base nodes in this section if they share concepts; and (iii) revise base nodes' RP in Eqn (2) (see Section 3.3.3).

3.3.2 Infobox: a Mini Version of Summary

Infobox tabulates the key properties about the topic concept of a wiki article. It could be considered as a mini summary, where many nuggets in TREC-QA are included. As properties in infobox are not complete sentences and do not present relevant arguments, it is inappropriate to concatenate them as a summary. However, they are good indicators for summary generation. Following the terms in a property (e.g., spouse name and graduation school), we can find the corresponding sentences in the body of the text that contains such terms[4]. It describes the details about the involved property and provides the relevant arguments. We call it ***support sentence***.

Now, again, we have a hierarchy: Infobox + properties + support sentences. This hierarchy can be used to render a summary by concatenating the support sentences. This summary is inferred from hand-crafted infobox directly and is a full version of infobox; so its quality is guaranteed. However, it is possibly inapplicable due to its improper length. Following the iterative reinforcement approach for summarization and keyword extraction (Wan et al., 2007), it could be used to refine other versions of summaries. Hence, we utilize infobox and its support sentences to modify nodes' RPs in DCL so that the priority of local topics has bias to infobox. To achieve it, we extend DCL by inserting a hierarchy from infobox: (i) generate a pseudo derived node for each property; (ii) link every derived node to its support sentences; and (iii) cover these pseudo nodes by a virtual derived node called infobox.

3.3.3 Summary Generation from EDCL

In DCL, sentences with common concepts form local topics by autonomous approach, where shared concepts are depicted in derived nodes. Now we introduce two additional hierarchies derived from outline and infobox into DCL to refine RPs of salient local topics for summarization, which will render a model named **extended document concept lattice** (EDCL). As shown in Figure 3, base nodes in EDCL covered by pseudo derived nodes will increase their RPs when they receive influence from outline and infobox. Also, if RPs of their covered base nodes changes, the original derived nodes will modify their RPs as well. Therefore, the new

[4]Sometimes, we can find more than one appropriate sentence for a property. In our investigation, we select top two sentences with the occurrence of the particular term if available.

RPs in derived nodes and based nodes will lead to better priority of ranking derived nodes, which is likely to result in a better summary. One important direct consequence of introducing the extra hierarchies is to increase the RP of nodes relevant to outline and infobox so that the summaries from EDCL are likely to follow human-crafted ones.

The influence of human effects are transmitted in a 'V' curve approach. We utilize the following steps to generate a summary with a given length (say m sentences) from EDCL.

1. Build a normal DCL, and compute RP for each node according to Eqn 2.
2. Generate pseudo derived nodes (denoted by P) based on outline and infobox, and link the pseudo derived nodes to their relevant base nodes (denoted by B_0).
3. Update RP in B_0 by magnifying the contribution of shared concepts between P and N_0[5].
4. Update RP in derived nodes that cover B_0 on account of the new RP in B_0.
5. Select m non-overlapping derived nodes with maximal RP as the current observation.
6. Concatenate representative sentences with top RP from each derived node in the current observation as output.
7. If one representative sentence is covered by more than one derived node in step 5, the output will be less than m sentences. In this case, we need to increase m and repeat step 5-6 until m sentences are selected.

4 Experiments

The purposes of our experiment are two-fold: (i) evaluate the effects of wiki definition to the TREC-QA task; and (ii) examine the characteristics and summarization performance of EDCL.

4.1 Corpus Construction

We adopt the tasks of TREC-QA in 2004-2006 (TREC 12-14) as test scope. We retrieve articles with identical topic names from Wikipedia[6]. Non-letter transformations are permitted (e.g., from *'Carolyn Bessette-Kennedy'* to *'Carolyn_Bessette-Kennedy'*). Because our focus is summarization evaluation, we ignore the cases in TREC-QA where the exact topics do not exist in Wikipedia, even though relevant topics are available (e.g., *'France wins World Cup in soccer'* in TREC-QA vs. *'France_national_football_team'* and *'2006_FIFA_World_Cup'* in Wikipedia). Finally, among the 215 topics in TREC 12-14, we obtain 180 wiki articles with the same topics.

We ask 15 undergraduate and graduate students from the Department of English Literature in National University of Singapore to choose 7-14 sentences in the above wiki articles as extractive summaries. Each wiki article is annotated by 3 persons separately. In order for the volunteers to avoid the bias from TREC-QA corpus, we do not provide queries and nuggets used in TREC-QA. Similar to TREC nuggets, we call the selected sentences ***wiki nuggets***. Wiki nuggets provides the ground truth of the performance evaluation, since some TREC nuggets are possibly unavailable in Wikipedia.

Here, we did not ask the volunteers to create snippets (like TREC-QA) or compose an abstractive summary (like DUC). This is because of the special style of wiki articles: the entire document is a long summary without trivial stuff. Usually, we do not need to concatenate key phrases from diverse sentences to form a recapitulative sentence. Meanwhile, selecting a set of salient sentences to form a concise version is a relatively less time-consuming but applicable approach. Snippets, by and large, lead to bad readability, and therefore we do not employ this approach.

In addition, the volunteers also annotate 7-10 pairs of question/answer for each article for further research on QA using Wikipedia. The corpus, called ***TREC-Wiki collection***, is available at our site (*http://nuscu.ddns.comp.nus.edu.sg*). The system of Wikipedia summarization using EDCL is launched on the Web as well.

4.2 Corpus Exploration

4.2.1 Answer availability

The availability of answers in Wikipedia for TREC-QA could be measured in two aspects: (i) how many TREC-QA topics have been covered by Wikipedia? and (ii) how many nuggets could be found in the corresponding wiki article? We find that (i) over 80% of topics (180/215) in the TREC 12-14 are available in Wikipedia, and (ii) about 47% TREC nuggets could be detected directly from Wikipedia (examining applet modified from Pourpre (Lin and Demner-Fushman, 2006)). In contrast, 6,463 nuggets existing in TREC-QA 12-14 are distributed in 4,175 articles from AQUAINT corpus. We can say that Wikipedia is the answer goldmine for TREC-QA questions.

When we look into these TREC nuggets in wiki articles closely, we find that most of them are embedded in wiki links or relevant to infobox. It suggests that they are indicators for sentences having nuggets.

[5] We magnify it by adding $|c_0| * w_c * \eta$. Here, c_0 is the shared concepts between P and N_0, and η is the influence factor and set to 2-5 in our experiments.

[6] The dump is available at http://download.wikimedia.org/. Our dump was downloaded in Sept 2007.

4.2.2 Correlation between TREC nuggets and non-text features

Analyzing the features used could let us understand summarization better (Nenkova and Louis, 2008). Here, we focus on the statistical analysis between TREC/wiki nuggets and non-textual features such as wiki links, infobox and outline. The features used are introduced in Table 1. The correlation coefficients are listed in Table 2.

Observation: (1) On the whole, wiki nuggets exhibit higher correlation to non-textual features than TREC nuggets do. The possible reason is that TREC nuggets are extracted from AQUAINT rather than Wikipedia. (2) As compared to other features, infobox and wiki links strongly relate to nuggets. They are thus reliable features beyond text for summarization. (3) Sentence positions exhibit weak correlation to nuggets, even though the first sentence of an article is a good one-sentence definition.

Feature	Description
Link	Does the sentence have link?
Topic rel.	Does the sentence contain any word in topic concept?
Outline rel.	Does the sentence hold word in its section title(s) (outline)?
Infobox rel.	Is it a support sentence?
Position	First sentence of the article, first sentence and last sentence of a paragraph, or others?

Table 1: Features for correlation measurement

Feature	TREC nuggets	Wiki nuggets
Link	0.087	0.120
Topic rel.	0.038	0.058
Outline rel.	0.078	0.076
Infobox rel.	0.089	0.170
Position	-0.047	0.021

Table 2: Correlation Coefficients between non-textual features in Wiki and TREC/wiki nuggets

4.3 Statistical Characteristics of EDCL

We design four runs with various configurations as shown in Table 3. We implement a sentence re-ranking program using MMR (maximal marginal relevance) (Carbonell and Goldstein, 1998) in Run 1, which is considered as the test baseline. We apply standard DCL in Run 2, where concepts are determined according to their definitions in WordNet (Ye et al., 2007). We introduce wiki concepts for standard DCL in Run 3. Run 4 is the full version of EDCL, which considers both outline and infobox.

Observations: (1) In Run 1, the average number of distinctive words per article is near to 1200 after stop words are filtered out. When we merge diverse words having similar semantic according to WordNet concepts , we obtain 873 concepts per article on average in Run 2. The word number decreases by about 28% as a result of the omission of close-class terms and the merging of synonyms and paraphrases. (2) When wiki concepts are introduced in Run 3, the number of concepts continues to decrease. Here, some adjacent single-word terms are merged into wiki concepts if they are annotated by wiki links. Even though the reduction of total concepts is limited, these new wiki concepts will group the terms that cannot be detected by WordNet. (3) DCL based on WordNet concepts has less derived nodes (Run 3) than DCL based on wiki concepts does, although the former has more concepts. It implies that wiki concepts lead to higher link density in DCL as more links between concepts can be detected. (4) Outline and infobox will bring additional 54 derived nodes (from 1695 to 1741). Additional computation cost is limited when they are introduced into EDCL.

Run 1	Word co-occurrence + MMR
Run 2	Basic DCL model (WordNet concepts)
Run 3	DCL + wiki concepts
Run 4	EDCL (DCL + wiki concepts + outline + infobox)

Table 3: Test configurations

	Concepts	Base nodes	Derived nodes
Run 1	1173 (number of words)		
Run 2	873	259	1517
Run 3	826	259	1695
Run 4	831	259	1741

Table 4: Average node/concept numbers in DCL and EDCL

4.4 Summarization Performance of EDCL

We evaluate the performance of EDCL from two aspects such as contribution to TREC-QA definition task and accuracy of summarization in our TREC-Wiki collection.

Since factoid/list questions are about the most essential information of the target as well, like Cui's approach (2005), we treat factoid/list answers as essential nuggets and add them to the gold standard list of definition nuggets. We set the sentence number of summaries generated by the system to

12. We examine the definition quality by nugget recall (NR) and an approximation to nugget precision (NP) on answer length. These scores are combined using the F_1 and F_3 measures. The recall in F_3 is weighted three times as important as precision. The evaluation is automatically conducted by Pourpre v1.1 (Lin and Demner-Fushman, 2006).

Based on the performance of EDCL for TREC-QA definition task listed in Table 5, we observe that: (i) When EDCL considers wiki concepts and structural features such as outline and infobox, its F-scores increase significantly (Run 3 and Run 4). (ii) Table 5 also lists the results of Cui's system (marked by asterisk) using bigram soft patterns (Cui et al., 2005), which is trained by TREC-12 and tested on TREC 13. Our EDCL can achieve comparable or better F-scores on the 180 topics in TREC 12-14. It suggests that Wikipedia could provide high-quality definition directly even though we do not use AQUAINT. (iii) The precision of EDCL in Run 4 outperforms that of soft-pattern approach remarkably (from 0.34 to 0.497). One possible reason is that all sentences in a wiki article are oriented to its topic, and the sentence irrelevant to its topic hardly occurs.

	NR	**NP**	F_1	F_3
Run 1	0.247	0.304	0.273	0.252
Run 2	0.262	0.325	0.290	0.267
Run 3	0.443	0.431	0.431	0.442
Run 4	0.538	0.497	0.517	0.534
Bigram SP*	0.552	0.340	0.421	0.510

Table 5: EDCL evaluated by TREC-QA nuggets

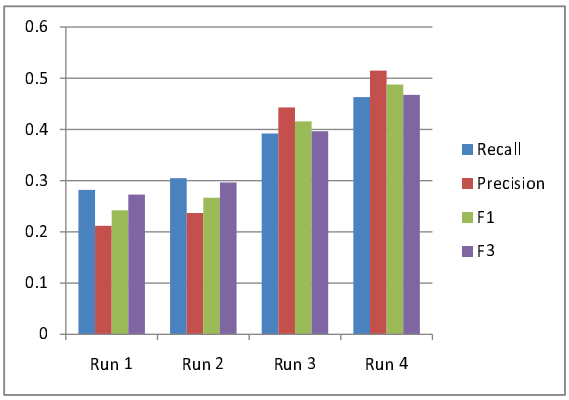


Figure 4: Performance of summarizing Wikipedia using EDCL with different configurations

We also test the performance of EDCL using extractive summaries in TREC-Wiki collection. By means of comparing to each set of sentences selected by a volunteer, we examine how many exact annotated sentences are selected by the system using different configurations. The average recalls and precisions as well as their F-scores are shown in Figure 4.

Observations: (i) The structural information of Wikipeida has significant contribution to EDCL for summarization. We manually examine some summaries and find that the sentences containing more wiki links are apt to be chosen when wiki concepts are introduced in EDCL. Most sentences in output summaries in Run 4 usually have 1-3 links and relevant to infobox or outline. (ii) When using wiki concepts, infobox and outline to enrich DCL, we find that the precision of sentence selection has improved more than the recall. It reaffirms the conclusion in the previous TREC-QA test in this subsection. (iii) In addition, we manually examine the summaries on some wiki articles with common topics, such as *car, house, money*, etc. We find that the summaries generated by EDCL could effectively grasp the key information about the topics when the sentence number of summaries exceeds 10.

5 Conclusion and Future Work

Wikipedia recursively defines enormous concepts in huge vector space of wiki concepts. The explicit semantic representation via wiki concepts allows us to obtain more reliable links between passages. Wikipedia's special structural features, such as wiki links, infobox and outline, reflect the hidden human *knowledge*. The first sentence of a wiki article, infobox (and its support sentences), outline (and its relevant sentences), as well as the entire document could be considered as diverse summaries with various lengths. In our proposed model, local topics are autonomously organized in a lattice structure according to their overlapping relations. The hierarchies derived from infobox and outline are imported to refine the representative powers of local topics by emphasizing the concepts relevant to infobox and outline. Experiments indicate that our proposed model exhibits promising performance in summarization and QA definition tasks.

Of course, there are rooms to further improve the model. Possible improvements includes: (a) using advanced semantic and parsing technologies to detect the support and relevant sentences for infobox and outline; (b) summarizing multiple articles in a wiki category; and (c) exploring the mapping from close-class terms to open-class terms for more links between passages is likely to forward some interesting results.

More generally, the *knowledge* hidden in non-textual features of Wikipedia allow the model to harvest better definition summaries. It is challenging but possibly fruitful to recast the normal documents with wiki styles so as to adopt EDCL for free text and enrich the research efforts on other NLP tasks.

References

[Ahn et al.2004] David Ahn, Valentin Jijkoun, et al. 2004. Using Wikipedia at the TREC QA Track. In *Text REtrieval Conference*.

[Carbonell and Goldstein1998] J. Carbonell and J. Goldstein. 1998. The use of mmr, diversity-based re-ranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336.

[Cui et al.2005] Hang Cui, Min-Yen Kan, and Tat-Seng Chua. 2005. Generic soft pattern models for definitional question answering. In *Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval*, pages 384–391, New York, NY, USA. ACM.

[Erkan and Radev2004] Güneş Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Artificial Intelligence Research*, 22:457–479.

[Furnas et al.1987] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. 1987. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971.

[Gabrilovich and Markovitch2007] Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of The Twentieth International Joint Conference for Artificial Intelligence*, pages 1606–1611, Hyderabad, India.

[Kor and Chua2007] Kian-Wei Kor and Tat-Seng Chua. 2007. Interesting nuggets and their impact on definitional question answering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–342, New York, NY, USA. ACM.

[Lin and Demner-Fushman2006] Jimmy J. Lin and Dina Demner-Fushman. 2006. Methods for automatically evaluating answers to complex questions. *Information Retrieval*, 9(5):565–587.

[Lin and Hovy2000] Chin-Yew Lin and Eduard Hovy. 2000. The automated acquisition of topic signatures for text summarization. In *Proceedings of the 18th conference on Computational linguistics*, pages 495–501, Morristown, NJ, USA. ACL.

[Lita et al.2004] Lucian Vlad Lita, Warren A. Hunt, and Eric Nyberg. 2004. Resource analysis for question answering. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 18, Morristown, NJ, USA. ACL.

[Liu et al.2004] Shuang Liu, Fang Liu, Clement Yu, and Weiyi Meng. 2004. An effective approach to document retrieval via utilizing wordnet and recognizing phrases. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 266–272, New York, NY, USA. ACM.

[Lu et al.2008] Jie Lu, Shiren Ye, and Tat-Seng Chua. 2008. Explore semantic similarity and semantic relatedness via wikipedia. Technical report, National Univeristy of Singapore, http://nuscu.ddns.comp.nus.edu.sg.

[Mei and Zhai2008] Qiaozhu Mei and ChengXiang Zhai. 2008. Generating impact-based summaries for scientific literature. In *Proceedings of ACL-08: HLT*, pages 816–824, Columbus, Ohio, June. ACL.

[Nenkova and Louis2008] Ani Nenkova and Annie Louis. 2008. Can you summarize this? identifying correlates of input difficulty for multi-document summarization. In *Proceedings of ACL-08: HLT*, pages 825–833, Columbus, Ohio, June. ACL.

[Nenkova et al.2007] Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Transactions on Speech and Language Processing*, 4(2):4.

[Silber and McCoy2002] H. Grogory Silber and Kathleen F. McCoy. 2002. Efficiently computed lexical chains as an intermediate representation for automatic text summarization. *Computational Linguistics*, 28(4):487–496.

[Teufel and Moens2002] Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: experiments with relevance and rhetorical status. *Computational Linguistics*, 28(4):409–445, December.

[Voorhees and Dang2005] Ellen M. Voorhees and Hoa Trang Dang. 2005. Overview of the trec 2005 question answering track. In *Text REtrieval Conference*.

[Voorhees2004] Ellen M. Voorhees. 2004. Overview of the trec 2004 question answering track. In *Text REtrieval Conference*.

[Wan et al.2007] Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. 2007. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 552–559, Prague, Czech Republic, June. ACL.

[Ye et al.2007] Shiren Ye, Tat-Seng Chua, Min-Yen Kan, and Long Qiu. 2007. Document concept lattice for text understanding and summarization. *Information Processing and Management*, 43(6):1643–1662.

Automatically Generating Wikipedia Articles: A Structure-Aware Approach

Christina Sauper and Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{csauper,regina}@csail.mit.edu

Abstract

In this paper, we investigate an approach for creating a comprehensive textual overview of a subject composed of information drawn from the Internet. We use the high-level structure of human-authored texts to automatically induce a domain-specific template for the topic structure of a new overview. The algorithmic innovation of our work is a method to learn topic-specific extractors for content selection jointly for the entire template. We augment the standard perceptron algorithm with a global integer linear programming formulation to optimize both local fit of information into each topic and global coherence across the entire overview. The results of our evaluation confirm the benefits of incorporating structural information into the content selection process.

1 Introduction

In this paper, we consider the task of automatically creating a multi-paragraph overview article that provides a comprehensive summary of a subject of interest. Examples of such overviews include actor biographies from IMDB and disease synopses from Wikipedia. Producing these texts by hand is a labor-intensive task, especially when relevant information is scattered throughout a wide range of Internet sources. Our goal is to automate this process. We aim to create an overview of a subject – e.g., *3-M Syndrome* – by intelligently combining relevant excerpts from across the Internet.

As a starting point, we can employ methods developed for multi-document summarization. However, our task poses additional technical challenges with respect to content planning. Generating a well-rounded overview article requires proactive strategies to gather relevant material, such as searching the Internet. Moreover, the challenge of maintaining output readability is magnified when creating a longer document that discusses multiple topics.

In our approach, we explore how the high-level structure of human-authored documents can be used to produce well-formed comprehensive overview articles. We select relevant material for an article using a domain-specific automatically generated content template. For example, a template for articles about diseases might contain *diagnosis*, *causes*, *symptoms*, and *treatment*. Our system induces these templates by analyzing patterns in the structure of human-authored documents in the domain of interest. Then, it produces a new article by selecting content from the Internet for each part of this template. An example of our system's output[1] is shown in Figure 1.

The algorithmic innovation of our work is a method for learning topic-specific extractors for content selection jointly across the entire template. Learning a single topic-specific extractor can be easily achieved in a standard classification framework. However, the choices for different topics in a template are mutually dependent; for example, in a multi-topic article, there is potential for redundancy across topics. Simultaneously learning content selection for all topics enables us to explicitly model these inter-topic connections.

We formulate this task as a *structured classification* problem. We estimate the parameters of our model using the perceptron algorithm augmented with an integer linear programming (ILP) formulation, run over a training set of example articles in the given domain.

The key features of this structure-aware approach are twofold:

[1] This system output was added to Wikipedia at http://en.wikipedia.org/wiki/3-M_syndrome on June 26, 2008. The page's history provides examples of changes performed by human editors to articles created by our system.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 208–216,
Suntec, Singapore, 2-7 August 2009.

Diagnosis ...No laboratories offering molecular genetic testing for prenatal diagnosis of 3-M syndrome are listed in the GeneTests Laboratory Directory. However, prenatal testing may be available for families in which the disease-causing mutations have been identified in an affected family member in a research or clinical laboratory.

Causes Three M syndrome is thought to be inherited as an autosomal recessive genetic trait. Human traits, including the classic genetic diseases, are the product of the interaction of two genes, one received from the father and one from the mother. In recessive disorders, the condition does not occur unless an individual inherits the same defective gene for the same trait from each parent. ...

Symptoms ...Many of the symptoms and physical features associated with the disorder are apparent at birth (congenital). In some cases, individuals who carry a single copy of the disease gene (heterozygotes) may exhibit mild symptoms associated with Three M syndrome.

Treatment ...Genetic counseling will be of benefit for affected individuals and their families. Family members of affected individuals should also receive regular clinical evaluations to detect any symptoms and physical characteristics that may be potentially associated with Three M syndrome or heterozygosity for the disorder. Other treatment for Three M syndrome is symptomatic and supportive.

Figure 1: A fragment from the automatically created article for 3-M Syndrome.

- **Automatic template creation:** Templates are automatically induced from human-authored documents. This ensures that the overview article will have the breadth expected in a comprehensive summary, with content drawn from a wide variety of Internet sources.
- **Joint parameter estimation for content selection:** Parameters are learned jointly for all topics in the template. This procedure optimizes both local relevance of information for each topic and global coherence across the entire article.

We evaluate our approach by creating articles in two domains: Actors and Diseases. For a data set, we use Wikipedia, which contains articles similar to those we wish to produce in terms of length and breadth. An advantage of this data set is that Wikipedia articles explicitly delineate topical sections, facilitating structural analysis. The results of our evaluation confirm the benefits of structure-aware content selection over approaches that do not explicitly model topical structure.

2 Related Work

Concept-to-text generation and text-to-text generation take very different approaches to content selection. In traditional concept-to-text generation, a content planner provides a detailed template for what information should be included in the output and how this information should be organized (Reiter and Dale, 2000). In text-to-text generation, such templates for information organization are not available; sentences are selected based on their salience properties (Mani and Maybury, 1999). While this strategy is robust and portable across domains, output summaries often suffer from coherence and coverage problems.

In between these two approaches is work on domain-specific text-to-text generation. Instances of these tasks are biography generation in summarization and answering definition requests in question-answering. In contrast to a generic summarizer, these applications aim to characterize the types of information that are essential in a given domain. This characterization varies greatly in granularity. For instance, some approaches coarsely discriminate between biographical and non-biographical information (Zhou et al., 2004; Biadsy et al., 2008), while others go beyond binary distinction by identifying atomic events – e.g., occupation and marital status – that are typically included in a biography (Weischedel et al., 2004; Filatova and Prager, 2005; Filatova et al., 2006). Commonly, such templates are specified manually and are hard-coded for a particular domain (Fujii and Ishikawa, 2004; Weischedel et al., 2004).

Our work is related to these approaches; however, content selection in our work is driven by domain-specific automatically induced templates. As our experiments demonstrate, patterns observed in domain-specific training data provide sufficient constraints for topic organization, which is crucial for a comprehensive text.

Our work also relates to a large body of recent work that uses Wikipedia material. Instances of this work include information extraction, ontology induction and resource acquisition (Wu and Weld, 2007; Biadsy et al., 2008; Nastase, 2008; Nastase and Strube, 2008). Our focus is on a different task — generation of new overview articles that follow the structure of Wikipedia articles.

3 Method

The goal of our system is to produce a comprehensive overview article given a title – e.g., *Cancer*. We assume that relevant information on the subject is available on the Internet but scattered among several pages interspersed with noise.

We are provided with a training corpus consisting of n documents $d_1 \ldots d_n$ in the same domain – e.g., *Diseases*. Each document d_i has a title and a set of delineated sections[2] $s_{i1} \ldots s_{im}$. The number of sections m varies between documents. Each section s_{ij} also has a corresponding heading h_{ij} – e.g., *Treatment*.

Our overview article creation process consists of three parts. First, a preprocessing step creates a template and searches for a number of candidate excerpts from the Internet. Next, parameters must be trained for the content selection algorithm using our training data set. Finally, a complete article may be created by combining a selection of candidate excerpts.

1. **Preprocessing** (Section 3.1) Our preprocessing step leverages previous work in topic segmentation and query reformulation to prepare a template and a set of candidate excerpts for content selection. Template generation must occur once per domain, whereas search occurs every time an article is generated in both learning and application.
   (a) **Template Induction** To create a content template, we cluster all section headings $h_{i1} \ldots h_{im}$ for all documents d_i. Each cluster is labeled with the most common heading h_{ij} within the cluster. The largest k clusters are selected to become topics $t_1 \ldots t_k$, which form the domain-specific content template.
   (b) **Search** For each document that we wish to create, we retrieve from the Internet a set of r excerpts $e_{j1} \ldots e_{jr}$ for each topic t_j from the template. We define appropriate search queries using the requested document title and topics t_j.
2. **Learning Content Selection** (Section 3.2) For each topic t_j, we learn the corresponding topic-specific parameters $\mathbf{w}_j$ to determine the quality of a given excerpt. Using the perceptron framework augmented with an ILP formulation for global optimization, the system is trained to select the best excerpt for each document d_i and each topic t_j. For training, we assume the best excerpt is the original human-authored text s_{ij}.
3. **Application** (Section 3.2) Given the title of a requested document, we select several excerpts from the candidate vectors returned by the search procedure (1b) to create a comprehensive overview article. We perform the decoding procedure jointly using learned parameters $\mathbf{w}_1 \ldots \mathbf{w}_k$ and the same ILP formulation for global optimization as in training. The result is a new document with k excerpts, one for each topic.

[2] In data sets where such mark-up is not available, one can employ topical segmentation algorithms as an additional preprocessing step.

3.1 Preprocessing

Template Induction A content template specifies the topical structure of documents in one domain. For instance, the template for articles about actors consists of four topics $t_1 \ldots t_4$: *biography*, *early life*, *career*, and *personal life*. Using this template to create the biography of a new actor will ensure that its information coverage is consistent with existing human-authored documents.

We aim to derive these templates by discovering common patterns in the organization of documents in a domain of interest. There has been a sizable amount of research on structure induction ranging from linear segmentation (Hearst, 1994) to content modeling (Barzilay and Lee, 2004). At the core of these methods is the assumption that fragments of text conveying similar information have similar word distribution patterns. Therefore, often a simple segment clustering across domain texts can identify strong patterns in content structure (Barzilay and Elhadad, 2003). Clusters containing fragments from many documents are indicative of topics that are essential for a comprehensive summary. Given the simplicity and robustness of this approach, we utilize it for template induction.

We cluster all section headings $h_{i1} \ldots h_{im}$ from all documents d_i using a repeated bisectioning algorithm (Zhao et al., 2005). As a similarity function, we use cosine similarity weighted with TF*IDF. We eliminate any clusters with low internal similarity (i.e., smaller than 0.5), as we assume these are "miscellaneous" clusters that will not yield unified topics.

We determine the average number of sections k over all documents in our training set, then select the k largest section clusters as topics. We order these topics as $t_1 \ldots t_k$ using a majority ordering algorithm (Cohen et al., 1998). This algorithm finds a total order among clusters that is consistent with a maximal number of pairwise relationships observed in our data set.

Each topic t_j is identified by the most frequent heading found within the cluster – e.g., *Causes*. This set of topics forms the content template for a domain.

Search To retrieve relevant excerpts, we must define appropriate search queries for each topic $t_1 \ldots t_k$. Query reformulation is an active area of research (Agichtein et al., 2001). We have experimented with several of these methods for drawing search queries from representative words in the body text of each topic; however, we find that the best performance is provided by deriving queries from a conjunction of the document title and topic – e.g., *"3-M syndrome" diagnosis*.

Using these queries, we search using Yahoo! and retrieve the first ten result pages for each topic. From each of these pages, we extract all possible excerpts consisting of chunks of text between standardized boundary indicators (such as <p> tags). In our experiments, there are an average of 6 excerpts taken from each page. For each topic t_j of each document we wish to create, the total number of excerpts r found on the Internet may differ. We label the excerpts $e_{j1} \ldots e_{jr}$.

3.2 Selection Model

Our selection model takes the content template $t_1 \ldots t_k$ and the candidate excerpts $e_{j1} \ldots e_{jr}$ for each topic t_j produced in the previous steps. It then selects a series of k excerpts, one from each topic, to create a coherent summary.

One possible approach is to perform individual selections from each set of excerpts $e_{j1} \ldots e_{jr}$ and then combine the results. This strategy is commonly used in multi-document summarization (Barzilay et al., 1999; Goldstein et al., 2000; Radev et al., 2000), where the combination step eliminates the redundancy across selected excerpts. However, separating the two steps may not be optimal for this task — the balance between coverage and redundancy is harder to achieve when a multi-paragraph summary is generated. In addition, a more discriminative selection strategy is needed when candidate excerpts are drawn directly from the web, as they may be contaminated with noise.

We propose a novel joint training algorithm that learns selection criteria for all the topics simultaneously. This approach enables us to maximize both local fit and global coherence. We implement this algorithm using the perceptron framework, as it can be easily modified for structured prediction while preserving convergence guarantees (Daumé III and Marcu, 2005; Snyder and Barzilay, 2007).

In this section, we first describe the structure and decoding procedure of our model. We then present an algorithm to jointly learn the parameters of all topic models.

3.2.1 Model Structure

The model inputs are as follows:

- The title of the desired document
- $t_1 \ldots t_k$ — topics from the content template
- $e_{j1} \ldots e_{jr}$ — candidate excerpts for each topic t_j

In addition, we define feature and parameter vectors:

- $\phi(e_{jl})$ — feature vector for the lth candidate excerpt for topic t_j
- $\mathbf{w}_1 \ldots \mathbf{w}_k$ — parameter vectors, one for each of the topics $t_1 \ldots t_k$

Our model constructs a new article by following these two steps:

Ranking First, we attempt to rank candidate excerpts based on how representative they are of each individual topic. For each topic t_j, we induce a ranking of the excerpts $e_{j1} \ldots e_{jr}$ by mapping each excerpt e_{jl} to a score:

$$score_j(e_{jl}) = \phi(e_{jl}) \cdot \mathbf{w}_j$$

Candidates for each topic are ranked from highest to lowest score. After this procedure, the position l of excerpt e_{jl} within the topic-specific candidate vector is the excerpt's rank.

Optimizing the Global Objective To avoid redundancy between topics, we formulate an optimization problem using excerpt rankings to create the final article. Given k topics, we would like to select one excerpt e_{jl} for each topic t_j, such that the rank is minimized; that is, $score_j(e_{jl})$ is high.

To select the optimal excerpts, we employ integer linear programming (ILP). This framework is

commonly used in generation and summarization applications where the selection process is driven by multiple constraints (Marciniak and Strube, 2005; Clarke and Lapata, 2007).

We represent excerpts included in the output using a set of indicator variables, x_{jl}. For each excerpt e_{jl}, the corresponding indicator variable $x_{jl} = 1$ if the excerpt is included in the final document, and $x_{jl} = 0$ otherwise.

Our objective is to minimize the ranks of the excerpts selected for the final document:

$$\min \sum_{j=1}^{k} \sum_{l=1}^{r} l \cdot x_{jl}$$

We augment this formulation with two types of constraints.

Exclusivity Constraints We want to ensure that exactly one indicator x_{jl} is nonzero for each topic t_j. These constraints are formulated as follows:

$$\sum_{l=1}^{r} x_{jl} = 1 \qquad \forall j \in \{1 \dots k\}$$

Redundancy Constraints We also want to prevent redundancy across topics. We define $sim(e_{jl}, e_{j'l'})$ as the cosine similarity between excerpts e_{jl} from topic t_j and $e_{j'l'}$ from topic $t_{j'}$. We introduce constraints that ensure no pair of excerpts has similarity above 0.5:

$$(x_{jl} + x_{j'l'}) \cdot sim(e_{jl}, e_{j'l'}) \leq 1$$

$$\forall j, j' = 1 \dots k \qquad \forall l, l' = 1 \dots r$$

If excerpts e_{jl} and $e_{j'l'}$ have cosine similarity $sim(e_{jl}, e_{j'l'}) > 0.5$, only one excerpt may be selected for the final document – i.e., either x_{jl} or $x_{j'l'}$ may be 1, but not both. Conversely, if $sim(e_{jl}, e_{j'l'}) \leq 0.5$, both excerpts may be selected.

Solving the ILP Solving an integer linear program is NP-hard (Cormen et al., 1992); however, in practice there exist several strategies for solving certain ILPs efficiently. In our study, we employed *lp_solve*,[3] an efficient mixed integer programming solver which implements the Branch-and-Bound algorithm. On a larger scale, there are several alternatives to approximate the ILP results, such as a dynamic programming approximation to the knapsack problem (McDonald, 2007).

[3] `http://lpsolve.sourceforge.net/5.5/`

Feature	Value
UNI_$word_i$	count of word occurrences
POS_$word_i$	first position of word in excerpt
BI_$word_i$_$word_{i+1}$	count of bigram occurrences
SENT	count of all sentences
EXCL	count of exclamations
QUES	count of questions
WORD	count of all words
NAME	count of title mentions
DATE	count of dates
PROP	count of proper nouns
PRON	count of pronouns
NUM	count of numbers
FIRST_$word_1$	1*
FIRST_$word_1$_$word_2$	1†
SIMS	count of similar excerpts‡

Table 1: Features employed in the ranking model.
* Defined as the first unigram in the excerpt.
† Defined as the first bigram in the excerpt.
‡ Defined as excerpts with cosine similarity > 0.5

Features As shown in Table 1, most of the features we select in our model have been employed in previous work on summarization (Mani and Maybury, 1999). All features except the SIMS feature are defined for individual excerpts in isolation. For each excerpt e_{jl}, the value of the SIMS feature is the count of excerpts $e_{jl'}$ in the same topic t_j for which $sim(e_{jl}, e_{jl'}) > 0.5$. This feature quantifies the degree of repetition within a topic, often indicative of an excerpt's accuracy and relevance.

3.2.2 Model Training

Generating Training Data For training, we are given n original documents $d_1 \dots d_n$, a content template consisting of topics $t_1 \dots t_k$, and a set of candidate excerpts $e_{ij1} \dots e_{ijr}$ for each document d_i and topic t_j. For each section of each document, we add the gold excerpt s_{ij} to the corresponding vector of candidate excerpts $e_{ij1} \dots e_{ijr}$. This excerpt represents the target for our training algorithm. Note that the algorithm does not require annotated ranking data; only knowledge of this "optimal" excerpt is required. However, if the excerpts provided in the training data have low quality, noise is introduced into the system.

Training Procedure Our algorithm is a modification of the perceptron ranking algorithm (Collins, 2002), which allows for joint learning across several ranking problems (Daumé III and Marcu, 2005; Snyder and Barzilay, 2007). Pseudocode for this algorithm is provided in Figure 2.

First, we define $Rank(e_{ij1} \dots e_{ijr}, \mathbf{w}_j)$, which

ranks all excerpts from the candidate excerpt vector $e_{ij1} \dots e_{ijr}$ for document d_i and topic t_j. Excerpts are ordered by $score_j(e_{jl})$ using the current parameter values. We also define $Optimize(e_{ij1} \dots e_{ijr})$, which finds the optimal selection of excerpts (one per topic) given ranked lists of excerpts $e_{ij1} \dots e_{ijr}$ for each document d_i and topic t_j. These functions follow the ranking and optimization procedures described in Section 3.2.1. The algorithm maintains k parameter vectors $\mathbf{w}_1 \dots \mathbf{w}_k$, one associated with each topic t_j desired in the final article. During initialization, all parameter vectors are set to zeros (line 2).

To learn the optimal parameters, this algorithm iterates over the training set until the parameters converge or a maximum number of iterations is reached (line 3). For each document in the training set (line 4), the following steps occur: First, candidate excerpts for each topic are ranked (lines 5-6). Next, decoding through ILP optimization is performed over all ranked lists of candidate excerpts, selecting one excerpt for each topic (line 7). Finally, the parameters are updated in a joint fashion. For each topic (line 8), if the selected excerpt is not similar enough to the gold excerpt (line 9), the parameters for that topic are updated using a standard perceptron update rule (line 10). When convergence is reached or the maximum iteration count is exceeded, the learned parameter values are returned (line 12).

The use of ILP during each step of training sets this algorithm apart from previous work. In prior research, ILP was used as a postprocessing step to remove redundancy and make other global decisions about parameters (McDonald, 2007; Marciniak and Strube, 2005; Clarke and Lapata, 2007). However, in our training, we intertwine the complete decoding procedure with the parameter updates. Our joint learning approach finds per-topic parameter values that are maximally suited for the global decoding procedure for content selection.

Input:
$d_1 \dots d_n$: A set of n documents, each containing k sections $s_{i1} \dots s_{ik}$
$e_{ij1} \dots e_{ijr}$: Sets of candidate excerpts for each topic t_j and document d_i
Define:
$Rank(e_{ij1} \dots e_{ijr}, \mathbf{w}_j)$:
As described in Section 3.2.1:
Calculates $score_j(e_{ijl})$ for all excerpts for document d_i and topic t_j, using parameters $\mathbf{w}_j$.
Orders the list of excerpts by $score_j(e_{ijl})$ from highest to lowest.
$Optimize(e_{i11} \dots e_{ikr})$:
As described in Section 3.2.1:
Finds the optimal selection of excerpts to form a final article, given ranked lists of excerpts for each topic $t_1 \dots t_k$.
Returns a list of k excerpts, one for each topic.
$\phi(e_{ijl})$:
Returns the feature vector representing excerpt e_{ijl}
Initialization:
1 **For** $j = 1 \dots k$
2 Set parameters $\mathbf{w}_j = 0$
Training:
3 Repeat until convergence or while $iter < iter_{max}$:
4 **For** $i = 1 \dots n$
5 **For** $j = 1 \dots k$
6 $Rank(e_{ij1} \dots e_{ijr}, \mathbf{w}_j)$
7 $x_1 \dots x_k = Optimize(e_{i11} \dots e_{ikr})$
8 **For** $j = 1 \dots k$
9 **If** $sim(x_j, s_{ij}) < 0.8$
10 $\mathbf{w}_j = \mathbf{w}_j + \phi(s_{ij}) - \phi(x_i)$
11 $iter = iter + 1$
12 Return parameters $\mathbf{w}_1 \dots \mathbf{w}_k$

Figure 2: An algorithm for learning several ranking problems with a joint decoding mechanism.

4 Experimental Setup

We evaluate our method by observing the quality of automatically created articles in different domains. We compute the similarity of a large number of articles produced by our system and several baselines to the original human-authored articles using ROUGE, a standard metric for summary quality. In addition, we perform an analysis of editor reaction to system-produced articles submitted to Wikipedia.

Data For evaluation, we consider two domains: American Film Actors and Diseases. These domains have been commonly used in prior work on summarization (Weischedel et al., 2004; Zhou et al., 2004; Filatova and Prager, 2005; Demner-Fushman and Lin, 2007; Biadsy et al., 2008). Our text corpus consists of articles drawn from the corresponding categories in Wikipedia. There are 2,150 articles in American Film Actors and 523 articles in Diseases. For each domain, we randomly select 90% of articles for training and test on the remaining 10%. Human-authored articles in both domains contain an average of four topics, and each topic contains an average of 193 words. In order to model the real-world scenario where Wikipedia articles are not always available (as for new or specialized topics), we specifically exclude Wikipedia sources during our search pro-

	Avg. Excerpts	Avg. Sources
Amer. Film Actors		
Search	2.3	1
No Template	4	4.0
Disjoint	4	2.1
Full Model	**4**	**3.4**
Oracle	4.3	4.3
Diseases		
Search	3.1	1
No Template	4	2.5
Disjoint	4	3.0
Full Model	**4**	**3.2**
Oracle	5.8	3.9

Table 2: Average number of excerpts selected and sources used in article creation for test articles.

cedure (Section 3.1) for evaluation.

Baselines Our first baseline, *Search*, relies solely on search engine ranking for content selection. Using the article title as a query – e.g., *Bacillary Angiomatosis*, this method selects the web page that is ranked first by the search engine. From this page we select the first k paragraphs where k is defined in the same way as in our full model. If there are less than k paragraphs on the page, all paragraphs are selected, but no other sources are used. This yields a document of comparable size with the output of our system. Despite its simplicity, this baseline is not naive: extracting material from a single document guarantees that the output is coherent, and a page highly ranked by a search engine may readily contain a comprehensive overview of the subject.

Our second baseline, *No Template*, does not use a template to specify desired topics; therefore, there are no constraints on content selection. Instead, we follow a simplified form of previous work on biography creation, where a classifier is trained to distinguish biographical text (Zhou et al., 2004; Biadsy et al., 2008).

In this case, we train a classifier to distinguish domain-specific text. Positive training data is drawn from all topics in the given domain corpus. To find negative training data, we perform the search procedure as in our full model (see Section 3.1) using only the article titles as search queries. Any excerpts which have very low similarity to the original articles are used as negative examples. During the decoding procedure, we use the same search procedure. We then classify each excerpt as relevant or irrelevant and select the k non-redundant excerpts with the highest relevance confidence scores.

Our third baseline, *Disjoint*, uses the ranking perceptron framework as in our full system; however, rather than perform an optimization step during training and decoding, we simply select the highest-ranked excerpt for each topic. This equates to standard linear classification for each section individually.

In addition to these baselines, we compare against an *Oracle* system. For each topic present in the human-authored article, the *Oracle* selects the excerpt from our full model's candidate excerpts with the highest cosine similarity to the human-authored text. This excerpt is the optimal automatic selection from the results available, and therefore represents an upper bound on our excerpt selection task. Some articles contain additional topics beyond those in the template; in these cases, the *Oracle* system produces a longer article than our algorithm.

Table 2 shows the average number of excerpts selected and sources used in articles created by our full model and each baseline.

Automatic Evaluation To assess the quality of the resulting overview articles, we compare them with the original human-authored articles. We use ROUGE, an evaluation metric employed at the Document Understanding Conferences (DUC), which assumes that proximity to human-authored text is an indicator of summary quality. We use the publicly available ROUGE toolkit (Lin, 2004) to compute recall, precision, and F-score for ROUGE-1. We use the Wilcoxon Signed Rank Test to determine statistical significance.

Analysis of Human Edits In addition to our automatic evaluation, we perform a study of reactions to system-produced articles by the general public. To achieve this goal, we insert automatically created articles[4] into Wikipedia itself and examine the feedback of Wikipedia editors. Selection of specific articles is constrained by the need to find topics which are currently of "stub" status that have enough information available on the Internet to construct a valid article. After a period of time, we analyzed the edits made to the articles to determine the overall editor reaction. We report results on 15 articles in the Diseases category[5].

[4] In addition to the summary itself, we also include proper citations to the sources from which the material is extracted.

[5] We are continually submitting new articles; however, we report results on those that have at least a 6 month history at time of writing.

	Recall	Precision	F-score
Amer. Film Actors			
Search	0.09	0.37	0.13 *
No Template	0.33	0.50	0.39 *
Disjoint	0.45	0.32	0.36 *
Full Model	**0.46**	**0.40**	**0.41**
Oracle	0.48	0.64	0.54 *
Diseases			
Search	0.31	0.37	0.32 †
No Template	0.32	0.27	0.28 *
Disjoint	0.33	0.40	0.35 *
Full Model	**0.36**	**0.39**	**0.37**
Oracle	0.59	0.37	0.44 *

Table 3: Results of ROUGE-1 evaluation.
* Significant with respect to our full model for $p \leq 0.05$.
† Significant with respect to our full model for $p \leq 0.10$.

Type	Count
Total articles	15
Promoted articles	10
Edit types	
Intra-wiki links	36
Formatting	25
Grammar	20
Minor topic edits	2
Major topic changes	1
Total edits	85

Table 4: Distribution of edits on Wikipedia.

Since Wikipedia is a live resource, we do not repeat this procedure for our baseline systems. Adding articles from systems which have previously demonstrated poor quality would be improper, especially in Diseases. Therefore, we present this analysis as an additional observation rather than a rigorous technical study.

5 Results

Automatic Evaluation The results of this evaluation are shown in Table 3. Our full model outperforms all of the baselines. By surpassing the *Disjoint* baseline, we demonstrate the benefits of joint classification. Furthermore, the high performance of both our full model and the *Disjoint* baseline relative to the other baselines shows the importance of structure-aware content selection. The *Oracle* system, which represents an upper bound on our system's capabilities, performs well.

The remaining baselines have different flaws: Articles produced by the *No Template* baseline tend to focus on a single topic extensively at the expense of breadth, because there are no constraints to ensure diverse topic selection. On the other hand, performance of the *Search* baseline varies dramatically. This is expected; this baseline relies heavily on both the search engine and individual web pages. The search engine must correctly rank relevant pages, and the web pages must provide the important material first.

Analysis of Human Edits The results of our observation of editing patterns are shown in Table 4. These articles have resided on Wikipedia for a period of time ranging from 5-11 months. All of them have been edited, and no articles were removed due to lack of quality. Moreover, ten automatically created articles have been promoted by human editors from stubs to regular Wikipedia entries based on the quality and coverage of the material. Information was removed in three cases for being irrelevant, one entire section and two smaller pieces. The most common changes were small edits to formatting and introduction of links to other Wikipedia articles in the body text.

6 Conclusion

In this paper, we investigated an approach for creating a multi-paragraph overview article by selecting relevant material from the web and organizing it into a single coherent text. Our algorithm yields significant gains over a structure-agnostic approach. Moreover, our results demonstrate the benefits of structured classification, which outperforms independently trained topical classifiers. Overall, the results of our evaluation combined with our analysis of human edits confirm that the proposed method can effectively produce comprehensive overview articles.

This work opens several directions for future research. Diseases and American Film Actors exhibit fairly consistent article structures, which are successfully captured by a simple template creation process. However, with categories that exhibit structural variability, more sophisticated statistical approaches may be required to produce accurate templates. Moreover, a promising direction is to consider hierarchical discourse formalisms such as RST (Mann and Thompson, 1988) to supplement our template-based approach.

Acknowledgments

The authors acknowledge the support of the NSF (CAREER grant IIS-0448168, grant IIS-0835445, and grant IIS-0835652) and NIH (grant V54LM008748). Thanks to Mike Collins, Julia Hirschberg, and members of the MIT NLP group for their helpful suggestions and comments. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

Eugene Agichtein, Steve Lawrence, and Luis Gravano. 2001. Learning search engine specific query transformations for question answering. In *Proceedings of WWW*, pages 169–178.

Regina Barzilay and Noemie Elhadad. 2003. Sentence alignment for monolingual comparable corpora. In *Proceedings of EMNLP*, pages 25–32.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceedings of HLT-NAACL*, pages 113–120.

Regina Barzilay, Kathleen R. McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of ACL*, pages 550–557.

Fadi Biadsy, Julia Hirschberg, and Elena Filatova. 2008. An unsupervised approach to biography production using wikipedia. In *Proceedings of ACL/HLT*, pages 807–815.

James Clarke and Mirella Lapata. 2007. Modelling compression with discourse constraints. In *Proceedings of EMNLP-CoNLL*, pages 1–11.

William W. Cohen, Robert E. Schapire, and Yoram Singer. 1998. Learning to order things. In *Proceedings of NIPS*, pages 451–457.

Michael Collins. 2002. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of ACL*, pages 489–496.

Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. 1992. *Intoduction to Algorithms*. The MIT Press.

Hal Daumé III and Daniel Marcu. 2005. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Proceedings of HLT/EMNLP*, pages 97–104.

Dina Demner-Fushman and Jimmy Lin. 2007. Answering clinical questions with knowledge-based and statistical techniques. *Computational Linguistics*, 33(1):63–103.

Elena Filatova and John M. Prager. 2005. Tell me what you do and I'll tell you what you are: Learning occupation-related activities for biographies. In *Proceedings of HLT/EMNLP*, pages 113–120.

Elena Filatova, Vasileios Hatzivassiloglou, and Kathleen McKeown. 2006. Automatic creation of domain templates. In *Proceedings of ACL*, pages 207–214.

Atsushi Fujii and Tetsuya Ishikawa. 2004. Summarizing encyclopedic term descriptions on the web. In *Proceedings of COLING*, page 645.

Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. 2000. Multi-document summarization by sentence extraction. In *Proceedings of NAACL-ANLP*, pages 40–48.

Marti A. Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of ACL*, pages 9–16.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of ACL*, pages 74–81.

Inderjeet Mani and Mark T. Maybury. 1999. *Advances in Automatic Text Summarization*. The MIT Press.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Tomasz Marciniak and Michael Strube. 2005. Beyond the pipeline: Discrete optimization in NLP. In *Proceedings of CoNLL*, pages 136–143.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proceedings of EICR*, pages 557–564.

Vivi Nastase and Michael Strube. 2008. Decoding wikipedia categories for knowledge acquisition. In *Proceedings of AAAI*, pages 1219–1224.

Vivi Nastase. 2008. Topic-driven multi-document summarization with encyclopedic knowledge and spreading activation. In *Proceedings of EMNLP*, pages 763–772.

Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of ANLP/NAACL*, pages 21–29.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge.

Benjamin Snyder and Regina Barzilay. 2007. Multiple aspect ranking using the good grief algorithm. In *Proceedings of HLT-NAACL*, pages 300–307.

Ralph M. Weischedel, Jinxi Xu, and Ana Licuanan. 2004. A hybrid approach to answering biographical questions. In *New Directions in Question Answering*, pages 59–70.

Fei Wu and Daniel S. Weld. 2007. Autonomously semantifying wikipedia. In *Proceedings of CIKM*, pages 41–50.

Ying Zhao, George Karypis, and Usama Fayyad. 2005. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery*, 10(2):141–168.

L. Zhou, M. Ticrea, and Eduard Hovy. 2004. Multi-document biography summarization. In *Proceedings of EMNLP*, pages 434–441.

Learning to Tell Tales: A Data-driven Approach to Story Generation

Neil McIntyre and **Mirella Lapata**
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh, EH8 9AB, UK
n.d.mcintyre@sms.ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

Computational story telling has sparked great interest in artificial intelligence, partly because of its relevance to educational and gaming applications. Traditionally, story generators rely on a large repository of background knowledge containing information about the story plot and its characters. This information is detailed and usually hand crafted. In this paper we propose a data-driven approach for generating short children's stories that does not require extensive manual involvement. We create an end-to-end system that realizes the various components of the generation pipeline stochastically. Our system follows a generate-and-and-rank approach where the space of multiple candidate stories is pruned by considering whether they are plausible, interesting, and coherent.

1 Introduction

Recent years have witnessed increased interest in the use of interactive language technology in educational and entertainment applications. Computational story telling could play a key role in these applications by effectively engaging learners and assisting them in creating a story. It could also allow teachers to generate stories on demand that suit their classes' needs. And enhance the entertainment value of role-playing games[1]. The majority of these games come with a set of pre-specified plots that the players must act out. Ideally, the plot should adapt dynamically in response to the players' actions.

Computational story telling has a longstanding tradition in the field of artificial intelligence. Early work has been largely inspired by Propp's (1968) typology of narrative structure. Propp identified in Russian fairy tales a small number of recurring units (e.g., the hero is defeated, the villain causes harm) and rules that could be used to describe their relation (e.g., the hero is pursued and the rescued). Story grammars (Thorndyke, 1977) were initially used to capture Propp's high-level plot elements and character interactions. A large body of more recent work views story generation as a form of agent-based planning (Theune et al., 2003; Fass, 2002; Oinonen et al., 2006). The agents act as characters with a list of goals. They form plans of action and try to fulfill them. Interesting stories emerge as agents' plans interact and cause failures and possible replanning.

Perhaps the biggest challenge faced by computational story generators is the amount of world knowledge required to create compelling stories. A hypothetical system must have information about the characters involved, how they interact, what their goals are, and how they influence their environment. Furthermore, all this information must be complete and error-free if it is to be used as input to a planning algorithm. Traditionally, this knowledge is created by hand, and must be recreated for different domains. Even the simple task of adding a new character requires a whole new set of action descriptions and goals.

A second challenge concerns the generation task itself and the creation of stories characterized by high-quality prose. Most story generation systems focus on generating plot outlines, without considering the actual linguistic structures found in the stories they are trying to mimic (but see Callaway and Lester 2002 for a notable exception). In fact, there seems to be little common ground between story generation and natural language generation (NLG), despite extensive research in both fields. The NLG process (Reiter and Dale, 2000) is often viewed as a pipeline consisting of content planning (selecting and structuring the story's content), microplanning (sentence ag-

[1] A role-playing game (RPG) is a game in which the participants assume the roles of fictional characters and act out an adventure.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 217–225,
Suntec, Singapore, 2-7 August 2009.

gregation, generation of referring expressions, lexical choice), and surface realization (agreement, verb-subject ordering). However, story generation systems typically operate in two phases: (a) creating a plot for the story and (b) transforming it into text (often by means of template-based NLG).

In this paper we address both challenges facing computational story telling. We propose a data-driven approach to story generation that does not require extensive manual involvement. Our goal is to create stories automatically by leveraging knowledge inherent in corpora. Stories within the same genre (e.g., fairy tales, parables) typically have similar structure, characters, events, and vocabularies. It is precisely this type of information we wish to extract and quantify. Of course, building a database of characters and their actions is merely the first step towards creating an automatic story generator. The latter must be able to select which information to include in the story, in what order to present it, how to convert it into English.

Recent work in natural language generation has seen the development of learning methods for realizing each of these tasks automatically without much hand coding. For example, Duboue and McKeown (2002) and Barzilay and Lapata (2005) propose to learn a content planner from a parallel corpus. Mellish et al. (1998) advocate stochastic search methods for document structuring. Stent et al. (2004) learn how to combine the syntactic structure of elementary speech acts into one or more sentences from a corpus of good and bad examples. And Knight and Hatzivassiloglou (1995) use a language model for selecting a fluent sentence among the vast number of surface realizations corresponding to a single semantic representation. Although successful on their own, these methods have not been yet integrated together into an end-to-end probabilistic system. Our work attempts to do this for the story generation task, while bridging the gap between story generators and NLG systems.

Our generator operates over predicate-argument and predicate-predicate co-occurrence statistics gathered from corpora. These are used to produce a large set of candidate stories which are subsequently ranked based on their interestingness and coherence. The top-ranked candidate is selected for presentation and verbalized using a language model interfaced with RealPro (Lavoie and Rambow, 1997), a text generation engine. This generate-and-rank architecture circumvents the complexity of traditional generation systems, where numerous, often conflicting constraints, have to be encoded during development in order to produce a single high-quality output.

This is a fat hen. The hen has a nest in the box. She has eggs in the nest. A cat sees the nest, and can get the eggs.
The sun will soon set. The cows are on their way to the barn. One old cow has a bell on her neck. She sees the dog, but she will not run. The dog is kind to the cows.

Figure 1: Children's stories from McGuffey's Eclectic Primer Reader; it contains primary reading matter to be used in the first year of school work.

As a proof of concept we initially focus on children's stories (see Figure 1 for an example). These stories exhibit several recurrent patterns and are thus amenable to a data-driven approach. Although they have limited vocabulary and non-elaborate syntax, they nevertheless present challenges at almost all stages of the generation process. Also from a practical point of view, children's stories have great potential for educational applications (Robertson and Good, 2003). For instance, the system we describe could serve as an assistant to a person who wants suggestions as to what could happen next in a story. In the remainder of this paper, we first describe the components of our story generator (Section 2) and explain how these are interfaced with our story ranker (Section 3). Next, we present the resources and evaluation methodology used in our experiments (Section 4) and discuss our results (Section 5).

2 The Story Generator

As common in previous work (e.g., Shim and Kim 2002), we assume that our generator operates in an interactive context. Specifically, the user supplies the topic of the story and its desired length. By topic we mean the entities (or characters) around which the story will revolve. These can be a list of nouns such as *dog* and *duck* or a sentence, such as *the dog chases the duck*. The generator next constructs several possible stories involving these entities by consulting a knowledge base containing information about dogs and ducks (e.g., dogs bark, ducks swim) and their interactions (e.g., dogs chase ducks, ducks love dogs). We conceptualize

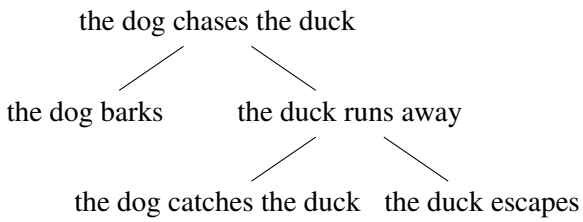


Figure 2: Example of a simplified story tree.

the story generation process as a tree (see Figure 2) whose levels represent different story lengths. For example, a tree of depth 3 will only generate stories with three sentences. The tree encodes many stories efficiently, the nodes correspond to different sentences and there is no sibling order (the tree in Figure 2 can generate three stories). Each sentence in the tree has a score. Story generation amounts to traversing the tree and selecting the nodes with the highest score

Specifically, our story generator applies two distinct search procedures. Although we are ultimately searching for the best overall story at the document level, we must also find the most suitable sentences that can be generated from the knowledge base (see Figure 4). The space of possible stories can increase dramatically depending on the size of the knowledge base so that an exhaustive tree search becomes computationally prohibitive. Fortunately, we can use beam search to prune low-scoring sentences and the stories they generate. For example, we may prefer sentences describing actions that are common for their characters. We also apply two additional criteria in selecting good stories, namely whether they are coherent and interesting. At each depth in the tree we maintain the N-best stories. Once we reach the required length, the highest scoring story is presented to the user. In the following we describe the components of our system in more detail.

2.1 Content Planning

As mentioned earlier our generator has access to a knowledge base recording entities and their interactions. These are essentially predicate argument structures extracted from a corpus. In our experiments this knowledge base was created using the RASP relational parser (Briscoe and Carroll, 2002). We collected all verb-subject, verb-object, verb-adverb, and noun-adjective relations from the parser's output and scored them with the mutual information-based metric proposed in Lin (1998):

dog:SUBJ:bark	whistle:OBJ:dog
dog:SUBJ:bite	treat:OBJ:dog
dog:SUBJ:see	give:OBJ:dog
dog:SUBJ:like	have: OBJ:dog
hungry:ADJ:dog	lovely:ADJ:dog

Table 1: Relations for the noun *dog* with high *MI* scores (SUBJ is a shorthand for subject-of, OBJ for object-of and ADJ for adjective-of).

$$MI = \ln\left(\frac{\| w,r,w' \| \times \| *,r,* \|}{\| w,r,* \| \times \| *,r,w' \|}\right) \quad (1)$$

where w and w' are two words with relation type r. $*$ denotes all words in that particular relation and $\| w,r,w' \|$ represents the number of times w,r,w' occurred in the corpus. These *MI* scores are used to inform the generation system about likely entity relationships at the sentence level. Table 1 shows high scoring relations for the noun *dog* extracted from the corpus used in our experiments (see Section 4 for details).

Note that *MI* weighs binary relations which in some cases may be likely on their own without making sense in a ternary relation. For instance, although both dog:SUBJ:run and president:OBJ:run are probable we may not want to create the sentence *"The dog runs for president"*. Ditransitive verbs pose a similar problem, where two incongruent objects may appear together (the sentence *John gives an apple to the highway* is semantically odd, whereas *John gives an apple to the teacher* would be fine). To help reduce these problems, we need to estimate the likelihood of ternary relations. We therefore calculate the conditional probability:

$$p(a_1,a_2 \mid s,v) = \frac{\| s,v,a_1,a_2 \|}{\| s,v,*,* \|} \quad (2)$$

where s is the subject of verb v, a_1 is the first argument of v and $a2$ is the second argument of v and $v,s,a_1 \neq \varepsilon$. When a verb takes two arguments, we first consult (2), to see if the combination is likely before backing off to (1).

The knowledge base described above can only inform the generation system about relationships on the sentence level. However, a story created simply by concatenating sentences in isolation will often be incoherent. Investigations into the interpretation of narrative discourse (Asher and Lascarides, 2003) have shown that lexical information plays an important role in determining

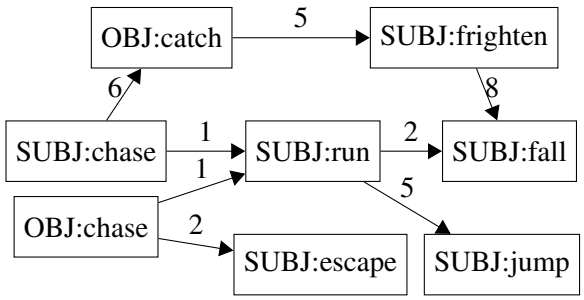


Figure 3: Graph encoding (partially ordered) chains of events

the discourse relations between propositions. Although we don't have an explicit model of rhetorical relations and their effects on sentence ordering, we capture the lexical inter-dependencies between sentences by focusing on events (verbs) and their precedence relationships in the corpus. For every entity in our training corpus we extract event chains similar to those proposed by Chambers and Jurafsky (2008). Specifically, we identify the events every entity relates to and record their (partial) order. We assume that verbs sharing the same arguments are more likely to be semantically related than verbs with no arguments in common. For example, if we know that someone steals and then runs, we may expect the next action to be that they hide or that they are caught.

In order to track entities and their associated events throughout a text, we first resolve entity mentions using OpenNLP[2]. The list of events performed by co-referring entities and their grammatical relation (i.e., subject or object) are subsequently stored in a graph. The edges between event nodes are scored using the *MI* equation given in (1). A fragment of the action graph is shown in Figure 3 (for simplicity, the edges in the example are weighted with co-occurrence frequencies). Contrary to Chambers and Jurafsky (2008) we do not learn *global* narrative chains over an entire corpus. Currently, we consider *local* chains of length two and three (i.e., chains of two or three events sharing grammatical arguments). The generator consults the graph when selecting a verb for an entity. It will favor verbs that are part of an event chain (e.g., SUBJ:chase → SUBJ:run → SUBJ:fall in Figure 3). This way, the search space is effectively pruned as finding a suitable verb in the current sentence is influenced by the choice of verb in the next sentence.

[2]See `http://opennlp.sourceforge.net/`.

2.2 Sentence Planning

So far we have described how we gather knowledge about entities and their interactions, which must be subsequently combined into a sentence. The backbone of our sentence planner is a grammar with subcategorization information which we collected from the lexicon created by Korhonen and Briscoe (2006) and the COMLEX dictionary (Grishman et al., 1994). The grammar rules act as templates. They each take a verb as their head and propose ways of filling its argument slots. This means that when generating a story, the choice of verb will affect the structure of the sentence. The subcategorization templates are weighted by their probability of occurrence in the reference dictionaries. This allows the system to prefer less elaborate grammatical structures. The grammar rules were converted to a format compatible with our surface realizer (see Section 2.3) and include information pertaining to mood, agreement, argument role, etc.

Our sentence planner aggregates together information from the knowledge base, without however generating referring expressions. Although this would be a natural extension, we initially wanted to assess whether the stochastic approach advocated here is feasible at all, before venturing towards more ambitious components.

2.3 Surface Realization

The surface realization process is performed by RealPro (Lavoie and Rambow (1997)). The system takes an abstract sentence representation and transforms it into English. There are several grammatical issues that will affect the final realization of the sentence. For nouns we must decide whether they are singular or plural, whether they are preceded by a definite or indefinite article or with no article at all. Adverbs can either be pre-verbal or post-verbal. There is also the issue of selecting an appropriate tense for our generated sentences, however, we simply assume all sentences are in the present tense. Since we do not know a priori which of these parameters will result in a grammatical sentence, we generate all possible combinations and select the most likely one according to a language model. We used the SRI toolkit to train a trigram language model on the British National Corpus, with interpolated Kneser-Ney smoothing and perplexity as the scoring metric for the generated sentences.

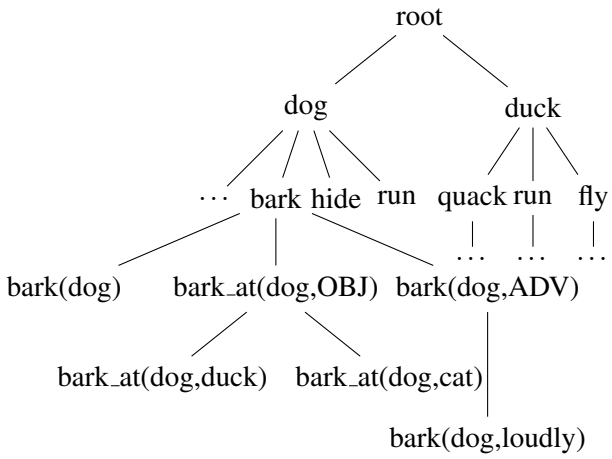


Figure 4: Simplified generation example for the input sentence *the dog chases the duck*.

2.4 Sentence Generation Example

It is best to illustrate the generation procedure with a simple example (see Figure 4). Given the sentence *the dog chases the duck* as input, our generator assumes that either *dog* or *duck* will be the subject of the following sentence. This is a somewhat simplistic attempt at generating coherent stories. Centering (Grosz et al., 1995) and other discourse theories argue that topical entities are likely to appear in prominent syntactic positions such as subject or object. Next, we select verbs from the knowledge base that take the words *duck* and *dog* as their subject (e.g., *bark, run, fly*). Our beam search procedure will reduce the list of verbs to a small subset by giving preference to those that are likely to follow *chase* and have *duck* and *dog* as their subjects or objects.

The sentence planner gives a set of possible frames for these verbs which may introduce additional entities (see Figure 4). For example, *bark* can be intransitive or take an object or adverbial complement. We select an object for *bark*, by retrieving from the knowledge base the set of objects it co-occurs with. Our surface realizer will take structures like "bark(dog,loudly)", "bark_at(dog,cat)", "bark_at(dog,duck)" and generate the sentences *the dog barks loudly*, *the dog barks at the cat* and *the dog barks at the duck*. This procedure is repeated to create a list of possible candidates for the third sentence, and so on.

As Figure 4 illustrates, there are many candidate sentences for each entity. In default of generating all of these exhaustively, our system utilizes the *MI* scores from the knowledge base to guide the search. So, at each choice point in the generation process, e.g., when selecting a verb for an entity or a frame for a verb, we consider the *N* best alternatives assuming that these are most likely to appear in a good story.

3 Story Ranking

We have so far described most modules of our story generator, save one important component, namely the story ranker. As explained earlier, our generator produces stories stochastically, by relying on co-occurrence frequencies collected from the training corpus. However, there is no guarantee that these stories will be interesting or coherent. Engaging stories have some element of surprise and originality in them (Turner, 1994). Our stories may simply contain a list of actions typically performed by the story characters. Or in the worst case, actions that make no sense when collated together.

Ideally, we would like to be able to discern interesting stories from tedious ones. Another important consideration is their coherence. We have to ensure that the discourse smoothly transitions from one topic to the next. To remedy this, we developed two ranking functions that assess the candidate stories based on their interest and coherence. Following previous work (Stent et al., 2004; Barzilay and Lapata, 2007) we learn these ranking functions from training data (i.e., stories labeled with numeric values for interestingness and coherence).

Interest Model A stumbling block to assessing how interesting a story may be, is that the very notion of interestingness is subjective and not very well understood. Although people can judge fairly reliably whether they like or dislike a story, they have more difficulty isolating what exactly makes it interesting. Furthermore, there are virtually no empirical studies investigating the linguistic (surface level) correlates of interestingness. We therefore conducted an experiment where we asked participants to rate a set of human authored stories in terms of interest. Our stories were Aesop's fables since they resemble the stories we wish to generate. They are fairly short (average length was 3.7 sentences) and with a few characters. We asked participants to judge 40 fables on a set of criteria: plot, events, characters, coherence and interest (using a 5-point rating scale). The fables were split into 5 sets of 8; each participant was randomly assigned one of the 5 sets to judge. We obtained rat-

ings (440 in total) from 55 participants, using the WebExp[3] experimental software.

We next investigated if easily observable syntactic and lexical features were correlated with interest. Participants gave the fables an average interest rating of 3.05. For each story we extracted the number of tokens and types for nouns, verbs, adverbs and adjectives as well as the number of verb-subject and verb-object relations. Using the MRC Psycholinguistic database[4] tokens were also annotated along the following dimensions: number of letters (NLET), number of phonemes (NPHON), number of syllables (NSYL), written frequency in the Brown corpus (Kucera and Francis 1967; K-F-FREQ), number of categories in the Brown corpus (K-F-NCATS), number of samples in the Brown corpus (K-F-NSAMP), familiarity (FAM), concreteness (CONC), imagery (IMAG), age of acquisition (AOA), and meaningfulness (MEANC and MEANP).

Correlation analysis was used to assess the degree of linear relationship between interest ratings and the above features. The results are shown in Table 2. As can be seen the highest predictor is the number of objects in a story, followed by the number of noun tokens and types. Imagery, concreteness and familiarity all seem to be significantly correlated with interest. Story length was not a significant predictor. Regressing the best predictors from Table 2 against the interest ratings yields a correlation coefficient of 0.608 ($p < 0.05$). The predictors account uniquely for 37.2% of the variance in interest ratings. Overall, these results indicate that a model of story interest can be trained using shallow syntactic and lexical features. We used the Aesop's fables with the human ratings as training data from which we extracted features that shown to be significant predictors in our correlation analysis. Word-based features were summed in order to obtain a representation for the entire story. We used Joachims's (2002) SVM^{light} package for training with cross-validation (all parameters set to their default values). The model achieved a correlation of 0.948 (Kendall's tau) with the human ratings on the test set.

Coherence Model As well as being interesting we have to ensure that our stories make sense to the reader. Here, we focus on *local coherence*, which captures text organization at the level of sentence to sentence transitions. We created a model of local coherence using using the Entity Grid approach described in Barzilay and Lapata (2007). This approach represents each document as a two-dimensional array in which the columns correspond to entities and the rows to sentences. Each cell indicates whether an entity appears in a given sentence or not and whether it is a subject, object or neither. This entity grid is then converted into a vector of entity transition sequences. Training the model required examples of both coherent and incoherent stories. An artificial training set was created by permuting the sentences of coherent stories, under the assumption that the original story is more coherent than its permutations. The model was trained and tested on the Andrew Lang fairy tales collection[5] on a random split of the data. It ranked the original stories higher than their corresponding permutations 67.40% of the time.

	Interest		Interest
NTokens	0.188**	NLET	0.120*
NTypes	0.173**	NPHON	0.140**
VTokens	0.123*	NSYL	0.125**
VTypes	0.154**	K-F-FREQ	0.054
AdvTokens	0.056	K-F-NCATS	0.137**
AdvTypes	0.051	K-F-NSAMP	0.103*
AdjTokens	0.035	FAM	0.162**
AdjTypes	0.029	CONC	0.166**
NumSubj	0.150**	IMAG	0.173**
NumObj	0.240**	AOA	0.111*
MEANC	0.169**	MEANP	0.156**

Table 2: Correlation values for the human ratings of interest against syntactic and lexical features; $^{*}: p < 0.05$, $^{**}: p < 0.01$.

4 Experimental Setup

In this section we present our experimental set-up for assessing the performance of our story generator. We give details on our training corpus, system, parameters (such as the width of the beam), the baselines used for comparison, and explain how our system output was evaluated.

Corpus The generator was trained on 437 stories from the Andrew Lang fairy tale corpus.[6] The stories had an average length of 125.18 sentences. The corpus contained 15,789 word tokens. We

[3] See http://www.webexp.info/.

[4] http://www.psy.uwa.edu.au/mrcdatabase/uwa_mrc.htm

[5] Aesop's fables were too short to learn a coherence model.

[6] See http://www.mythfolklore.net/andrewlang/.

discarded word tokens that did not appear in the Children's Printed Word Database[7], a database of printed word frequencies as read by children aged between five and nine.

Story search When searching the story space, we set the beam width to 500. This means that we allow only 500 sentences to be considered at a particular depth before generating the next set of sentences in the story. For each entity we select the five most likely events and event sequences. Analogously, we consider the five most likely subcategorization templates for each verb. Considerable latitude is available when applying the ranking functions. We may use only one of them, or one after the other, or both of them. To evaluate which system configuration was best, we asked two human evaluators to rate (on a 1–5 scale) stories produced in the following conditions: (a) score the candidate stories using the interest function first and then coherence (and vice versa), (b) score the stories simultaneously using both rankers and select the story with the highest score. We also examined how best to prune the search space, i.e., by selecting the highest scoring stories, the lowest scoring one, or simply at random. We created ten stories of length five using the fairy tale corpus for each permutation of the parameters. The results showed that the evaluators preferred the version of the system that applied both rankers simultaneously and maintained the highest scoring stories in the beam.

Baselines We compared our system against two simpler alternatives. The first one does not use a beam. Instead, it decides deterministically how to generate a story on the basis of the most likely predicate-argument and predicate-predicate counts in the knowledge base. The second one creates a story randomly without taking any co-occurrence frequency into account. Neither of these systems therefore creates more than one story hypothesis whilst generating.

Evaluation The system generated stories for 10 input sentences. These were created using commonly occurring sentences in the fairy tales corpus (e.g., *The family has the baby*, *The monkey climbs the tree*, *The giant guards the child*). Each system generated one story for each sentence resulting in 30 (3×10) stories for evaluation. All stories had the same length, namely five sentences. Human judges (21 in total) were asked to rate the stories on a scale of 1 to 5 for fluency (was the sentence grammatical?), coherence (does the story make sense overall?) and interest (how interesting is the story?). The stories were presented in random order. Participants were told that all stories were generated by a computer program. They were instructed to rate more favorably interesting stories, stories that were comprehensible and overall grammatical.

[7] http://www.essex.ac.uk/psychology/cpwd/

System	Fluency	Coherence	Interest
Random	1.95*	2.40*	2.09*
Deterministic	2.06*	2.53*	2.09*
Rank-based	2.20	2.65	2.20

Table 3: Human evaluation results: mean story ratings for three versions of our system; *: significantly different from Rank-based.

5 Results

Our results are summarized in Table 3 which lists the average human ratings for the three systems. We performed an Analysis of Variance (ANOVA) to examine the effect of system type on the story generation task. Statistical tests were carried out on the mean of the ratings shown in Table 3 for fluency, coherence, and interest. We observed a reliable effect of system type by subjects and items on all three dimensions. Post-hoc Tukey tests revealed that the stories created with our rank-based system are perceived as significantly better in terms of fluency, interest, and coherence than those generated by both the deterministic and random systems ($\alpha < 0.05$). The deterministic system is not significantly better than the random one except in terms of coherence.

These results are not entirely surprising. The deterministic system maintains a local restricted view of what constitutes a good story. It creates a story by selecting isolated entity-event relationships with high *MI* scores. As a result, the stories are unlikely to have a good plot. Moreover, it tends to primarily favor verb-object or verb-subject relations, since these are most frequent in the corpus. The stories thus have little structural variation and feel repetitive. The random system uses even less information in generating a story (entity-action relationships are chosen at random without taking note of the *MI* scores). In contrast to these baselines, the rank-based system assesses candidate stories more globally. It thus favors coherent stories, with varied word choice and structure.

	The family has the baby	*The giant guards the child*
Random	The family has the baby. The family is how to empty up to a fault. The baby vanishes into the cave. The family meets with a stranger. The baby says for the boy to fancy the creature.	The giant guards the child. The child calls for the window to order the giant. The child suffers from a pleasure. The child longer hides the forest. The child reaches presently.
Determ	The family has the baby. The family rounds up the waist. The family comes in. The family wonders. The family meets with the terrace.	The giant guards the child. The child rescues the clutch. The child beats down on a drum. The child feels out of a shock. The child hears from the giant.
Rank-based	The family has the baby. The baby is to seat the lady at the back. The baby sees the lady in the family. The family marries a lady for the triumph. The family quickly wishes the lady vanishes.	The giant guards the child. The child rescues the son from the power. The child begs the son for a pardon. The giant cries that the son laughs the happiness out of death. The child hears if the happiness tells a story.

Table 4: Stories generated by the random, deterministic, and rank-based systems.

A note of caution here concerns referring expressions which our systems cannot at the moment generate. This may have disadvantaged the stories overall, rendering them stylistically awkward.

The stories generated by both the deterministic and random systems are perceived as less interesting in comparison to the rank-based system. This indicates that taking interest into account is a promising direction even though the overall interestingness of the stories we generate is somewhat low (see third column in Table 3). Our interest ranking function was trained on well-formed human authored stories. It is therefore possible that the ranker was not as effective as it could be simply because it was applied to out-of-domain data. An interesting extension which we plan for the future is to evaluate the performance of a ranker trained on machine generated stories.

Table 4 illustrates the stories generated by each system for two input sentences. The rank-based stories read better overall and are more coherent. Our subjects also gave them high interest scores. The deterministic system tends to select simplistic sentences which although read well by themselves do not lead to an overall narrative. Interestingly, the story generated by the random system for the input *The family has the baby*, scored high on interest too. The story indeed contains interesting imagery (e.g. *The baby vanishes into the cave*) although some of the sentences are syntactically odd (e.g. *The family is how to empty up to a fault*).

6 Conclusions and Future Work

In this paper we proposed a novel method to computational story telling. Our approach has three key features. Firstly, story plot is created dynamically by consulting an automatically created knowledge base. Secondly, our generator realizes the various components of the generation pipeline stochastically, without extensive manual coding. Thirdly, we generate and store multiple stories efficiently in a tree data structure. Story creation amounts to traversing the tree and selecting the nodes with the highest score. We develop two scoring functions that rate stories in terms of how coherent and interesting they are. Experimental results show that these bring improvements over versions of the system that rely solely on the knowledge base. Overall, our results indicate that the overgeneration-and-ranking approach advocated here is viable in producing short stories that exhibit narrative structure. As our system can be easily rertrained on different corpora, it can potentially generate stories that vary in vocabulary, style, genre, and domain.

An important future direction concerns a more detailed assessment of our search procedure. Currently we don't have a good estimate of the type of stories being overlooked due to the restrictions we impose on the search space. An appealing alternative is the use of Genetic Algorithms (Goldberg, 1989). The operations of mutation and crossover have the potential of creating more varied and original stories. Our generator would also benefit from an explicit model of causality which is currently approximated by the entity chains. Such a model could be created from existing resources such as ConceptNet (Liu and Davenport, 2004), a freely available commonsense knowledge base. Finally, improvements such as the generation of referring expressions and the modeling of selectional restrictions would create more fluent stories.

Acknowledgements The authors acknowledge the support of EPSRC (grant GR/T04540/01). We are grateful to Richard Kittredge for his help with RealPro. Special thanks to Johanna Moore for insightful comments and suggestions.

References

Asher, Nicholas and Alex Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.

Barzilay, Regina and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the HLT/EMNLP*. Vancouver, pages 331–338.

Barzilay, Regina and Mirella Lapata. 2007. Modeling local coherence: An entity-based approach. *Computational Linguistics* 34(1):1–34.

Briscoe, E. and J. Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd LREC*. Las Palmas, Gran Canaria, pages 1499–1504.

Callaway, Charles B. and James C. Lester. 2002. Narrative prose generation. *Artificial Intelligence* 2(139):213–252.

Chambers, Nathanael and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*. Columbus, OH, pages 789–797.

Duboue, Pablo A. and Kathleen R. McKeown. 2002. Content planner construction via evolutionary algorithms and a corpus-based fitness function. In *Proceedings of the 2nd INLG*. Ramapo Mountains, NY.

Fass, S. 2002. *Virtual Storyteller: An Approach to Computational Storytelling*. Master's thesis, Dept. of Computer Science, University of Twente.

Goldberg, David E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

Grishman, Ralph, Catherine Macleod, and Adam Meyers. 1994. COMLEX syntax: Building a computational lexicon. In *Proceedings of the 15th COLING*. Kyoto, Japan, pages 268–272.

Grosz, Barbara J., Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics* 21(2):203–225.

Joachims, Thorsten. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD*. Edmonton, AL, pages 133–142.

Knight, Kevin and Vasileios Hatzivassiloglou. 1995. Two-level, many-paths generation. In *Proceedings of the 33rd ACL*. Cambridge, MA, pages 252–260.

Korhonen, Y. Krymolowski, A. and E.J. Briscoe. 2006. A large subcategorization lexicon for natural language processing applications. In *Proceedings of the 5th LREC*. Genova, Italy.

Kucera, Henry and Nelson Francis. 1967. *Computational Analysis of Present-day American English*. Brown University Press, Providence, RI.

Lavoie, Benoit and Owen Rambow. 1997. A fast and portable realizer for text generation systems. In *Proceedings of the 5th ANCL*. Washington, D.C., pages 265–268.

Lin, Dekang. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th COLING*. Montréal, QC, pages 768–774.

Liu, Hugo and Glorianna Davenport. 2004. ConceptNet: a practical commonsense reasoning toolkit. *BT Technology Journal* 22(4):211–226.

Mellish, Chris, Alisdair Knott, Jon Oberlander, and Mick O'Donnell. 1998. Experiments using stochastic search for text planning. In Eduard Hovy, editor, *Proceedings of the 9th INLG*. New Brunswick, NJ, pages 98–107.

Oinonen, K.M., M. Theune, A. Nijholt, and J.R.R. Uijlings. 2006. Designing a story database for use in automatic story generation. In R. Harper, M. Rauterberg, and M. Combetto, editors, *Entertainment Computing – ICEC 2006*. Springer Verlag, Berlin, volume 4161 of *Lecture Notes in Computer Science*, pages 298–301.

Propp, Vladimir. 1968. *The Morphology of Folk Tale*. University of Texas Press, Austin, TX.

Reiter, E and R Dale. 2000. *Building Natural-Language Generation Systems*. Cambridge University Press.

Robertson, Judy and Judith Good. 2003. Ghostwriter: A narrative virtual environment for children. In *Proceedings of IDC2003*. Preston, England, pages 85–91.

Shim, Yunju and Minkoo Kim. 2002. Automatic short story generator based on autonomous agents. In *Proceedings of PRIMA*. London, UK, pages 151–162.

Stent, Amanda, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd ACL*. Barcelona, Spain, pages 79–86.

Theune, M., S. Faas, D.K.J. Heylen, and A. Nijholt. 2003. The virtual storyteller: Story creation by intelligent agents. In S. Gbel, N. Braun, U. Spierling, J. Dechau, and H. Diener, editors, *TIDSE-2003*. Fraunhofer IRB Verlag, Darmstadt, pages 204–215.

Thorndyke, Perry W. 1977. Cognitive structures in comprehension and memory of narrative discourse. *Cognitive Psychology* 9(1):77–110.

Turner, Scott T. 1994. *The creative process: A computer model of storytelling and creativity*. Erlbaum, Hillsdale, NJ.

Recognizing Stances in Online Debates

Swapna Somasundaran
Dept. of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
swapna@cs.pitt.edu

Janyce Wiebe
Dept. of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
wiebe@cs.pitt.edu

Abstract

This paper presents an unsupervised opinion analysis method for *debate-side classification*, i.e., recognizing which stance a person is taking in an online debate. In order to handle the complexities of this genre, we mine the web to learn associations that are indicative of opinion stances in debates. We combine this knowledge with discourse information, and formulate the debate side classification task as an Integer Linear Programming problem. Our results show that our method is substantially better than challenging baseline methods.

1 Introduction

This paper presents a method for *debate-side classification*, i.e., recognizing which stance a person is taking in an online debate posting. In online debate forums, people debate issues, express their preferences, and argue why their viewpoint is right. In addition to expressing positive sentiments about one's preference, a key strategy is also to express negative sentiments about the other side. For example, in the debate *"which mobile phone is better: iPhone or Blackberry,"* a participant on the iPhone side may explicitly assert and rationalize why the iPhone is better, and, alternatively, also argue why the Blackberry is worse. Thus, to recognize stances, we need to consider not only which opinions are positive and negative, but also what the opinions are about (their *targets*).

Participants directly express their opinions, such as *"The iPhone is cool,"* but, more often, they mention associated aspects. Some aspects are particular to one topic (e.g., Active-X is part of IE but not Firefox), and so distinguish between them. But even an aspect the topics share may distinguish between them, because people who are positive toward one topic may value that aspect more. For example, both the iPhone and Blackberry have keyboards, but we observed in our corpus that positive opinions about the keyboard are associated with the pro Blackberry stance. Thus, we need to find distinguishing aspects, which the topics may or may not share.

Complicating the picture further, participants may concede positive aspects of the opposing issue or topic, without coming out in favor of it, and they may concede negative aspects of the issue or topic they support. For example, in the following sentence, the speaker says positive things about the iPhone, even though he does not prefer it: *"Yes, the iPhone may be cool to take it out and play with and show off, but past that, it offers nothing."* Thus, we need to consider discourse relations to sort out which sentiments in fact reveal the writer's stance, and which are merely concessions.

Many opinion mining approaches find negative and positive words in a document, and aggregate their counts to determine the final document polarity, ignoring the targets of the opinions. Some work in product review mining finds aspects of a central topic, and summarizes opinions with respect to these aspects. However, they do not find distinguishing factors associated with a preference for a stance. Finally, while other opinion analysis systems have considered discourse information, they have not distinguished between concessionary and non-concessionary opinions when determining the overall stance of a document.

This work proposes an unsupervised opinion analysis method to address the challenges described above. First, for each debate side, we mine the web for opinion-target pairs that are associated with a preference for that side. This information is employed, in conjunction with discourse information, in an Integer Linear Programming (ILP) framework. This framework combines the individual pieces of information to arrive at debate-side

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 226–234,
Suntec, Singapore, 2-7 August 2009.

classifications of posts in online debates.

The remainder of this paper is organized as follows. We introduce our debate genre in Section 2 and describe our method in Section 3. We present the experiments in Section 4 and analyze the results in Section 5. Related work is in Section 6, and the conclusions are in Section 7.

2 The Debate Genre

In this section, we describe our debate data, and elaborate on characteristic ways of expressing opinions in this genre. For our current work, we use the online debates from the website http://www.convinceme.net.[1]

In this work, we deal only with dual-sided, dual-topic debates about named entities, for example iPhone vs. Blackberry, where $topic_1$ = iPhone, $topic_2$ =Blackberry, $side_1$ = pro-iPhone, and $side_2$=pro-Blackberry.

Our test data consists of posts of 4 debates: Windows vs. Mac, Firefox vs. Internet Explorer, Firefox vs. Opera, and Sony Ps3 vs. Nintendo Wii. The iPhone vs. Blackberry debate and two other debates, were used as development data.

Given below are examples of debate posts. Post 1 is taken from the iPhone vs. Blackberry debate, Post 2 is from the Firefox vs. Internet Explorer debate, and Post 3 is from the Windows vs. Mac debate:

(1) While the iPhone may appeal to younger generations and the BB to older, there is no way it is geared towards a less rich population. In fact it's exactly the opposite. It's a gimmick. The initial purchase may be half the price, but when all is said and done you pay at least $200 more for the 3g.

(2) In-line spell check...helps me with big words like onomatopoeia

(3) Apples are nice computers with an exceptional interface. Vista will close the gap on the interface some but Apple still has the prettiest, most pleasing interface and most likely will for the next several years.

2.1 Observations

As described in Section 1, the debate genre poses significant challenges to opinion analysis. This subsection elaborates upon some of the complexities.

Multiple polarities to argue for a side. Debate participants, in advocating their choice, switch back and forth between their opinions towards the sides. This makes it difficult for approaches that use only positive and negative word counts to decide which side the post is on. Posts 1 and 3 illustrate this phenomenon.

Sentiments towards both sides (topics) within a single post. The above phenomenon gives rise to an additional problem: often, conflicting sides (and topics) are addressed within the same post, sometimes within the same sentence. The second sentence of Post 3 illustrates this, as it has opinions about both Windows and Mac.

Differentiating aspects and personal preferences. People seldom repeatedly mention the topic/side; they show their evaluations indirectly, by evaluating aspects of each topic/side. *Differentiating* aspects determine the debate-post's side.

Some aspects are unique to one side/topic or the other, e.g., "3g" in Example 1 and "inline spell check" in Example 2. However, the debates are about topics that belong to the same domain and which therefore share many aspects. Hence, a purely ontological approach of finding "has-a" and "is-a" relations, or an approach looking only for product specifications, would not be sufficient for finding differentiating features.

When the two topics do share an aspect (e.g., a keyboard in the iPhone vs. Blackberry debate), the writer may perceive it to be more positive for one than the other. And, if the writer values that aspect, it will influence his or her overall stance. For example, many people prefer the Blackberry keyboard over the iPhone keyboard; people to whom phone keyboards are important are more likely to prefer the Blackberry.

Concessions. While debating, participants often refer to and acknowledge the viewpoints of the opposing side. However, they do not endorse this rival opinion. Uniform treatment of all opinions in a post would obviously cause errors in such cases. The first sentence of Example 1 is an instance of this phenomenon. The participant concedes that the iPhone appeals to young consumers, but this positive opinion is opposite to his overall stance.

[1] http://www.forandagainst.com and http://www.createdebate.com are other similar debating websites.

DIRECT OBJECT Rule: dobj(opinion, target) In words: The target is the direct object of the opinion Example: I love$_{opinion1}$ Firefox$_{target1}$ and defended$_{opinion2}$ it$_{target2}$
NOMINAL SUBJECT Rule: nsubj(opinion, target) In words: The target is the subject of the opinion Example: IE$_{target}$ breaks$_{opinion}$ with everything.
ADJECTIVAL MODIFIER Rule: amod(target, opinion) In words: The opinion is an adjectival modifier of the target Example: The annoying$_{opinion}$ popup$_{target}$
PREPOSITIONAL OBJECT Rule: if prep(target1,IN) $\Rightarrow$ pobj(IN, target2) In words: The prepositional object of a known target is also a target of the same opinion Example: The annoying$_{opinion}$ popup$_{target1}$ in IE$_{target2}$ ("popup" and "IE" are targets of "annoying")
RECURSIVE MODIFIERS Rule: if conj(adj2, opinion$_{adj1}$) $\Rightarrow$ amod(target, adj2) In words: If the opinion is an adjective (adj1) and it is conjoined with another adjective (adj2), then the opinion is tied to what adj2 modifies Example: It is a powerful$_{opinion(adj1)}$ and easy$_{opinion(adj2)}$ application$_{target}$ ("powerful" is attached to the target "application" via the adjective "easy")

Table 1: Examples of syntactic rules for finding targets of opinions

3 Method

We propose an unsupervised approach to classifying the stance of a post in a dual-topic debate. For this, we first use a web corpus to learn preferences that are likely to be associated with a side. These learned preferences are then employed in conjunction with discourse constraints to identify the side for a given post.

3.1 Finding Opinions and Pairing them with Targets

We need to find opinions and pair them with targets, both to mine the web for general preferences and to classify the stance of a debate post. We use straightforward methods, as these tasks are not the focus of this paper.

To find opinions, we look up words in a subjectivity lexicon: all instances of those words are treated as opinions. An opinion is assigned the prior polarity that is listed for that word in the lexicon, except that, if the prior polarity is positive or negative, and the instance is modified by a negation word (e.g., "not"), then the polarity of that instance is reversed. We use the subjectivity lexicon of (Wilson et al., 2005),[2] which contains approximately 8000 words which may be used to express opinions. Each entry consists of a subjective word, its prior polarity (positive ($^+$), negative ($^-$), neutral (*)), morphological information, and part of speech information.

To pair opinions with targets, we built a rule-based system based on dependency parse information. The dependency parses are obtained using the Stanford parser.[3] We developed the syntactic rules on separate data that is not used elsewhere in this paper. Table 1 illustrates some of these rules. Note that the rules are constructed (and explained in Table 1) with respect to the grammatical relation notations of the Stanford parser. As illustrated in the table, it is possible for an opinion to have more than one target. In such cases, the single opinion results in multiple opinion-target pairs, one for each target.

Once these opinion-target pairs are created, we mask the identity of the opinion word, replacing the word with its polarity. Thus, the opinion-target pair is converted to a polarity-target pair. For instance, "pleasing-interface" is converted to $interface^+$. This abstraction is essential for handling the sparseness of the data.

3.2 Learning aspects and preferences from the web

We observed in our development data that people highlight the aspects of topics that are the bases for their stances, both positive opinions toward aspects of the preferred topic, and negative opinions toward aspects of the dispreferred one. Thus, we decided to mine the web for aspects associated with a side in the debate, and then use that information to recognize the stances expressed in individual posts.

Previous work mined web data for aspects associated with topics (Hu and Liu, 2004; Popescu et al., 2005). In our work, we search for aspects associated with a topic, but particularized to polarity. Not all aspects associated with a topic are

[2] Available at http://www.cs.pitt.edu/mpqa.

[3] http://nlp.stanford.edu/software/lex-parser.shtml.

	$side_1$ (pro-iPhone)		$side_2$ (pro-blackberry)	
$term^p$	$P(iPhone^+\|term^p)$	$P(blackberry^-\|term^p)$	$P(iPhone^-\|term^p)$	$P(blackberry^+\|term^p)$
$storm^+$	0.227	0.068	0.022	0.613
$storm^-$	0.062	0.843	0.06	0.03
$phone^+$	0.333	0.176	0.137	0.313
$e\text{-}mail^+$	0	0.333	0.166	0.5
$ipod^+$	0.5	0	0.33	0
$battery^-$	0	0	0.666	0.333
$network^-$	0.333	0	0.666	0
$keyboard^+$	0.09	0.12	0	0.718
$keyboard^-$	0.25	0.25	0.125	0.375

Table 2: Probabilities learned from the web corpus (iPhone vs. blackberry debate)

discriminative with respect to stance; we hypothesized that, by including polarity, we would be more likely to find useful associations. An aspect may be associated with both of the debate topics, but not, by itself, be discriminative between stances toward the topics. However, *opinions* toward that aspect might discriminate between them. Thus, the basic unit in our web mining process is a polarity-target pair. Polarity-target pairs which explicitly mention one of the topics are used to anchor the mining process. Opinions about relevant aspects are gathered from the surrounding context.

For each debate, we downloaded weblogs and forums that talk about the main topics (corresponding to the sides) of that debate. For example, for the iPhone vs. Blackberry debate, we search the web for pages containing "iPhone" and "Blackberry." We used the Yahoo search API and imposed the search restriction that the pages should contain both topics in the http URL. This ensured that we downloaded relevant pages. An average of 3000 documents were downloaded per debate.

We apply the method described in Section 3.1 to the downloaded web pages. That is, we find all instances of words in the lexicon, extract their targets, and mask the words with their polarities, yielding polarity-target pairs. For example, suppose the sentence *"The interface is pleasing"* is in the corpus. The system extracts the pair "pleasing-interface," which is masked to "positive-interface," which we notate as $interface^+$. If the target in a polarity-target pair happens to be one of the topics, we select the polarity-target pairs in its vicinity for further processing (the rest are discarded). The intuition behind this is that, if someone expresses an opinion about a topic, he or she is likely to follow it up with reasons for that opinion. The sentiments in the surrounding context thus reveal factors that influence the preference or dislike towards the topic. We define the vicinity as the same sentence plus the following 5 sentences.

Each unique target word $target_i$ in the web corpus, i.e., each word used as the target of an opinion one or more times, is processed to generate the following conditional probabilities.

$$P(topic_j^q|target_i^p) = \frac{\#(topic_j^q, target_i^p)}{\#target_i^p} \quad (1)$$

where $p = \{^+, ^-, ^*\}$ and $q = \{^+, ^-, ^*\}$ denote the polarities of the target and the topic, respectively; $j = \{1, 2\}$; and $i = \{1...M\}$, where M is the number of unique targets in the corpus. For example, $P(Mac^+|interface^+)$ is the probability that "interface" is the target of a positive opinion that is in the vicinity of a positive opinion toward "Mac."

Table 2 lists some of the probabilities learned by this approach. (Note that the neutral cases are not shown.)

3.2.1 Interpreting the learned probabilities

Table 2 contains examples of the learned probabilities. These probabilities align with what we qualitatively found in our development data. For example, the opinions towards "Storm" essentially follow the opinions towards "Blackberry;" that is, positive opinions toward "Storm" are usually found in the vicinity of positive opinions toward "Blackberry," and negative opinions toward "Storm" are usually found in the vicinity of negative opinions toward "Blackberry" (for example, in the row for $storm^+$, $P(blackberry^+|storm^+)$ is much higher than the other probabilities). Thus, an opinion expressed about "Storm" is usually the opinion one has toward "Blackberry." This is expected, as Storm is a type of Blackberry. A similar example is $ipod^+$, which follows the opinion toward the iPhone. This is interesting because an

iPod is not a phone; the association is due to preference for the brand. In contrast, the probability distribution for "phone" does not show a preference for any one side, even though both iPhone and Blackberry are phones. This indicates that opinions towards phones in general will not be able to distinguish between the debate sides.

Another interesting case is illustrated by the probabilities for "e-mail." People who like e-mail capability are more likely to praise the Blackberry, or even criticize the iPhone — they would thus belong to the pro-Blackberry camp.

While we noted earlier that positive evaluations of keyboards are associated with positive evaluations of the Blackberry (by far the highest probability in that row), negative evaluations of keyboards, are, however, *not* a strong discriminating factor.

For the other entries in the table, we see that criticisms of batteries and the phone network are more associated with negative sentiments towards the iPhones.

The possibility of these various cases motivates our approach, in which opinions and their polarities are considered when searching for associations between debate topics and their aspects.

3.3 Debate-side classification

Once we have the probabilities collected from the web, we can build our classifier to classify the debate posts.

Here again, we use the process described in Section 3.1 to extract polarity-target pairs for each opinion expressed in the post. Let N be the number of instances of polarity-target pairs in the post. For each instance I_j ($j = \{1...N\}$), we look up the learned probabilities of Section 3.2 to create two scores, w_j and u_j:

$$w_j = P(topic_1^+|target_i^p) + P(topic_2^-|target_i^p) \quad (2)$$

$$u_j = P(topic_1^-|target_i^p) + P(topic_2^+|target_i^p) \quad (3)$$

where $target_i^p$ is the polarity-target type of which I_j is an instance.

Score w_j corresponds to $side_1$ and u_j corresponds to $side_2$. A point to note is that, if a target word is repeated, and it occurs in different polarity-target instances, it is counted as a separate instance each time — that is, here we account for tokens, not types. Via Equations 2 and 3, we interpret the observed polarity-target instance I_j in terms of debate sides.

We formulate the problem of finding the overall side of the post as an Integer Linear Programming (ILP) problem. The side that maximizes the overall side-score for the post, given all the N instances I_j, is chosen by maximizing the objective function

$$\sum_{j=1}^{N}(w_j x_j + u_j y_j) \quad (4)$$

subject to the following constraints

$$x_j \in \{0,1\},\ \forall j \quad (5)$$

$$y_j \in \{0,1\},\ \forall j \quad (6)$$

$$x_j + y_j = 1,\ \forall j \quad (7)$$

$$x_j - x_{j-1} = 0,\ j \in \{2..N\} \quad (8)$$

$$y_j - y_{j-1} = 0,\ j \in \{2..N\} \quad (9)$$

Equations 5 and 6 implement binary constraints. Equation 7 enforces the constraint that each I_j can belong to exactly one side. Finally, Equations 8 and 9 ensure that a single side is chosen for the entire post.

3.4 Accounting for concession

As described in Section 2, debate participants often acknowledge the opinions held by the opposing side. We recognize such discourse constructs using the Penn Discourse Treebank (Prasad et al., 2007) list of discourse connectives. In particular, we use the list of connectives from the Concession and Contra-expectation category. Examples of connectives in these categories are "while," "nonetheless," "however," and "even if." We use approximations to finding the arguments to the discourse connectives (*ARG1* and *ARG2* in Penn Discourse Treebank terms). If the connective is mid-sentence, the part of the sentence prior to the connective is considered conceded, and the part that follows the connective is considered non-conceded. An example is the second sentence of Example 3. If, on the other hand, the connective is sentence-initial, the sentence is split at the first comma that occurs mid sentence. The first part is considered conceded, and the second part is considered non-conceded. An example is the first sentence of Example 1.

The opinions occurring in the conceded part are interpreted in reverse. That is, the weights corresponding to the sides w_j and u_j are interchanged in equation 4. Thus, conceded opinions are effectively made to count towards the opposing side.

4 Experiments

On http://www.convinceme.net, the html page for each debate contains side information for each post ($side_1$ is blue in color and $side_2$ is green). This gives us automatically labeled data for our evaluations. For each of the 4 debates in our test set, we use posts with at least 5 sentences for evaluation.

4.1 Baselines

We implemented two baselines: the OpTopic system that uses topic information only, and the OpPMI system that uses topic as well as related word (noun) information. All systems use the same lexicon, as well as exactly the same processes for opinion finding and opinion-target pairing.

The OpTopic system This system considers only explicit mentions of the topic for the opinion analysis. Thus, for this system, the step of opinion-target pairing only finds all $topic_1^+$, $topic_1^-$, $topic_2^+$, $topic_2^-$ instances in the post (where, for example, an instance of $topic_1^+$ is a positive opinion whose target is explicitly $topic_1$). The polarity-topic pairs are counted for each debate side according to the following equations.

$$score(side_1) = \#topic_1^+ + \#topic_2^- \quad (10)$$

$$score(side_2) = \#topic_1^- + \#topic_2^+ \quad (11)$$

The post is assigned the side with the higher score.

The OpPMI system This system finds opinion-target pairs for not only the topics, but also for the words in the debate that are significantly related to either of the topics.

We find semantic relatedness of each noun in the post with the two main topics of the debate by calculating the Pointwise Mutual Information (PMI) between the term and each topic over the entire web corpus. We use the API provided by the Measures of Semantic Relatedness (MSR)[4] engine for this purpose. The MSR engine issues Google queries to retrieve documents and finds the PMI between any two given words. Table 3 lists PMIs between the topics and the words from Table 2.

Each noun k is assigned to the topic with the higher PMI score. That is, if
$PMI(topic_1,\text{k}) > PMI(topic_2,\text{k}) \Rightarrow \text{k}= topic_1$
and if
$PMI(topic_2,\text{k}) > PMI(topic_1,\text{k}) \Rightarrow \text{k}= topic_2$
Next, the polarity-target pairs are found for the post, as before, and Equations 10 and 11 are used to assign a side to the post as in the OpTopic system, except that here, related nouns are also counted as instances of their associated topics.

[4] http://cwl-projects.cogsci.rpi.edu/msr/

word	iPhone	blackberry
storm	0.923	0.941
phone	0.908	0.885
e-mail	0.522	0.623
ipod	0.909	0.976
battery	0.974	0.927
network	0.658	0.961
keyboard	0.961	0.983

Table 3: PMI of words with the topics

4.2 Results

Performance is measured using the following metrics: Accuracy ($\frac{\#Correct}{\#Total\ posts}$), Precision ($\frac{\#Correct}{\#guessed}$), Recall ($\frac{\#Correct}{\#relevant}$) and F-measure ($\frac{2*Precision*Recall}{(Precision+Recall)}$).

In our task, it is desirable to make a prediction for all the posts; hence $\#relevant = \#Total\ posts$. This results in Recall and Accuracy being the same. However, all of the systems do not classify a post if the post does not contain the information it needs. Thus, $\#guessed \leq \#Total\ posts$, and Precision is not the same as Accuracy.

Table 4 reports the performance of four systems on the test data: the two baselines, our method using the preferences learned from the web corpus (OpPr) and the method additionally using discourse information to reverse conceded opinions.

The OpTopic has low recall. This is expected, because it relies only on opinions explicitly toward the topics.

The OpPMI has better recall than OpTopic; however, the precision drops for some debates. We believe this is due to the addition of noise. This result suggests that not all terms that are relevant to a topic are useful for determining the debate side.

Finally, both of the OpPr systems are better than both baselines in Accuracy as well as F-measure for all four debates.

The accuracy of the full OpPr system improves, on average, by 35 percentage points over the OpTopic system, and by 20 percentage points over the

OpPMI system. The F-measure improves, on average, by 25 percentage points over the OpTopic system, and by 17 percentage points over the OpPMI system. Note that in 3 out of 4 of the debates, the full system is able to make a guess for all of the posts (hence, the metrics all have the same values).

In three of the four debates, the system using concession handling described in Section 3.4 outperforms the system without it, providing evidence that our treatment of concessions is effective. On average, there is a 3 percentage point improvement in Accuracy, 5 percentage point improvement in Precision and 5 percentage point improvement in F-measure due to the added concession information.

	OpTopic	OpPMI	OpPr	OpPr + Disc
Firefox Vs Internet explorer (62 posts)				
Acc	33.87	53.23	64.52	66.13
Prec	67.74	60.0	64.52	66.13
Rec	33.87	53.23	64.52	66.13
F1	45.16	56.41	64.52	66.13
Windows vs. Mac (15 posts)				
Acc	13.33	46.67	66.67	66.67
Prec	40.0	53.85	66.67	66.67
Rec	13.33	46.67	66.67	66.67
F1	20.0	50.00	66.67	66.67
SonyPs3 vs. Wii (36 posts)				
Acc	33.33	33.33	56.25	61.11
Prec	80.0	46.15	56.25	68.75
Rec	33.33	33.33	50.0	61.11
F1	47.06	38.71	52.94	64.71
Opera vs. Firefox (4 posts)				
Acc	25.0	50.0	75.0	100.0
Prec	33.33	100	75.0	100.0
Rec	25.0	50	75.0	100.0
F1	28.57	66.67	75.0	100.0

Table 4: Performance of the systems on the test data

5 Discussion

In this section, we discuss the results from the previous section and describe the sources of errors.

As reported in the previous section, the OpPr system outperforms both the OpTopic and the OpPMI systems. In order to analyze why OpPr outperforms OpPMI, we need to compare Tables 2 and 3. Table 2 reports the conditional probabilities learned from the web corpus for polarity-target pairs used in OpPr, and Table 3 reports the PMI of these same targets with the debate topics used in OpPMI. First, we observe that the PMI numbers are intuitive, in that all the words, except for "e-mail," show a high PMI relatedness to both topics. All of them are indeed semantically related to the domain. Additionally, we see that some conclusions of the OpPMI system are similar to those of the OpPr system, for example, that "Storm" is more closely related to the Blackberry than the iPhone.

However, notice two cases: the PMI values for "phone" and "e-mail" are intuitive, but they may cause errors in debate analysis. Because the iPhone and the Blackberry are both phones, the word "phone" does not have any distinguishing power in debates. On the other hand, the PMI measure of "e-mail" suggests that it is not closely related to the debate topics, though it is, in fact, a desirable feature for smart phone users, even more so with Blackberry users. The PMI measure does not reflect this.

The "network" aspect shows a comparatively greater relatedness to the blackberry than to the iPhone. Thus, OpPMI uses it as a proxy for the Blackberry. This may be erroneous, however, because negative opinions towards "network" are more indicative of negative opinions towards iPhones, a fact revealed by Table 2.

In general, even if the OpPMI system knows what topic the given word is more related to, it still does not know what the opinion towards that word *means* in the debate scenario. The OpPr system, on the other hand, is able to map it to a debate side.

5.1 Errors

False lexicon hits. The lexicon is word based, but, as shown by (Wiebe and Mihalcea, 2006; Su and Markert, 2008), many subjective words have both objective and subjective senses. Thus, one major source of errors is a false hit of a word in the lexicon.

Opinion-target pairing. The syntactic rule-based opinion-target pairing system is a large source of errors in the OpPr as well as the baseline systems. Product review mining work has explored finding opinions with respect to, or in conjunction with, aspects (Hu and Liu, 2004; Popescu et al., 2005); however, in our work, we need to find

information in the other direction – that is, given the opinion, what is the opinion about. Stoyanov and Cardie (2008) work on opinion co-reference; however, we need to identify the specific target.

Pragmatic opinions. Some of the errors are due to the fact that the opinions expressed in the post are pragmatic. This becomes a problem especially when the debate post is small, and we have few other lexical clues in the post. The following post is an example:

(4) The blackberry is something like $150 and the iPhone is $500. I don't think it's worth it. You could buy a iPod separate and have a boatload of extra money left over.

In this example, the participant mentions the difference in the prices in the first sentence. This sentence implies a negative opinion towards the iPhone. However, recognizing this would require a system to have extensive world knowledge. In the second sentence, the lexicon does hit the word "worth," and, using syntactic rules, we can determine it is negated. However, the opinion-target pairing system only tells us that the opinion is tied to the "it." A co-reference system would be needed to tie the "it" to "iPhone" in the first sentence.

6 Related Work

Several researchers have worked on similar tasks. Kim and Hovy (2007) predict the results of an election by analyzing forums discussing the elections. Theirs is a supervised bag-of-words system using unigrams, bigrams, and trigrams as features. In contrast, our approach is unsupervised, and exploits different types of information. Bansal et al. (2008) predict the vote from congressional floor debates using agreement/disagreement features. We do not model inter-personal exchanges; instead, we model factors that influence stance taking. Lin at al (2006) identify opposing perspectives. Though apparently related at the task level, perspectives as they define them are not the same as opinions. Their approach does not involve any opinion analysis. Fujii and Ishikawa (2006) also work with arguments. However, their focus is on argument visualization rather than on recognizing stances.

Other researchers have also mined data to learn associations among products and features. In their work on mining opinions in comparative sentences, Ganapathibhotla and Liu (2008) look for user preferences for one product's features over another's. We do not exploit comparative constructs, but rather probabilistic associations. Thus, our approach and theirs are complementary. A number of works in product review mining (Hu and Liu, 2004; Popescu et al., 2005; Kobayashi et al., 2005; Bloom et al., 2007) automatically find features of the reviewed products. However, our approach is novel in that it learns and exploits associations among opinion/polarity, topics, and aspects.

Several researchers have recognized the important role discourse plays in opinion analysis (Polanyi and Zaenen, 2005; Snyder and Barzilay, 2007; Somasundaran et al., 2008; Asher et al., 2008; Sadamitsu et al., 2008). However, previous work did not account for concessions in determining whether an opinion supports one side or the other.

More sophisticated approaches to identifying opinions and recognizing their contextual polarity have been published (e.g., (Wilson et al., 2005; Ikeda et al., 2008; Sadamitsu et al., 2008)). Those components are not the focus of our work.

7 Conclusions

This paper addresses challenges faced by opinion analysis in the debate genre. In our method, factors that influence the choice of a debate side are learned by mining a web corpus for opinions. This knowledge is exploited in an unsupervised method for classifying the side taken by a post, which also accounts for concessionary opinions.

Our results corroborate our hypothesis that finding relations between aspects associated with a topic, but particularized to polarity, is more effective than finding relations between topics and aspects alone. The system that implements this information, mined from the web, outperforms the web PMI-based baseline. Our hypothesis that addressing concessionary opinions is useful is also corroborated by improved performance.

Acknowledgments

This research was supported in part by the Department of Homeland Security under grant N000140710152. We would also like to thank Vladislav D. Veksler for help with the MSR engine, and the anonymous reviewers for their helpful comments.

References

Nicholas Asher, Farah Benamara, and Yvette Yannick Mathieu. 2008. Distilling opinion in discourse: A preliminary study. In *Coling 2008: Companion volume: Posters and Demonstrations*, pages 5–8, Manchester, UK, August.

Mohit Bansal, Claire Cardie, and Lillian Lee. 2008. The power of negative thinking: Exploiting label disagreement in the min-cut classification framework. In *Proceedings of COLING: Companion volume: Posters*.

Kenneth Bloom, Navendu Garg, and Shlomo Argamon. 2007. Extracting appraisal expressions. In *HLT-NAACL 2007*, pages 308–315, Rochester, NY.

Atsushi Fujii and Tetsuya Ishikawa. 2006. A system for summarizing and visualizing arguments in subjective documents: Toward supporting decision making. In *Proceedings of the Workshop on Sentiment and Subjectivity in Text*, pages 15–22, Sydney, Australia, July. Association for Computational Linguistics.

Murthy Ganapathibhotla and Bing Liu. 2008. Mining opinions in comparative sentences. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 241–248, Manchester, UK, August.

Minqing Hu and Bing Liu. 2004. Mining opinion features in customer reviews. In *AAAI-2004*.

Daisuke Ikeda, Hiroya Takamura, Lev-Arie Ratinov, and Manabu Okumura. 2008. Learning to shift the polarity of words for sentiment classification. In *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP)*.

Soo-Min Kim and Eduard Hovy. 2007. Crystal: Analyzing predictive opinions on the web. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1056–1064.

Nozomi Kobayashi, Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2005. Opinion extraction using a learning-based anaphora resolution technique. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP-05), poster*, pages 175–180.

Wei-Hao Lin, Theresa Wilson, Janyce Wiebe, and Alexander Hauptmann. 2006. Which side are you on? Identifying perspectives at the document and sentence levels. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-2006)*, pages 109–116, New York, New York.

Livia Polanyi and Annie Zaenen. 2005. Contextual valence shifters. In *Computing Attitude and Affect in Text*. Springer.

Ana-Maria Popescu, Bao Nguyen, and Oren Etzioni. 2005. OPINE: Extracting product features and opinions from reviews. In *Proceedings of HLT/EMNLP 2005 Interactive Demonstrations*, pages 32–33, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

R. Prasad, E. Miltsakaki, N. Dinesh, A. Lee, A. Joshi, L. Robaldo, and B. Webber, 2007. *PDTB 2.0 Annotation Manual*.

Kugatsu Sadamitsu, Satoshi Sekine, and Mikio Yamamoto. 2008. Sentiment analysis based on probabilistic models using inter-sentence information. In European Language Resources Association (ELRA), editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May.

Benjamin Snyder and Regina Barzilay. 2007. Multiple aspect ranking using the good grief algorithm. In *Proceedings of NAACL-2007*.

Swapna Somasundaran, Janyce Wiebe, and Josef Ruppenhofer. 2008. Discourse level opinion interpretation. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 801–808, Manchester, UK, August.

Veselin Stoyanov and Claire Cardie. 2008. Topic identification for fine-grained opinion analysis. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 817–824, Manchester, UK, August. Coling 2008 Organizing Committee.

Fangzhong Su and Katja Markert. 2008. From word to sense: a case study of subjectivity recognition. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-2008)*, Manchester, UK, August.

Janyce Wiebe and Rada Mihalcea. 2006. Word sense and subjectivity. In *Proceedings of COLING-ACL 2006*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT-EMNLP*, pages 347–354, Vancouver, Canada.

Co-Training for Cross-Lingual Sentiment Classification

Xiaojun Wan
Institute of Compute Science and Technology & Key Laboratory of Computational Linguistics, MOE
Peking University, Beijing 100871, China
wanxiaojun@icst.pku.edu.cn

Abstract

The lack of Chinese sentiment corpora limits the research progress on Chinese sentiment classification. However, there are many freely available English sentiment corpora on the Web. This paper focuses on the problem of cross-lingual sentiment classification, which leverages an available English corpus for Chinese sentiment classification by using the English corpus as training data. Machine translation services are used for eliminating the language gap between the training set and test set, and English features and Chinese features are considered as two independent views of the classification problem. We propose a co-training approach to making use of unlabeled Chinese data. Experimental results show the effectiveness of the proposed approach, which can outperform the standard inductive classifiers and the transductive classifiers.

1 Introduction

Sentiment classification is the task of identifying the sentiment polarity of a given text. The sentiment polarity is usually positive or negative and the text genre is usually product review. In recent years, sentiment classification has drawn much attention in the NLP field and it has many useful applications, such as opinion mining and summarization (Liu et al., 2005; Ku et al., 2006; Titov and McDonald, 2008).

To date, a variety of corpus-based methods have been developed for sentiment classification. The methods usually rely heavily on an annotated corpus for training the sentiment classifier. The sentiment corpora are considered as the most valuable resources for the sentiment classification task. However, such resources in different languages are very imbalanced. Because most previous work focuses on English sentiment classification, many annotated corpora for English sentiment classification are freely available on the Web. However, the annotated corpora for Chinese sentiment classification are scarce and it is not a trivial task to manually label reliable Chinese sentiment corpora. The challenge before us is how to leverage rich English corpora for Chinese sentiment classification. In this study, we focus on the problem of cross-lingual sentiment classification, which leverages only English training data for supervised sentiment classification of Chinese product reviews, without using any Chinese resources. Note that the above problem is not only defined for Chinese sentiment classification, but also for various sentiment analysis tasks in other different languages.

Though pilot studies have been performed to make use of English corpora for subjectivity classification in other languages (Mihalcea et al., 2007; Banea et al., 2008), the methods are very straightforward by directly employing an inductive classifier (e.g. SVM, NB), and the classification performance is far from satisfactory because of the language gap between the original language and the translated language.

In this study, we propose a co-training approach to improving the classification accuracy of polarity identification of Chinese product reviews. Unlabeled Chinese reviews can be fully leveraged in the proposed approach. First, machine translation services are used to translate English training reviews into Chinese reviews and also translate Chinese test reviews and additional unlabeled reviews into English reviews. Then, we can view the classification problem in two independent views: Chinese view with only Chinese features and English view with only English features. We then use the co-training approach to making full use of the two redundant views of features. The SVM classifier is adopted as the basic classifier in the proposed approach. Experimental results show that the proposed approach can outperform the baseline inductive classifiers and the more advanced transductive classifiers.

The rest of this paper is organized as follows: Section 2 introduces related work. The proposed

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 235–243,
Suntec, Singapore, 2-7 August 2009.

co-training approach is described in detail in Section 3. Section 4 shows the experimental results. Lastly we conclude this paper in Section 5.

2 Related Work

2.1 Sentiment Classification

Sentiment classification can be performed on words, sentences or documents. In this paper we focus on document sentiment classification. The methods for document sentiment classification can be generally categorized into lexicon-based and corpus-based.

Lexicon-based methods usually involve deriving a sentiment measure for text based on sentiment lexicons. Turney (2002) predicates the sentiment orientation of a review by the average semantic orientation of the phrases in the review that contain adjectives or adverbs, which is denoted as the semantic oriented method. Kim and Hovy (2004) build three models to assign a sentiment category to a given sentence by combining the individual sentiments of sentiment-bearing words. Hiroshi et al. (2004) use the technique of deep language analysis for machine translation to extract sentiment units in text documents. Kennedy and Inkpen (2006) determine the sentiment of a customer review by counting positive and negative terms and taking into account contextual valence shifters, such as negations and intensifiers. Devitt and Ahmad (2007) explore a computable metric of positive or negative polarity in financial news text.

Corpus-based methods usually consider the sentiment analysis task as a classification task and they use a labeled corpus to train a sentiment classifier. Since the work of Pang et al. (2002), various classification models and linguistic features have been proposed to improve the classification performance (Pang and Lee, 2004; Mullen and Collier, 2004; Wilson et al., 2005; Read, 2005). Most recently, McDonald et al. (2007) investigate a structured model for jointly classifying the sentiment of text at varying levels of granularity. Blitzer et al. (2007) investigate domain adaptation for sentiment classifiers, focusing on online reviews for different types of products. Andreevskaia and Bergler (2008) present a new system consisting of the ensemble of a corpus-based classifier and a lexicon-based classifier with precision-based vote weighting.

Chinese sentiment analysis has also been studied (Tsou et al., 2005; Ye et al., 2006; Li and Sun, 2007) and most such work uses similar lexicon-based or corpus-based methods for Chinese sentiment classification.

To date, several pilot studies have been performed to leverage rich English resources for sentiment analysis in other languages. Standard Naïve Bayes and SVM classifiers have been applied for subjectivity classification in Romanian (Mihalcea et al., 2007; Banea et al., 2008), and the results show that automatic translation is a viable alternative for the construction of resources and tools for subjectivity analysis in a new target language. Wan (2008) focuses on leveraging both Chinese and English lexicons to improve Chinese sentiment analysis by using lexicon-based methods. In this study, we focus on improving the corpus-based method for cross-lingual sentiment classification of Chinese product reviews by developing novel approaches.

2.2 Cross-Domain Text Classification

Cross-domain text classification can be considered as a more general task than cross-lingual sentiment classification. In the problem of cross-domain text classification, the labeled and unlabeled data come from different domains, and their underlying distributions are often different from each other, which violates the basic assumption of traditional classification learning.

To date, many semi-supervised learning algorithms have been developed for addressing the cross-domain text classification problem by transferring knowledge across domains, including Transductive SVM (Joachims, 1999), EM(Nigam et al., 2000), EM-based Naïve Bayes classifier (Dai et al., 2007a), Topic-bridged PLSA (Xue et al., 2008), Co-Clustering based classification (Dai et al., 2007b), two-stage approach (Jiang and Zhai, 2007). DauméIII and Marcu (2006) introduce a statistical formulation of this problem in terms of a simple mixture model.

In particular, several previous studies focus on the problem of cross-lingual text classification, which can be considered as a special case of general cross-domain text classification. Bel et al. (2003) present practical and cost-effective solutions. A few novel models have been proposed to address the problem, e.g. the EM-based algorithm (Rigutini et al., 2005), the information bottleneck approach (Ling et al., 2008), the multilingual domain models (Gliozzo and Strapparava, 2005), etc. To the best of our knowledge, co-training has not yet been investigated for cross-domain or cross-lingual text classification.

3 The Co-Training Approach

3.1 Overview

The purpose of our approach is to make use of the annotated English corpus for sentiment polarity identification of Chinese reviews in a supervised framework, without using any Chinese resources. Given the labeled English reviews and unlabeled Chinese reviews, two straightforward methods for addressing the problem are as follows:

1) We first learn a classifier based on the labeled English reviews, and then translate Chinese reviews into English reviews. Lastly, we use the classifier to classify the translated English reviews.

2) We first translate the labeled English reviews into Chinese reviews, and then learn a classifier based on the translated Chinese reviews with labels. Lastly, we use the classifier to classify the unlabeled Chinese reviews.

The above two methods have been used in (Banea et al., 2008) for Romanian subjectivity analysis, but the experimental results are not very promising. As shown in our experiments, the above two methods do not perform well for Chinese sentiment classification, either, because the underlying distribution between the original language and the translated language are different.

In order to address the above problem, we propose to use the co-training approach to make use of some amounts of unlabeled Chinese reviews to improve the classification accuracy. The co-training approach can make full use of both the English features and the Chinese features in a unified framework. The framework of the proposed approach is illustrated in Figure 1.

The framework consists of a training phase and a classification phase. In the training phase, the input is the labeled English reviews and some amounts of unlabeled Chinese reviews[1]. The labeled English reviews are translated into labeled Chinese reviews, and the unlabeled Chinese reviews are translated into unlabeled English reviews, by using machine translation services. Therefore, each review is associated with an English version and a Chinese version. The English features and the Chinese features for each review are considered two independent and redundant views of the review. The co-training algorithm is then applied to learn two classifiers and finally the two classifiers are combined into a single sentiment classifier. In the classification phase, each unlabeled Chinese review for testing is first translated into English review, and then the learned classifier is applied to classify the review into either positive or negative.

The steps of review translation and the co-training algorithm are described in details in the next sections, respectively.

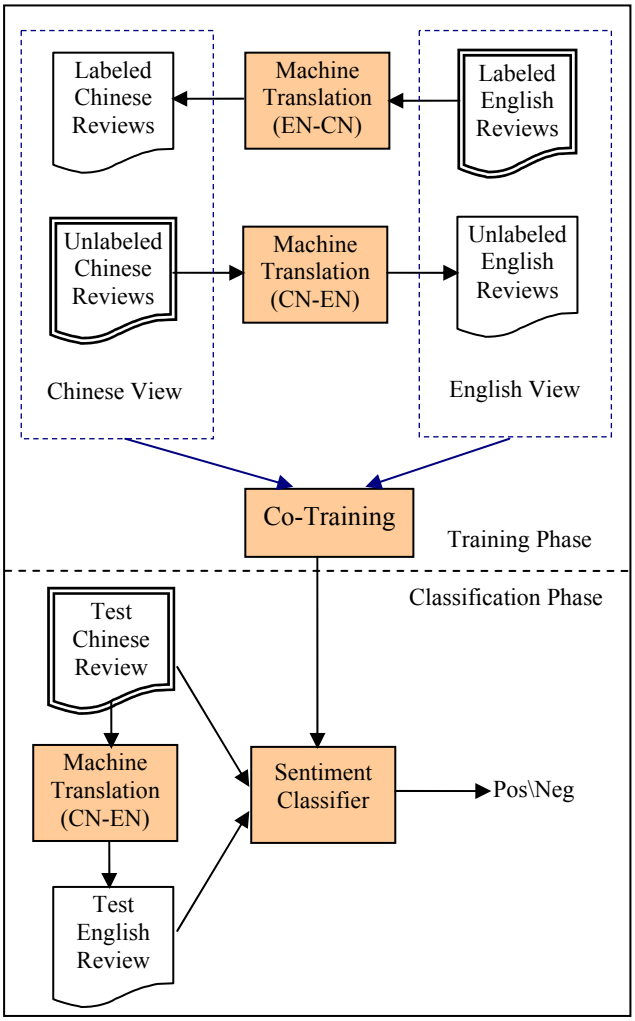


Figure 1. Framework of the proposed approach

3.2 Review Translation

In order to overcome the language gap, we must translate one language into another language. Fortunately, machine translation techniques have been well developed in the NLP field, though the translation performance is far from satisfactory. A few commercial machine translation services can be publicly accessed, e.g. *Google Translate*[2], *Yahoo Babel Fish*[3] and *Windows Live Translate*[4].

[1] The unlabeled Chinese reviews used for co-training do not include the unlabeled Chinese reviews for testing, i.e., the Chinese reviews for testing are blind to the training phase.

[2] http://translate.google.com/translate_t

[3] http://babelfish.yahoo.com/translate_txt

[4] http://www.windowslivetranslator.com/

In this study, we adopt *Google Translate* for both English-to-Chinese Translation and Chinese-to-English Translation, because it is one of the state-of-the-art commercial machine translation systems used today. *Google Translate* applies statistical learning techniques to build a translation model based on both monolingual text in the target language and aligned text consisting of examples of human translations between the languages.

3.3 The Co-Training Algorithm

The co-training algorithm (Blum and Mitchell, 1998) is a typical bootstrapping method, which starts with a set of labeled data, and increase the amount of annotated data using some amounts of unlabeled data in an incremental way. One important aspect of co-training is that two conditional independent views are required for co-training to work, but the independence assumption can be relaxed. Till now, co-training has been successfully applied to statistical parsing (Sarkar, 2001), reference resolution (Ng and Cardie, 2003), part of speech tagging (Clark et al., 2003), word sense disambiguation (Mihalcea, 2004) and email classification (Kiritchenko and Matwin, 2001).

In the context of cross-lingual sentiment classification, each labeled English review or unlabeled Chinese review has two views of features: English features and Chinese features. Here, a review is used to indicate both its Chinese version and its English version, until stated otherwise. The co-training algorithm is illustrated in Figure 2. In the algorithm, the class distribution in the labeled data is maintained by balancing the parameter values of p and n at each iteration.

The intuition of the co-training algorithm is that if one classifier can confidently predict the class of an example, which is very similar to some of labeled ones, it can provide one more training example for the other classifier. But, of course, if this example happens to be easy to be classified by the first classifier, it does not mean that this example will be easy to be classified by the second classifier, so the second classifier will get useful information to improve itself and vice versa (Kiritchenko and Matwin, 2001).

In the co-training algorithm, a basic classification algorithm is required to construct C_{en} and C_{cn}. Typical text classifiers include Support Vector Machine (SVM), Naïve Bayes (NB), Maximum Entropy (ME), K-Nearest Neighbor (KNN), etc. In this study, we adopt the widely-used SVM classifier (Joachims, 2002). Viewing input data as two sets of vectors in a feature space, SVM constructs a separating hyperplane in the space by maximizing the margin between the two data sets. The English or Chinese features used in this study include both unigrams and bigrams[5] and the feature weight is simply set to term frequency[6]. Feature selection methods (e.g. Document Frequency (DF), Information Gain (IG), and Mutual Information (MI)) can be used for dimension reduction. But we use all the features in the experiments for comparative analysis, because there is no significant performance improvement after applying the feature selection techniques in our empirical study. The output value of the SVM classifier for a review indicates the confidence level of the review's classification. Usually, the sentiment polarity of a review is indicated by the sign of the prediction value.

Given:
- F_{en} and F_{cn} are redundantly sufficient sets of features, where F_{en} represents the English features, F_{cn} represents the Chinese features;
- L is a set of labeled training reviews;
- U is a set of unlabeled reviews;

Loop for I iterations:
1. Learn the first classifier C_{en} from L based on F_{en};
2. Use C_{en} to label reviews from U based on F_{en};
3. Choose p positive and n negative the most confidently predicted reviews E_{en} from U;
4. Learn the second classifier C_{cn} from L based on F_{cn};
5. Use C_{cn} to label reviews from U based on F_{cn};
6. Choose p positive and n negative the most confidently predicted reviews E_{cn} from U;
7. Removes reviews $E_{en} \cup E_{cn}$ from U[7];
8. Add reviews $E_{en} \cup E_{cn}$ with the corresponding labels to L;

Figure 2. The co-training algorithm

In the training phase, the co-training algorithm learns two separate classifiers: C_{en} and C_{cn}.

[5] For Chinese text, a unigram refers to a Chinese word and a bigram refers to two adjacent Chinese words.

[6] Term frequency performs better than TFIDF by our empirical analysis.

[7] Note that the examples with conflicting labels are not included in $E_{en} \cup E_{cn}$ In other words, if an example is in both E_{en} and E_{cn}, but the labels for the example is conflicting, the example will be excluded from $E_{en} \cup E_{cn}$.

Therefore, in the classification phase, we can obtain two prediction values for a test review. We normalize the prediction values into [-1, 1] by dividing the maximum absolute value. Finally, the average of the normalized values is used as the overall prediction value of the review.

4 Empirical Evaluation

4.1 Evaluation Setup

4.1.1 Data set

The following three datasets were collected and used in the experiments:

Test Set (Labeled Chinese Reviews): In order to assess the performance of the proposed approach, we collected and labeled 886 product reviews (451 positive reviews + 435 negative reviews) from a popular Chinese IT product web site-IT168[8]. The reviews focused on such products as mp3 players, mobile phones, digital camera and laptop computers.

Training Set (Labeled English Reviews): There are many labeled English corpora available on the Web and we used the corpus constructed for multi-domain sentiment classification (Blitzer et al., 2007)[9], because the corpus was large-scale and it was within similar domains as the test set. The dataset consisted of 8000 Amazon product reviews (4000 positive reviews + 4000 negative reviews) for four different product types: books, DVDs, electronics and kitchen appliances.

Unlabeled Set (Unlabeled Chinese Reviews): We downloaded additional 1000 Chinese product reviews from IT168 and used the reviews as the unlabeled set. Therefore, the unlabeled set and the test set were in the same domain and had similar underlying feature distributions.

Each Chinese review was translated into English review, and each English review was translated into Chinese review. Therefore, each review has two independent views: English view and Chinese view. A review is represented by both its English view and its Chinese view.

Note that the training set and the unlabeled set are used in the training phase, while the test set is blind to the training phase.

4.1.2 Evaluation Metric

We used the standard precision, recall and F-measure to measure the performance of positive and negative class, respectively, and used the accuracy metric to measure the overall performance of the system. The metrics are defined the same as in general text categorization.

4.1.3 Baseline Methods

In the experiments, the proposed co-training approach (CoTrain) is compared with the following baseline methods:

SVM(CN): This method applies the inductive SVM with only Chinese features for sentiment classification in the Chinese view. Only English-to-Chinese translation is needed. And the unlabeled set is not used.

SVM(EN): This method applies the inductive SVM with only English features for sentiment classification in the English view. Only Chinese-to-English translation is needed. And the unlabeled set is not used.

SVM(ENCN1): This method applies the inductive SVM with both English and Chinese features for sentiment classification in the two views. Both English-to-Chinese and Chinese-to-English translations are required. And the unlabeled set is not used.

SVM(ENCN2): This method combines the results of SVM(EN) and SVM(CN) by averaging the prediction values in the same way with the co-training approach.

TSVM(CN): This method applies the transductive SVM with only Chinese features for sentiment classification in the Chinese view. Only English-to-Chinese translation is needed. And the unlabeled set is used.

TSVM(EN): This method applies the transductive SVM with only English features for sentiment classification in the English view. Only Chinese-to-English translation is needed. And the unlabeled set is used.

TSVM(ENCN1): This method applies the transductive SVM with both English and Chinese features for sentiment classification in the two views. Both English-to-Chinese and Chinese-to-English translations are required. And the unlabeled set is used.

TSVM(ENCN2): This method combines the results of TSVM(EN) and TSVM(CN) by averaging the prediction values.

Note that the first four methods are straightforward methods used in previous work, while the latter four methods are strong baselines because the transductive SVM has been widely used for improving the classification accuracy by leveraging additional unlabeled examples.

[8] http://www.it168.com
[9] http://www.cis.upenn.edu/~mdredze/datasets/sentiment/

4.2 Evaluation Results

4.2.1 Method Comparison

In the experiments, we first compare the proposed co-training approach (I=40 and p=n=5) with the eight baseline methods. The three parameters in the co-training approach are empirically set by considering the total number (i.e. 1000) of the unlabeled Chinese reviews. In our empirical study, the proposed approach can perform well with a wide range of parameter values, which will be shown later. Table 1 shows the comparison results.

Seen from the table, the proposed co-training approach outperforms all eight baseline methods over all metrics. Among the eight baselines, the best one is TSVM(ENCN2), which combines the results of two transductive SVM classifiers. Actually, TSVM(ENCN2) is similar to CoTrain because CoTrain also combines the results of two classifiers in the same way. However, the co-training approach can train two more effective classifiers, and the accuracy values of the component English and Chinese classifiers are 0.775 and 0.790, respectively, which are higher than the corresponding TSVM classifiers. Overall, the use of transductive learning and the combination of English and Chinese views are beneficial to the final classification accuracy, and the co-training approach is more suitable for making use of the unlabeled Chinese reviews than the transductive SVM.

4.2.2 Influences of Iteration Number (I)

Figure 3 shows the accuracy curve of the co-training approach (Combined Classifier) with different numbers of iterations. The iteration number I is varied from 1 to 80. When I is set to 1, the co-training approach is degenerated into SVM(ENCN2). The accuracy curves of the component English and Chinese classifiers learned in the co-training approach are also shown in the figure. We can see that the proposed co-training approach can outperform the best baseline-TSVM(ENCN2) after 20 iterations. After a large number of iterations, the performance of the co-training approach decreases because noisy training examples may be selected from the remaining unlabeled set. Finally, the performance of the approach does not change any more, because the algorithm runs out of all possible examples in the unlabeled set. Fortunately, the proposed approach performs well with a wide range of iteration numbers. We can also see that the two component classifier has similar trends with the co-training approach. It is encouraging that the component Chinese classifier alone can perform better than the best baseline when the iteration number is set between 40 and 70.

4.2.3 Influences of Growth Size (p, n)

Figure 4 shows how the growth size at each iteration (p positive and n negative confident examples) influences the accuracy of the proposed co-training approach. In the above experiments, we set p=n, which is considered as a balanced growth. When p differs very much from n, the growth is considered as an imbalanced growth. Balanced growth of (2, 2), (5, 5), (10, 10) and (15, 15) examples and imbalanced growth of (1, 5), (5, 1) examples are compared in the figure. We can see that the performance of the co-training approach with the balanced growth can be improved after a few iterations. And the performance of the co-training approach with large p and n will more quickly become unchanged, because the approach runs out of the limited examples in the unlabeled set more quickly. However, the performance of the co-training approaches with the two imbalanced growths is always going down quite rapidly, because the labeled unbalanced examples hurt the performance badly at each iteration.

Method	Positive			Negative			Total
	Precision	Recall	F-measure	Precision	Recall	F-measure	Accuracy
SVM(CN)	0.733	0.865	0.793	0.828	0.674	0.743	0.771
SVM(EN)	0.717	0.803	0.757	0.766	0.671	0.716	0.738
SVM(ENCN1)	0.744	0.820	0.781	0.792	0.708	0.748	0.765
SVM(ENCN2)	0.746	0.847	0.793	0.816	0.701	0.754	0.775
TSVM(CN)	0.724	0.878	0.794	0.838	0.653	0.734	0.767
TSVM(EN)	0.732	0.860	0.791	0.823	0.674	0.741	0.769
TSVM(ENCN1)	0.743	0.878	0.805	0.844	0.685	0.756	0.783
TSVM(ENCN2)	0.744	0.896	0.813	0.863	0.680	0.761	0.790
CoTrain (I=40; p=n=5)	**0.768**	**0.905**	**0.831**	**0.879**	**0.717**	**0.790**	**0.813**

Table 1. Comparison results

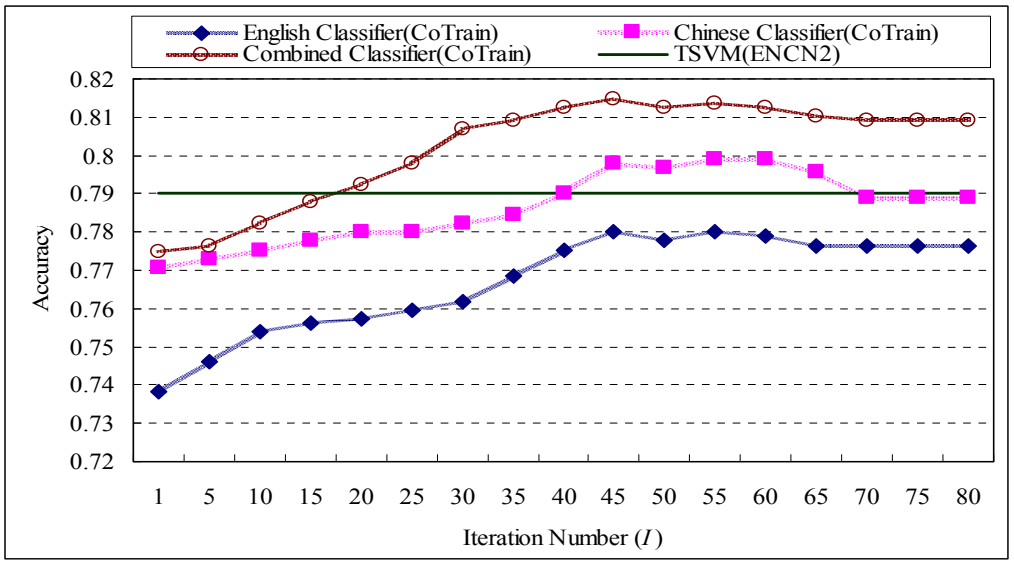


Figure 3. Accuracy vs. number of iterations for co-training (p=n=5)

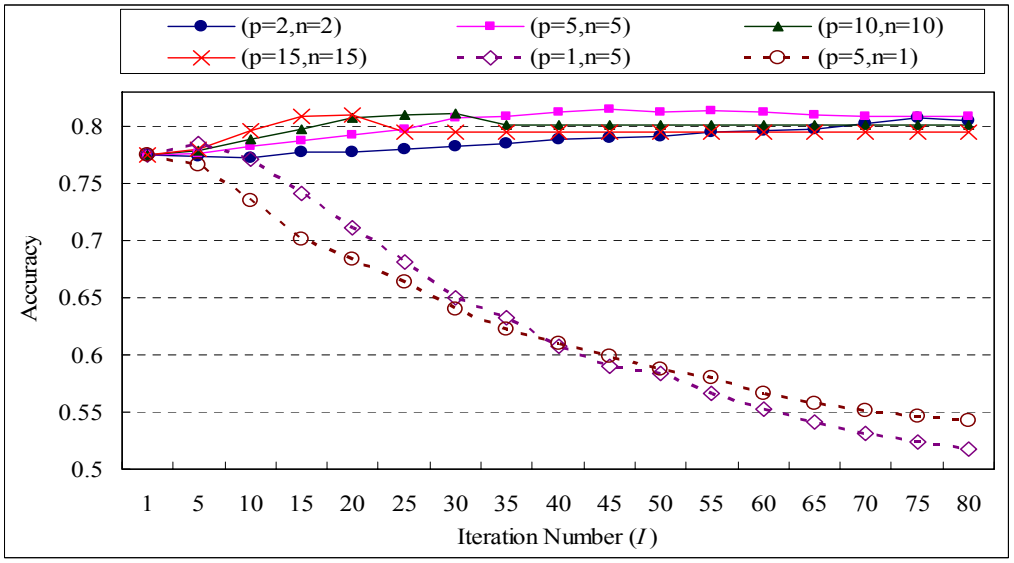


Figure 4. Accuracy vs. different (p, n) for co-training

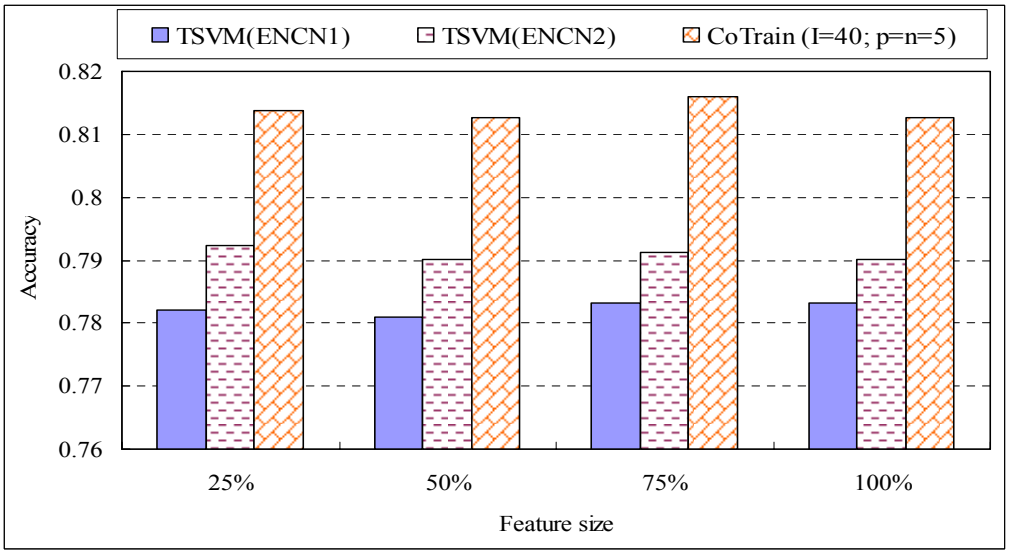


Figure 5. Influences of feature size

4.2.4 *Influences of Feature Selection*

In the above experiments, all features (unigram + bigram) are used. As mentioned earlier, feature selection techniques are widely used for dimension reduction. In this section, we further conduct experiments to investigate the influences of feature selection techniques on the classification results. We use the simple but effective document frequency (DF) for feature selection. Figures 6 show the comparison results of different feature sizes for the co-training approach and two strong baselines. The feature size is measured as the proportion of the selected features against the total features (i.e. 100%).

We can see from the figure that the feature selection technique has very slight influences on the classification accuracy of the methods. It can be seen that the co-training approach can always outperform the two baselines with different feature sizes. The results further demonstrate the effectiveness and robustness of the proposed co-training approach.

5 Conclusion and Future Work

In this paper, we propose to use the co-training approach to address the problem of cross-lingual sentiment classification. The experimental results show the effectiveness of the proposed approach.

In future work, we will improve the sentiment classification accuracy in the following two ways:

1) The smoothed co-training approach used in (Mihalcea, 2004) will be adopted for sentiment classification. The approach has the effect of "smoothing" the learning curves. During the bootstrapping process of smoothed co-training, the classifier at each iteration is replaced with a majority voting scheme applied to all classifiers constructed at previous iterations.

2) The feature distributions of the translated text and the natural text in the same language are still different due to the inaccuracy of the machine translation service. We will employ the structural correspondence learning (SCL) domain adaption algorithm used in (Blitzer et al., 2007) for linking the translated text and the natural text.

Acknowledgments

This work was supported by NSFC (60873155), RFDP (20070001059), Beijing Nova Program (2008B03), National High-tech R&D Program (2008AA01Z421) and NCET (NCET-08-0006). We also thank the anonymous reviewers for their useful comments.

References

A. Andreevskaia and S. Bergler. 2008. When specialists and generalists work together: overcoming domain dependence in sentiment tagging. In *Proceedings of ACL-08: HLT*.

C. Banea, R. Mihalcea, J. Wiebe and S. Hassan. 2008. Multilingual subjectivity analysis using machine translation. In *Proceedings of EMNLP-2008*.

N. Bel, C. H. A. Koster, and M. Villegas. 2003. Cross-lingual text categorization. In *Proceedings of ECDL-03*.

J. Blitzer, M. Dredze and F. Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: domain adaptation for sentiment classification. In *Proceedings of ACL-07*.

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with cotraining. In *Proceedings of COLT-98*.

S. Brody, R. Navigli and M. Lapata. 2006. Ensemble methods for unsupervised WSD. In *Proceedings of COLING-ACL-2006*.

S. Clark, J. R. Curran, and M. Osborne. 2003. Bootstrapping POS taggers using unlabelled data. In *Proceedings of CoNLL-2003*.

W. Dai, G.-R. Xue, Q. Yang, Y. Yu. 2007a. Transferring Naïve Bayes Classifiers for text classification. In *Proceedings of AAAI-07*.

W. Dai, G.-R. Xue, Q. Yang, Y. Yu. 2007b. Co-clustering based classification for out-of-domain documents. In *Proceedings of KDD-07*.

H. DauméIII and D. Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

A. Devitt and K. Ahmad. 2007. Sentiment polarity identification in financial news: a cohesion-based approach. In *Proceedings of ACL2007*.

T. G. Dietterich. 1997. Machine learning research: four current directions. *AI Magazine*, 18(4), 1997.

A. Gliozzo and C. Strapparava. 2005. Cross language text categorization by acquiring multilingual domain models from comparable corpora. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*.

K. Hiroshi, N. Tetsuya and W. Hideo. 2004. Deeper sentiment analysis using machine translation technology. In *Proceedings of COLING-04*.

J. Jiang and C. Zhai. 2007. A two-stage approach to domain adaptation for statistical classifiers. In *Proceedings of CIKM-07*.

T. Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of ICML-99*.

T. Joachims. 2002. Learning to classify text using support vector machines. Dissertation, Kluwer, 2002.

A. Kennedy and D. Inkpen. 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22(2):110-125.

S.-M. Kim and E. Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of COLING-04*.

S. Kiritchenko and S. Matwin. 2001. Email classification with co-training. In *Proceedings of the 2001 Conference of the Centre for Advanced Studies on Collaborative Research*.

L.-W. Ku, Y.-T. Liang and H.-H. Chen. 2006. Opinion extraction, summarization and tracking in news and blog corpora. In *Proceedings of AAAI-2006*.

J. Li and M. Sun. 2007. Experimental study on sentiment classification of Chinese review using machine learning techniques. In *Proceeding of IEEE-NLPKE-07*.

X. Ling, W. Dai, Y. Jiang, G.-R. Xue, Q. Yang, and Y. Yu. 2008. Can Chinese Web pages be classified with English data source? In *Proceedings of WWW-08*.

B. Liu, M. Hu and J. Cheng. 2005. Opinion observer: Analyzing and comparing opinions on the web. In *Proceedings of WWW-2005*.

R. McDonald, K. Hannan, T. Neylon, M. Wells and J. Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. In *Proceedings of ACL-07*.

R. Mihalcea. 2004. Co-training and self-training for word sense disambiguation. In *Proceedings of CONLL-04*.

R. Mihalcea, C. Banea and J. Wiebe. 2007. Learning multilingual subjective language via cross-lingual projections. In *Proceedings of ACL-2007*.

T. Mullen and N. Collier. 2004. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of EMNLP-04*.

V. Ng and C. Cardie. 2003. Weakly supervised natural language learning without redundant views. In *Proceedings of HLT-NAACL-03*.

K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. 2000. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39(2-3):103–134.

B. Pang, L. Lee and S. Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP-02*.

B. Pang and L. Lee. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL-04*.

J. Read. 2005. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of ACL-05*.

L. Rigutini, M. Maggini and B. Liu. 2005. An EM based training algorithm for cross-language text categorization. In *Proceedings of WI-05*.

A. Sarkar. 2001. Applying cotraining methods to statistical parsing. In *Proceedings of NAACL-2001*.

I. Titov and R. McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of ACL-08:HLT*.

B. K. Y. Tsou, R. W. M. Yuen, O. Y. Kwong, T. B. Y. La and W. L. Wong. 2005. Polarity classification of celebrity coverage in the Chinese press. In *Proceedings of International Conference on Intelligence Analysis*.

P. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of ACL-2002*.

X. Wan. 2008. Using bilingual knowledge and ensemble techniques for unsupervised Chinese sentiment analysis. In *Proceedings of EMNLP-2008*.

T. Wilson, J. Wiebe and P. Hoffmann. 2005. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In *Proceedings of HLT/EMNLP-05*.

G.-R. Xue, W. Dai, Q. Yang, Y. Yu. 2008. Topic-bridged PLSA for cross-domain text classification. In *Proceedings of SIGIR-08*.

Q. Ye, W. Shi and Y. Li. 2006. Sentiment classification for movie reviews in Chinese by improved semantic oriented approach. In *Proceedings of 39th Hawaii International Conference on System Sciences*, 2006.

A Non-negative Matrix Tri-factorization Approach to Sentiment Classification with Lexical Prior Knowledge

Tao Li **Yi Zhang**
School of Computer Science
Florida International University
{taoli,yzhan004}@cs.fiu.edu

Vikas Sindhwani
Mathematical Sciences
IBM T.J. Watson Research Center
vsindhw@us.ibm.com

Abstract

Sentiment classification refers to the task of automatically identifying whether a given piece of text expresses positive or negative opinion towards a subject at hand. The proliferation of user-generated web content such as blogs, discussion forums and online review sites has made it possible to perform large-scale mining of public opinion. Sentiment modeling is thus becoming a critical component of market intelligence and social media technologies that aim to tap into the collective wisdom of crowds. In this paper, we consider the problem of learning high-quality sentiment models with minimal manual supervision. We propose a novel approach to learn from lexical prior knowledge in the form of domain-independent sentiment-laden terms, in conjunction with domain-dependent unlabeled data and a few labeled documents. Our model is based on a constrained non-negative tri-factorization of the term-document matrix which can be implemented using simple update rules. Extensive experimental studies demonstrate the effectiveness of our approach on a variety of real-world sentiment prediction tasks.

1 Introduction

Web 2.0 platforms such as blogs, discussion forums and other such social media have now given a public voice to every consumer. Recent surveys have estimated that a massive number of internet users turn to such forums to collect recommendations for products and services, guiding their own choices and decisions by the opinions that other consumers have publically expressed. Gleaning insights by monitoring and analyzing large amounts of such user-generated data is thus becoming a key competitive differentiator for many companies. While tracking brand perceptions in traditional media is hardly a new challenge, handling the unprecedented scale of unstructured user-generated web content requires new methodologies. These methodologies are likely to be rooted in natural language processing and machine learning techniques.

Automatically classifying the sentiment expressed in a blog around selected topics of interest is a canonical machine learning task in this discussion. A standard approach would be to manually label documents with their sentiment orientation and then apply off-the-shelf text classification techniques. However, sentiment is often conveyed with subtle linguistic mechanisms such as the use of sarcasm and highly domain-specific contextual cues. This makes manual annotation of sentiment time consuming and error-prone, presenting a bottleneck in learning high quality models. Moreover, products and services of current focus, and associated community of bloggers with their idiosyncratic expressions, may rapidly evolve over time causing models to potentially lose performance and become stale. This motivates the problem of learning robust sentiment models from minimal supervision.

In their seminal work, (Pang et al., 2002) demonstrated that supervised learning significantly outperformed a competing body of work where hand-crafted dictionaries are used to assign sentiment labels based on relative frequencies of positive and negative terms. As observed by (Ng et al., 2006), most semi-automated dictionary-based approaches yield unsatisfactory lexicons, with either high coverage and low precision or vice versa. However, the treatment of such dictionaries as forms of prior knowledge that can be incorporated in machine learning models is a relatively less explored topic; even lesser so in conjunction with semi-supervised models that attempt to utilize un-

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 244–252,
Suntec, Singapore, 2-7 August 2009.

labeled data. This is the focus of the current paper.

Our models are based on a constrained non-negative tri-factorization of the term-document matrix, which can be implemented using simple update rules. Treated as a set of *labeled features*, the sentiment lexicon is incorporated as one set of constraints that enforce domain-independent prior knowledge. A second set of constraints introduce domain-specific supervision via a few document labels. Together these constraints enable learning from partial supervision along both dimensions of the term-document matrix, in what may be viewed more broadly as a framework for incorporating dual-supervision in matrix factorization models. We provide empirical comparisons with several competing methodologies on four, very different domains – blogs discussing enterprise software products, political blogs discussing US presidential candidates, amazon.com product reviews and IMDB movie reviews. Results demonstrate the effectiveness and generality of our approach.

The rest of the paper is organized as follows. We begin by discussing related work in Section 2. Section 3 gives a quick background on Non-negative Matrix Tri-factorization models. In Section 4, we present a constrained model and computational algorithm for incorporating lexical knowledge in sentiment analysis. In Section 5, we enhance this model by introducing document labels as additional constraints. Section 6 presents an empirical study on four datasets. Finally, Section 7 concludes this paper.

2 Related Work

We point the reader to a recent book (Pang and Lee, 2008) for an in-depth survey of literature on sentiment analysis. In this section, we briskly cover related work to position our contributions appropriately in the sentiment analysis and machine learning literature.

Methods focussing on the use and generation of dictionaries capturing the sentiment of words have ranged from manual approaches of developing domain-dependent lexicons (Das and Chen, 2001) to semi-automated approaches (Hu and Liu, 2004; Zhuang et al., 2006; Kim and Hovy, 2004), and even an almost fully automated approach (Turney, 2002). Most semi-automated approaches have met with limited success (Ng et al., 2006) and supervised learning models have tended to outperform dictionary-based classification schemes (Pang et al., 2002). A two-tier scheme (Pang and Lee, 2004) where sentences are first classified as *subjective* versus *objective*, and then applying the sentiment classifier on only the *subjective* sentences further improves performance. Results in these papers also suggest that using more sophisticated linguistic models, incorporating parts-of-speech and n-gram language models, do not improve over the simple unigram bag-of-words representation. In keeping with these findings, we also adopt a unigram text model. A subjectivity classification phase before our models are applied may further improve the results reported in this paper, but our focus is on driving the polarity prediction stage with minimal manual effort.

In this regard, our model brings two interrelated but distinct themes from machine learning to bear on this problem: *semi-supervised learning* and *learning from labeled features*. The goal of the former theme is to learn from few labeled examples by making use of unlabeled data, while the goal of the latter theme is to utilize weak prior knowledge about term-class affinities (e.g., the term "awful" indicates negative sentiment and therefore may be considered as a negatively labeled feature). Empirical results in this paper demonstrate that simultaneously attempting both these goals in a single model leads to improvements over models that focus on a single goal. (Goldberg and Zhu, 2006) adapt semi-supervised graph-based methods for sentiment analysis but do not incorporate lexical prior knowledge in the form of labeled features. Most work in machine learning literature on utilizing labeled features has focused on using them to generate weakly labeled examples that are then used for standard supervised learning: (Schapire et al., 2002) propose one such framework for boosting logistic regression; (Wu and Srihari, 2004) build a modified SVM and (Liu et al., 2004) use a combination of clustering and EM based methods to instantiate similar frameworks. By contrast, we incorporate lexical knowledge directly as constraints on our matrix factorization model. In recent work, Druck et al. (Druck et al., 2008) constrain the predictions of a multinomial logistic regression model on unlabeled instances in a Generalized Expectation formulation for learning from labeled features. Unlike their approach which uses only unlabeled instances, our method uses both labeled and unlabeled documents in conjunction with labeled and

unlabeled words.

The matrix tri-factorization models explored in this paper are closely related to the models proposed recently in (Li et al., 2008; Sindhwani et al., 2008). Though, their techniques for proving algorithm convergence and correctness can be readily adapted for our models, (Li et al., 2008) do not incorporate dual supervision as we do. On the other hand, while (Sindhwani et al., 2008) do incorporate dual supervision in a non-linear kernel-based setting, they do not enforce non-negativity or orthogonality – aspects of matrix factorization models that have shown benefits in prior empirical studies, see e.g., (Ding et al., 2006).

We also note the very recent work of (Sindhwani and Melville, 2008) which proposes a dual-supervision model for semi-supervised sentiment analysis. In this model, bipartite graph regularization is used to diffuse label information along both sides of the term-document matrix. Conceptually, their model implements a co-clustering assumption closely related to Singular Value Decomposition (see also (Dhillon, 2001; Zha et al., 2001) for more on this perspective) while our model is based on Non-negative Matrix Factorization. In another recent paper (Sandler et al., 2008), standard regularization models are constrained using graphs of word co-occurences. These are very recently proposed competing methodologies, and we have not been able to address empirical comparisons with them in this paper.

Finally, recent efforts have also looked at transfer learning mechanisms for sentiment analysis, e.g., see (Blitzer et al., 2007). While our focus is on single-domain learning in this paper, we note that cross-domain variants of our model can also be orthogonally developed.

3 Background

3.1 Basic Matrix Factorization Model

Our proposed models are based on non-negative matrix Tri-factorization (Ding et al., 2006). In these models, an $m \times n$ term-document matrix X is approximated by three factors that specify soft membership of terms and documents in one of k-classes:

$$X \approx FSG^T. \tag{1}$$

where F is an $m \times k$ non-negative matrix representing knowledge in the word space, i.e., i-th row of F represents the posterior probability of word i belonging to the k classes, G is an $n \times k$ non-negative matrix representing knowledge in document space, i.e., the i-th row of G represents the posterior probability of document i belonging to the k classes, and S is an $k \times k$ nonnegative matrix providing a condensed view of X.

The matrix factorization model is similar to the probabilistic latent semantic indexing (PLSI) model (Hofmann, 1999). In PLSI, X is treated as the joint distribution between words and documents by the scaling $X \rightarrow \bar{X} = X/\sum_{ij} X_{ij}$ thus $\sum_{ij} \bar{X}_{ij} = 1$). $\bar{X}$ is factorized as

$$\bar{X} \approx WSD^T, \sum_k W_{ik} = 1, \sum_k D_{jk} = 1, \sum_k S_{kk} = 1. \tag{2}$$

where X is the $m \times n$ word-document semantic matrix, $X = WSD$, W is the word class-conditional probability, and D is the document class-conditional probability and S is the class probability distribution.

PLSI provides a simultaneous solution for the word and document class conditional distribution. Our model provides simultaneous solution for clustering the rows and the columns of X. To avoid ambiguity, the orthogonality conditions

$$F^T F = I,\ G^T G = I. \tag{3}$$

can be imposed to enforce each row of F and G to possess only one nonzero entry. Approximating the term-document matrix with a tri-factorization while imposing non-negativity and orthogonality constraints gives a principled framework for simultaneously clustering the rows (words) and columns (documents) of X. In the context of co-clustering, these models return excellent empirical performance, see e.g., (Ding et al., 2006). Our goal now is to bias these models with constraints incorporating (a) labels of features (coming from a domain-independent sentiment lexicon), and (b) labels of documents for the purposes of domain-specific adaptation. These enhancements are addressed in Sections 4 and 5 respectively.

4 Incorporating Lexical Knowledge

We used a sentiment lexicon generated by the IBM India Research Labs that was developed for other text mining applications (Ramakrishnan et al., 2003). It contains 2,968 words that have been human-labeled as expressing positive or negative sentiment. In total, there are 1,267 positive (e.g. "great") and 1,701 negative (e.g., "bad") unique

terms after stemming. We eliminated terms that were ambiguous and dependent on context, such as "dear" and "fine". It should be noted, that this list was constructed without a specific domain in mind; which is further motivation for using training examples and unlabeled data to learn domain specific connotations.

Lexical knowledge in the form of the polarity of terms in this lexicon can be introduced in the matrix factorization model. By partially specifying term polarities via F, the lexicon influences the sentiment predictions G over documents.

4.1 Representing Knowledge in Word Space

Let F_0 represent prior knowledge about sentiment-laden words in the lexicon, i.e., if word i is a positive word $(F_0)_{i1} = 1$ while if it is negative $(F_0)_{i2} = 1$. Note that one may also use soft sentiment polarities though our experiments are conducted with hard assignments. This information is incorporated in the tri-factorization model via a squared loss term,

$$\min_{F,G,S} \|X - FSG^T\|^2 + \alpha \mathrm{Tr}\left[(F - F_0)^T C_1 (F - F_0)\right] \tag{4}$$

where the notation $\mathrm{Tr}(A)$ means trace of the matrix A. Here, $\alpha > 0$ is a parameter which determines the extent to which we enforce $F \approx F_0$, C_1 is a $m \times m$ diagonal matrix whose entry $(C_1)_{ii} = 1$ if the category of the i-th word is known (i.e., specified by the i-th row of F_0) and $(C_1)_{ii} = 0$ otherwise. The squared loss terms ensure that the solution for F in the otherwise unsupervised learning problem be close to the prior knowledge F_0. Note that if $C_1 = I$, then we know the class orientation of all the words and thus have a full specification of F_0, Eq.(4) is then reduced to

$$\min_{F,G,S} \|X - FSG^T\|^2 + \alpha \|F - F_0\|^2 \tag{5}$$

The above model is generic and it allows certain flexibility. For example, in some cases, our prior knowledge on F_0 is not very accurate and we use smaller α so that the final results are not dependent on F_0 very much, i.e., the results are mostly unsupervised learning results. In addition, the introduction of C_1 allows us to incorporate partial knowledge on word polarity information.

4.2 Computational Algorithm

The optimization problem in Eq.(4) can be solved using the following update rules

$$G_{jk} \leftarrow G_{jk} \frac{(X^T FS)_{jk}}{(GG^T X^T FS)_{jk}}, \tag{6}$$

$$S_{ik} \leftarrow S_{ik} \frac{(F^T XG)_{ik}}{(F^T FSG^T G)_{ik}}. \tag{7}$$

$$F_{ik} \leftarrow F_{ik} \frac{(XGS^T + \alpha C_1 F_0)_{ik}}{(FF^T XGS^T + \alpha C_1 F)_{ik}}. \tag{8}$$

The algorithm consists of an iterative procedure using the above three rules until convergence. We call this approach Matrix Factorization with Lexical Knowledge (MFLK) and outline the precise steps in the table below.

Algorithm 1 Matrix Factorization with Lexical Knowledge (MFLK)

begin

1. **Initialization:**
   Initialize $F = F_0$
   G to K-means clustering results,
   $S = (F^T F)^{-1} F^T XG(G^T G)^{-1}$.
2. **Iteration:**
   Update G: fixing F, S, updating G
   Update F: fixing S, G, updating F
   Update S: fixing F, G, updating S

end

4.3 Algorithm Correctness and Convergence

Updating F, G, S using the rules above leads to an asymptotic convergence to a local minima. This can be proved using arguments similar to (Ding et al., 2006). We outline the proof of correctness for updating F since the squared loss term that involves F is a new component in our models.

Theorem 1 *The above iterative algorithm converges.*

Theorem 2 *At convergence, the solution satisfies the Karuch, Kuhn, Tucker optimality condition, i.e., the algorithm converges correctly to a local optima.*

Theorem 1 can be proved using the standard auxiliary function approach used in (Lee and Seung, 2001).

Proof of Theorem 2. Following the theory of constrained optimization (Nocedal and Wright, 1999),

we minimize the following function

$$L(F) = ||X - FSG^T||^2 + \alpha \mathrm{Tr}\left[(F - F_0)^T C_1 (F - F0)\right]$$

Note that the gradient of L is,

$$\frac{\partial L}{\partial F} = -2XGS^T + 2FSG^T GS^T + 2\alpha C_1 (F - F_0). \tag{9}$$

The KKT complementarity condition for the non-negativity of F_{ik} gives

$$[-2XGS^T + FSG^T GS^T + 2\alpha C_1 (F - F_0)]_{ik} F_{ik} = 0. \tag{10}$$

This is the fixed point relation that local minima for F must satisfy. Given an initial guess of F, the successive update of F using Eq.(8) will converge to a local minima. At convergence, we have

$$F_{ik} = F_{ik} \frac{(XGS^T + \alpha C_1 F_0)_{ik}}{(FF^T XGS^T + \alpha C_1 F)_{ik}}.$$

which is equivalent to the KKT condition of Eq.(10). The correctness of updating rules for G in Eq.(6) and S in Eq.(7) have been proved in (Ding et al., 2006). □

Note that we do not enforce exact orthogonality in our updating rules since this often implies softer class assignments.

5 Semi-Supervised Learning With Lexical Knowledge

So far our models have made no demands on human effort, other than unsupervised collection of the term-document matrix and a one-time effort in compiling a domain-independent sentiment lexicon. We now assume that a few documents are manually labeled for the purposes of capturing some domain-specific connotations leading to a more domain-adapted model. The partial labels on documents can be described using G_0 where $(G_0)_{i1} = 1$ if the document expresses positive sentiment, and $(G_0)_{i2} = 1$ for negative sentiment. As with F_0, one can also use soft sentiment labeling for documents, though our experiments are conducted with hard assignments.

Therefore, the semi-supervised learning with lexical knowledge can be described as

$$\min_{F,G,S} \|X - FSG^T\|^2 + \alpha \mathrm{Tr}\left[(F - F_0)^T C_1 (F - F_0)\right] + \beta \mathrm{Tr}\left[(G - G_0)^T C_2 (G - G_0)\right]$$

Where $\alpha > 0, \beta > 0$ are parameters which determine the extent to which we enforce $F \approx F_0$ and $G \approx G_0$ respectively, C_1 and C_2 are diagonal matrices indicating the entries of F_0 and G_0 that correspond to labeled entities. The squared loss terms ensure that the solution for F, G, in the otherwise unsupervised learning problem, be close to the prior knowledge F_0 and G_0.

5.1 Computational Algorithm

The optimization problem in Eq.(4) can be solved using the following update rules

$$G_{jk} \leftarrow G_{jk} \frac{(X^T FS + \beta C_2 G_0)_{jk}}{(GG^T X^T FS + \beta GG^T C_2 G_0)_{jk}} \tag{11}$$

$$S_{ik} \leftarrow S_{ik} \frac{(F^T XG)_{ik}}{(F^T FSG^T G)_{ik}}. \tag{12}$$

$$F_{ik} \leftarrow F_{ik} \frac{(XGS^T + \alpha C_1 F_0)_{ik}}{(FF^T XGS^T + \alpha C_1 F)_{ik}}. \tag{13}$$

Thus the algorithm for semi-supervised learning with lexical knowledge based on our matrix factorization framework, referred as SSMFLK, consists of an iterative procedure using the above three rules until convergence. The correctness and convergence of the algorithm can also be proved using similar arguments as what we outlined earlier for MFLK in Section 4.3.

A quick word about computational complexity. The term-document matrix is typically very sparse with $z \ll nm$ non-zero entries while k is typically also much smaller than n, m. By using sparse matrix multiplications and avoiding dense intermediate matrices, the updates can be very efficiently and easily implemented. In particular, updating F, S, G each takes $O(k^2(m+n) + kz)$ time per iteration which scales linearly with the dimensions and density of the data matrix. Empirically, the number of iterations before practical convergence is usually very small (less than 100). Thus, computationally our approach scales to large datasets even though our experiments are run on relatively small-sized datasets.

6 Experiments

6.1 Datasets Description

Four different datasets are used in our experiments.

Movies Reviews: This is a popular dataset in sentiment analysis literature (Pang et al., 2002). It consists of 1000 positive and 1000 negative movie reviews drawn from the IMDB archive of the rec.arts.movies.reviews newsgroups.

Lotus blogs: The data set is targeted at detecting sentiment around enterprise software, specifically pertaining to the IBM Lotus brand (Sindhwani and Melville, 2008). An unlabeled set of blog posts was created by randomly sampling 2000 posts from a universe of 14,258 blogs that discuss issues relevant to Lotus software. In addition to this unlabeled set, 145 posts were chosen for manual labeling. These posts came from 14 individual blogs, 4 of which are actively posting negative content on the brand, with the rest tending to write more positive or neutral posts. The data was collected by downloading the latest posts from each blogger's RSS feeds, or accessing the blog's archives. Manual labeling resulted in 34 positive and 111 negative examples. **Political candidate blogs**: For our second blog domain, we used data gathered from 16,742 political blogs, which contain over 500,000 posts. As with the Lotus dataset, an unlabeled set was created by randomly sampling 2000 posts. 107 posts were chosen for labeling. A post was labeled as having positive or negative sentiment about a specific candidate (Barack Obama or Hillary Clinton) if it explicitly mentioned the candidate in positive or negative terms. This resulted in 49 positively and 58 negatively labeled posts. **Amazon Reviews**: The dataset contains product reviews taken from Amazon.com from 4 product types: Kitchen, Books, DVDs, and Electronics (Blitzer et al., 2007). The dataset contains about 4000 positive reviews and 4000 negative reviews and can be obtained from `http://www.cis.upenn.edu/~mdredze/datasets/sentiment/`.

For all datasets, we picked 5000 words with highest document-frequency to generate the vocabulary. Stopwords were removed and a normalized term-frequency representation was used. Genuinely unlabeled posts for Political and Lotus were used for semi-supervised learning experiments in section 6.3; they were not used in section 6.2 on the effect of lexical prior knowledge. In the experiments, we set α, the parameter determining the extent to which to enforce the feature labels, to be 1/2, and β, the corresponding parameter for enforcing document labels, to be 1.

6.2 Sentiment Analysis with Lexical Knowledge

Of course, one can remove all burden on human effort by simply using unsupervised techniques. Our interest in the first set of experiments is to explore the benefits of incorporating a sentiment lexicon over unsupervised approaches. Does a one-time effort in compiling a domain-independent dictionary and using it for different sentiment tasks pay off in comparison to simply using unsupervised methods? In our case, matrix tri-factorization and other co-clustering methods form the obvious unsupervised baseline for comparison and so we start by comparing our method (MFLK) with the following methods:

- Four document clustering methods: K-means, Tri-Factor Nonnegative Matrix Factorization (TNMF) (Ding et al., 2006), Information-Theoretic Co-clustering (ITCC) (Dhillon et al., 2003), and Euclidean Co-clustering algorithm (ECC) (Cho et al., 2004). These methods do not make use of the sentiment lexicon.

- Feature Centroid (FC): This is a simple dictionary-based baseline method. Recall that each word can be expressed as a "bag-of-documents" vector. In this approach, we compute the centroids of these vectors, one corresponding to positive words and another corresponding to negative words. This yields a two-dimensional representation for documents, on which we then perform K-means clustering.

Performance Comparison Figure 1 shows the experimental results on four datasets using accuracy as the performance measure. The results are obtained by averaging 20 runs. It can be observed that our MFLK method can effectively utilize the lexical knowledge to improve the quality of sentiment prediction.

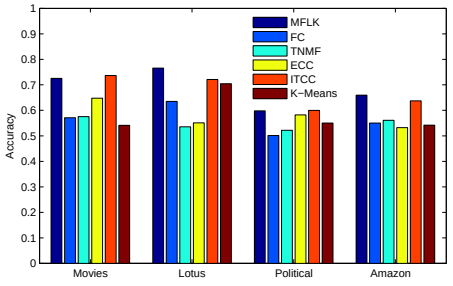


Figure 1: Accuracy results on four datasets

Size of Sentiment Lexicon We also investigate the effects of the size of the sentiment lexicon on the performance of our model. Figure 2 shows results with random subsets of the lexicon of increasing size. We observe that generally the performance increases as more and more lexical supervision is provided.

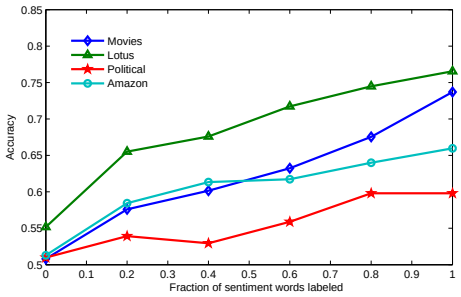


Figure 2: MFLK accuracy as size of sentiment lexicon (i.e., number of words in the lexicon) increases on the four datasets

Robustness to Vocabulary Size High dimensionality and noise can have profound impact on the comparative performance of clustering and semi-supervised learning algorithms. We simulate scenarios with different vocabulary sizes by selecting words based on information gain. It should, however, be kept in mind that in a truely unsupervised setting document labels are unavailable and therefore information gain cannot be practically computed. Figure 3 and Figure 4 show results for Lotus and Amazon datasets respectively and are representative of performance on other datasets. MLFK tends to retain its position as the best performing method even at different vocabulary sizes. ITCC performance is also noteworthy given that it is a completely unsupervised method.

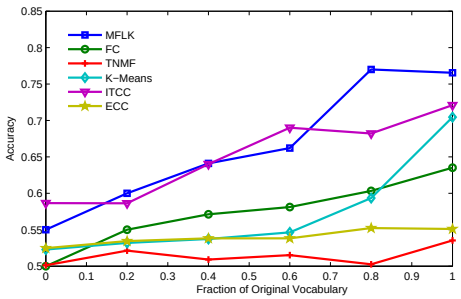


Figure 3: Accuracy results on Lotus dataset with increasing vocabulary size

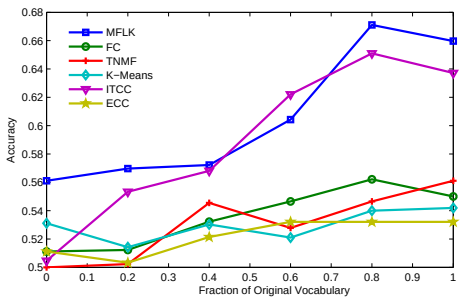


Figure 4: Accuracy results on Amazon dataset with increasing vocabulary size

6.3 Sentiment Analysis with Dual Supervision

We now assume that together with labeled features from the sentiment lexicon, we also have access to a few labeled documents. The natural question is whether the presence of lexical constraints leads to better semi-supervised models. In this section, we compare our method (SSMFLK) with the following three semi-supervised approaches: (1) The algorithm proposed in (Zhou et al., 2003) which conducts semi-supervised learning with local and global consistency (Consistency Method); (2) Zhu et al.'s harmonic Gaussian field method coupled with the Class Mass Normalization (Harmonic-CMN) (Zhu et al., 2003); and (3) Green's function learning algorithm (Green's Function) proposed in (Ding et al., 2007).

We also compare the results of SSMFLK with those of two supervised classification methods: Support Vector Machine (SVM) and Naive Bayes. Both of these methods have been widely used in sentiment analysis. In particular, the use of SVMs in (Pang et al., 2002) initially sparked interest in using machine learning methods for sentiment classification. Note that none of these competing methods utilizes lexical knowledge.

The results are presented in Figure 5, Figure 6, Figure 7, and Figure 8. We note that our SSMFLK method either outperforms all other methods over the entire range of number of labeled documents (Movies, Political), or ultimately outpaces other methods (Lotus, Amazon) as a few document labels come in.

Learning Domain-Specific Connotations In our first set of experiments, we incorporated the sentiment lexicon in our models and learnt the sentiment orientation of words and documents via F, G factors respectively. In the second set of

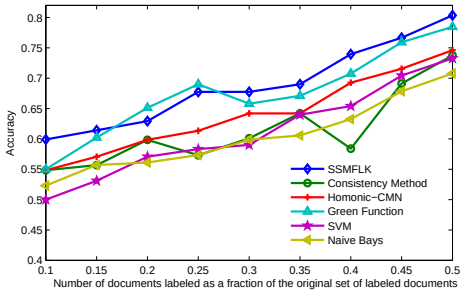


Figure 5: Accuracy results with increasing number of labeled documents on Movies dataset

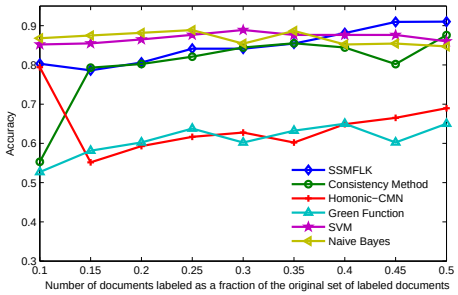


Figure 6: Accuracy results with increasing number of labeled documents on Lotus dataset

experiments, we additionally introduced labeled documents for domain-specific adjustments. Between these experiments, we can now look for words that switch sentiment polarity. These words are interesting because their domain-specific connotation differs from their lexical orientation. For amazon reviews, the following words switched polarity from positive to negative: *fan, important, learning, cons, fast, feature, happy, memory, portable, simple, small, work* while the following words switched polarity from negative to positive: *address, finish, lack, mean, budget, rent, throw*. Note that words like *fan, memory* probably refer to product or product components (i.e., computer fan and memory) in the amazon review context but have a very different connotation say in the context of movie reviews where they probably refer to movie fanfare and memorable performances. We were surprised to see *happy* switch polarity! Two examples of its negative-sentiment usage are: *I ended up buying a Samsung and I couldn't be more happy* and *BORING, not one single exciting thing about this book. I was happy when my lunch break ended so I could go back to work and stop reading.*

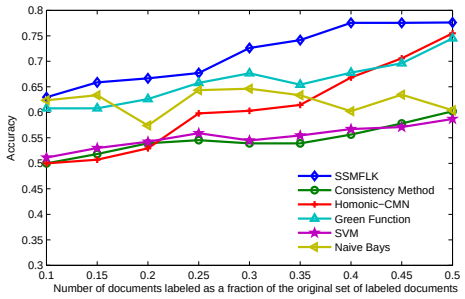


Figure 7: Accuracy results with increasing number of labeled documents on Political dataset

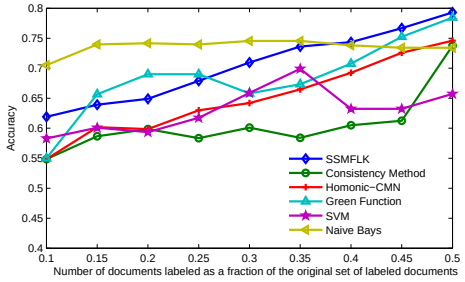


Figure 8: Accuracy results with increasing number of labeled documents on Amazon dataset

7 Conclusion

The primary contribution of this paper is to propose and benchmark new methodologies for sentiment analysis. Non-negative Matrix Factorizations constitute a rich body of algorithms that have found applicability in a variety of machine learning applications: from recommender systems to document clustering. We have shown how to build effective sentiment models by appropriately constraining the factors using lexical prior knowledge and document annotations. To more effectively utilize unlabeled data and induce domain-specific adaptation of our models, several extensions are possible: facilitating learning from related domains, incorporating hyperlinks between documents, incorporating synonyms or co-occurences between words etc. As a topic of vigorous current activity, there are several very recently proposed competing methodologies for sentiment analysis that we would like to benchmark against. These are topics for future work.

Acknowledgement: The work of T. Li is partially supported by NSF grants DMS-0844513 and CCF-0830659. We would also like to thank Prem Melville and Richard Lawrence for their support.

References

J. Blitzer, M. Dredze, and F. Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of ACL*, pages 440–447.

H. Cho, I. Dhillon, Y. Guan, and S. Sra. 2004. Minimum sum squared residue co-clustering of gene expression data. In *Proceedings of The 4th SIAM Data Mining Conference*, pages 22–24, April.

S. Das and M. Chen. 2001. Yahoo! for amazon: Extracting market sentiment from stock message boards. In *Proceedings of the 8th Asia Pacifi c Finance Association (APFA)*.

I. S. Dhillon, S. Mallela, and D. S. Modha. 2003. Information-theoretical co-clustering. In *Proceedings of ACM SIGKDD*, pages 89–98.

I. S. Dhillon. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of ACM SIGKDD*.

C. Ding, T. Li, W. Peng, and H. Park. 2006. Orthogonal nonnegative matrix tri-factorizations for clustering. In *Proceedings of ACM SIGKDD*, pages 126–135.

C. Ding, R. Jin, T. Li, and H.D. Simon. 2007. A learning framework using green's function and kernel regularization with application to recommender system. In *Proceedings of ACM SIGKDD*, pages 260–269.

G. Druck, G. Mann, and A. McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *SIGIR*.

A. Goldberg and X. Zhu. 2006. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *HLT-NAACL 2006: Workshop on Textgraphs*.

T. Hofmann. 1999. Probabilistic latent semantic indexing. *Proceeding of SIGIR*, pages 50–57.

M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. In *KDD*, pages 168–177.

S.-M. Kim and E. Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of International Conference on Computational Linguistics*.

D.D. Lee and H.S. Seung. 2001. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13*.

T. Li, C. Ding, Y. Zhang, and B. Shao. 2008. Knowledge transformation from word space to document space. In *Proceedings of SIGIR*, pages 187–194.

B. Liu, X. Li, W.S. Lee, and P. Yu. 2004. Text classification by labeling words. In *AAAI*.

V. Ng, S. Dasgupta, and S. M. Niaz Arifin. 2006. Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. In *COLING & ACL*.

J. Nocedal and S.J. Wright. 1999. *Numerical Optimization*. Springer-Verlag.

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

B. Pang and L. Lee. 2008. *Opinion mining and sentiment analysis*. Foundations and Trends in Information Retrieval: Vol. 2: No 12, pp 1-135 http://www.cs.cornell.edu/home/llee/opinion-mining-sentiment-analysis-survey.html.

B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *EMNLP*.

G. Ramakrishnan, A. Jadhav, A. Joshi, S. Chakrabarti, and P. Bhattacharyya. 2003. Question answering via bayesian inference on lexical relations. In *ACL*, pages 1–10.

T. Sandler, J. Blitzer, P. Talukdar, and L. Ungar. 2008. Regularized learning with networks of features. In *NIPS*.

R.E. Schapire, M. Rochery, M.G. Rahim, and N. Gupta. 2002. Incorporating prior knowledge into boosting. In *ICML*.

V. Sindhwani and P. Melville. 2008. Document-word co-regularization for semi-supervised sentiment analysis. In *Proceedings of IEEE ICDM*.

V. Sindhwani, J. Hu, and A. Mojsilovic. 2008. Regularized co-clustering with dual supervision. In *Proceedings of NIPS*.

P. Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 417–424.

X. Wu and R. Srihari. 2004. Incorporating prior knowledge with weighted margin support vector machines. In *KDD*.

H. Zha, X. He, C. Ding, M. Gu, and H.D. Simon. 2001. Bipartite graph partitioning and data clustering. *Proceedings of ACM CIKM*.

D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Scholkopf. 2003. Learning with local and global consistency. In *Proceedings of NIPS*.

X. Zhu, Z. Ghahramani, and J. Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of ICML*.

L. Zhuang, F. Jing, and X. Zhu. 2006. Movie review mining and summarization. In *CIKM*, pages 43–50.

Discovering the Discriminative Views: Measuring Term Weights for Sentiment Analysis

Jungi Kim, Jin-Ji Li and Jong-Hyeok Lee
Division of Electrical and Computer Engineering
Pohang University of Science and Technology, Pohang, Republic of Korea
{yangpa,ljj,jhlee}@postech.ac.kr

Abstract

This paper describes an approach to utilizing term weights for sentiment analysis tasks and shows how various term weighting schemes improve the performance of sentiment analysis systems. Previously, sentiment analysis was mostly studied under data-driven and lexicon-based frameworks. Such work generally exploits textual features for fact-based analysis tasks or lexical indicators from a sentiment lexicon. We propose to model term weighting into a sentiment analysis system utilizing collection statistics, contextual and topic-related characteristics as well as opinion-related properties. Experiments carried out on various datasets show that our approach effectively improves previous methods.

1 Introduction

With the explosion in the amount of commentaries on current issues and personal views expressed in weblogs on the Internet, the field of studying how to analyze such remarks and sentiments has been increasing as well. The field of opinion mining and sentiment analysis involves extracting opinionated pieces of text, determining the polarities and strengths, and extracting holders and targets of the opinions.

Much research has focused on creating testbeds for sentiment analysis tasks. Most notable and widely used are Multi-Perspective Question Answering (MPQA) and Movie-review datasets. MPQA is a collection of newspaper articles annotated with opinions and private states at the subsentence level (Wiebe et al., 2003). Movie-review dataset consists of positive and negative reviews from the Internet Movie Database (IMDb) archive (Pang et al., 2002).

Evaluation workshops such as TREC and NTCIR have recently joined in this new trend of research and organized a number of successful meetings. At the TREC Blog Track meetings, researchers have dealt with the problem of retrieving topically-relevant blog posts and identifying documents with opinionated contents (Ounis et al., 2008). NTCIR Multilingual Opinion Analysis Task (MOAT) shared a similar mission, where participants are provided with a number of topics and a set of relevant newspaper articles for each topic, and asked to extract opinion-related properties from enclosed sentences (Seki et al., 2008).

Previous studies for sentiment analysis belong to either the data-driven approach where an annotated corpus is used to train a machine learning (ML) classifier, or to the lexicon-based approach where a pre-compiled list of sentiment terms is utilized to build a sentiment score function.

This paper introduces an approach to the sentiment analysis tasks with an emphasis on how to represent and evaluate the weights of sentiment terms. We propose a number of characteristics of good sentiment terms from the perspectives of informativeness, prominence, topic–relevance, and semantic aspects using collection statistics, contextual information, semantic associations as well as opinion–related properties of terms. These term weighting features constitute the sentiment analysis model in our opinion retrieval system. We test our opinion retrieval system with TREC and NTCIR datasets to validate the effectiveness of our term weighting features. We also verify the effectiveness of the statistical features used in data-driven approaches by evaluating an ML classifier with labeled corpora.

2 Related Work

Representing text with salient features is an important part of a text processing task, and there exists many works that explore various features for

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 253–261,
Suntec, Singapore, 2-7 August 2009.

text analysis systems (Sebastiani, 2002; Forman, 2003). Sentiment analysis task have also been using various lexical, syntactic, and statistical features (Pang and Lee, 2008). Pang et al. (2002) employed n-gram and POS features for ML methods to classify movie-review data. Also, syntactic features such as the dependency relationship of words and subtrees have been shown to effectively improve the performances of sentiment analysis (Kudo and Matsumoto, 2004; Gamon, 2004; Matsumoto et al., 2005; Ng et al., 2006).

While these features are usually employed by data-driven approaches, there are unsupervised approaches for sentiment analysis that make use of a set of terms that are semantically oriented toward expressing subjective statements (Yu and Hatzivassiloglou, 2003). Accordingly, much research has focused on recognizing terms' semantic orientations and strength, and compiling sentiment lexicons (Hatzivassiloglou and Mckeown, 1997; Turney and Littman, 2003; Kamps et al., 2004; Whitelaw et al., 2005; Esuli and Sebastiani, 2006).

Interestingly, there are conflicting conclusions about the usefulness of the statistical features in sentiment analysis tasks (Pang and Lee, 2008). Pang et al. (2002) presents empirical results indicating that using term presence over term frequency is more effective in a data-driven sentiment classification task. Such a finding suggests that sentiment analysis may exploit different types of characteristics from the topical tasks, that, unlike fact-based text analysis tasks, repetition of terms does not imply a significance on the overall sentiment. On the other hand, Wiebe et al. (2004) have noted that hapax legomena (terms that only appear once in a collection of texts) are good signs for detecting subjectivity. Other works have also exploited rarely occurring terms for sentiment analysis tasks (Dave et al., 2003; Yang et al., 2006).

The opinion retrieval task is a relatively recent issue that draws both the attention of IR and NLP communities. Its task is to find relevant documents that also contain sentiments about a given topic. Generally, the opinion retrieval task has been approached as a two–stage task: first, retrieving topically relevant documents, then reranking the documents by the opinion scores (Ounis et al., 2006). This approach is also appropriate for evaluation systems such as NTCIR MOAT that assumes that the set of topically relevant documents are already known in advance. On the other hand, there are also some interesting works on modeling the topic and sentiment of documents in a unified way (Mei et al., 2007; Zhang and Ye, 2008).

3 Term Weighting and Sentiment Analysis

In this section, we describe the characteristics of terms that are useful in sentiment analysis, and present our sentiment analysis model as part of an opinion retrieval system and an ML sentiment classifier.

3.1 Characteristics of Good Sentiment Terms

This section examines the qualities of useful terms for sentiment analysis tasks and corresponding features. For the sake of organization, we categorize the sources of features into either global or local knowledge, and either topic-independent or topic-dependent knowledge.

Topic-independently speaking, a good sentiment term is *discriminative* and *prominent*, such that the appearance of the term imposes greater influence on the judgment of the analysis system. The rare occurrence of terms in document collections has been regarded as a very important feature in IR methods, and effective IR models of today, either explicitly or implicitly, accommodate this feature as an Inverse Document Frequency (IDF) heuristic (Fang et al., 2004). Similarly, prominence of a term is recognized by the frequency of the term in its local context, formulated as Term Frequency (TF) in IR.

If a topic of the text is known, terms that are relevant and descriptive of the subject should be regarded to be more useful than topically-irrelevant and extraneous terms. One way of measuring this is using associations between the query and terms. Statistical measures of associations between terms include estimations by the co-occurrence in the whole collection, such as Point-wise Mutual Information (PMI) and Latent Semantic Analysis (LSA). Another method is to use proximal information of the query and the word, using syntactic structure such as dependency relations of words that provide the graphical representation of the text (Mullen and Collier, 2004). The minimum spans of words in such graph may represent their associations in the text. Also, the distance between words in the local context or in the thesaurus-like dictionaries such as WordNet may be approximated as such measure.

3.2 Opinion Retrieval Model

The goal of an opinion retrieval system is to find a set of opinionated documents that are relevant to a given topic. We decompose the opinion retrieval system into two tasks: the topical retrieval task and the sentiment analysis task. This two-stage approach for opinion retrieval has been taken by many systems and has been shown to perform well (Ounis et al., 2006). The topic and the sentiment aspects of the opinion retrieval task are modeled separately, and linearly combined together to produce a list of topically-relevant and opinionated documents as below.

$$Score_{OpRet}(D,Q) = \lambda \cdot Score_{rel}(D,Q) + (1-\lambda) \cdot Score_{op}(D,Q)$$

The topic-relevance model $Score_{rel}$ may be substituted by any IR system that retrieves relevant documents for the query Q. For tasks such as NTCIR MOAT, relevant documents are already known in advance and it becomes unnecessary to estimate the relevance degree of the documents. We focus on modeling the sentiment aspect of the opinion retrieval task, assuming that the topic-relevance of documents is provided in some way.

To assign documents with sentiment degrees, we estimate the probability of a document D to generate a query Q and to possess opinions as indicated by a random variable Op.[1] Assuming uniform prior probabilities of documents D, query Q, and Op, and conditional independence between Q and Op, the opinion score function reduces to estimating the generative probability of Q and Op given D.

$$Score_{op}(D,Q) \equiv p(D \mid Op, Q) \propto p(Op, Q \mid D)$$

If we regard that the document D is represented as a bag of words and that the words are uniformly distributed, then

$$\begin{aligned} p(Op, Q \mid D) &= \sum_{w \in D} p(Op, Q \mid w) \cdot p(w \mid D) \\ &= \sum_{w \in D} p(Op \mid w) \cdot p(Q \mid w) \cdot p(w \mid D) \end{aligned} \quad (1)$$

Equation 1 consists of three factors: the probability of a word to be opinionated ($P(Op|w)$), the likelihood of a query given a word ($P(Q|w)$), and the probability of a document generating a word ($P(w|D)$). Intuitively speaking, the probability of a document embodying topically related opinion is estimated by accumulating the probabilities of all words from the document to have sentiment meanings and associations with the given query.

In the following sections, we assess the three factors of the sentiment models from the perspectives of term weighting.

3.2.1 Word Sentiment Model

Modeling the sentiment of a word has been a popular approach in sentiment analysis. There are many publicly available lexicon resources. The size, format, specificity, and reliability differ in all these lexicons. For example, lexicon sizes range from a few hundred to several hundred thousand. Some lexicons assign real number scores to indicate sentiment orientations and strengths (i.e. probabilities of having positive and negative sentiments) (Esuli and Sebastiani, 2006) while other lexicons assign discrete classes (weak/strong, positive/negative) (Wilson et al., 2005). There are manually compiled lexicons (Stone et al., 1966) while some are created semi-automatically by expanding a set of seed terms (Esuli and Sebastiani, 2006).

The goal of this paper is not to create or choose an appropriate sentiment lexicon, but rather it is to discover useful term features other than the sentiment properties. For this reason, one sentiment lexicon, namely SentiWordNet, is utilized throughout the whole experiment.

SentiWordNet is an automatically generated sentiment lexicon using a semi-supervised method (Esuli and Sebastiani, 2006). It consists of WordNet synsets, where each synset is assigned three probability scores that add up to 1: positive, negative, and objective.

These scores are assigned at sense level (synsets in WordNet), and we use the following equations to assess the sentiment scores at the word level.

$$\begin{aligned} p(Pos \mid w) &= \max_{s \in synset(w)} SWN_{Pos}(s) \\ p(Neg \mid w) &= \max_{s \in synset(w)} SWN_{Neg}(s) \\ p(Op \mid w) &= \max\left(p(Pos \mid w), p(Neg \mid w)\right) \end{aligned}$$

where $synset(w)$ is the set of synsets of w and $SWN_{Pos}(s)$, $SWN_{Neg}(s)$ are positive and negative scores of a synset in SentiWordNet. We assess the subjective score of a word as the maximum value of the positive and the negative scores, because a word has either a positive or a negative sentiment in a given context.

The word sentiment model can also make use of other types of sentiment lexicons. The sub-

[1] Throughout this paper, Op indicates $Op = 1$.

jectivity lexicon used in OpinionFinder[2] is compiled from several manually and automatically built resources. Each word in the lexicon is tagged with the strength (*strong*/*weak*) and polarity (*Positive*/*Negative*/*Neutral*). The word sentiment can be modeled as below.

$$P(Pos|w) = \begin{cases} 1.0 & \text{if w is } \textit{Positive and Strong} \\ 0.5 & \text{if w is } \textit{Positive and Weak} \\ 0.0 & \text{otherwise} \end{cases}$$

$$P(Op \mid w) = \max\left(p(Pos \mid w), p(Neg \mid w)\right)$$

3.2.2 Topic Association Model

If a topic is given in the sentiment analysis, terms that are closely associated with the topic should be assigned heavy weighting. For example, sentiment words such as *scary* and *funny* are more likely to be associated with topic words such as *book* and *movie* than *grocery* or *refrigerator*.

In the topic association model, $p(Q \mid w)$ is estimated from the associations between the word w and a set of query terms Q.

$$p(Q \mid w) = \frac{\sum_{q \in Q} \textit{Asc-Score}(q, w)}{\mid Q \mid} \propto \sum_{q \in Q} \textit{Asc-Score}(q, w)$$

$\textit{Asc-Score}(q, w)$ is the association score between q and w, and $\mid Q \mid$ is the number of query words.

To measure associations between words, we employ statistical approaches using document collections such as LSA and PMI, and local proximity features using the distance in dependency trees or texts.

Latent Semantic Analysis (LSA) (Landauer and Dumais, 1997) creates a semantic space from a collection of documents to measure the semantic relatedness of words. Point-wise Mutual Information (PMI) is a measure of associations used in information theory, where the association between two words is evaluated with the joint and individual distributions of the two words. PMI-IR (Turney, 2001) uses an IR system and its search operators to estimate the probabilities of two terms and their conditional probabilities. Equations for association scores using LSA and PMI are given below.

$$\textit{Asc-Score}_{LSA}(w_1, w_2) = \frac{1 + LSA(w_1, w_2)}{2}$$

$$\textit{Asc-Score}_{PMI}(w_1, w_2) = \frac{1 + \textit{PMI-IR}(w_1, w_2)}{2}$$

[2]http://www.cs.pitt.edu/mpqa/

For the experimental purpose, we used publicly available online demonstrations for LSA and PMI. For LSA, we used the online demonstration mode from the Latent Semantic Analysis page from the University of Colorado at Boulder.[3] For PMI, we used the online API provided by the CogWorks Lab at the Rensselaer Polytechnic Institute.[4]

Word associations between two terms may also be evaluated in the local context where the terms appear together. One way of measuring the proximity of terms is using the syntactic structures. Given the dependency tree of the text, we model the association between two terms as below.

$$\textit{Asc-Score}_{DTP}(w_1, w_2) = \begin{cases} 1.0 & \text{min. span in dep. tree} \leq D_{syn} \\ 0.5 & \text{otherwise} \end{cases}$$

where, D_{syn} is arbitrarily set to 3.

Another way is to use co-occurrence statistics as below.

$$\textit{Asc-Score}_{WP}(w_1, w_2) = \begin{cases} 1.0 & \text{if distance between} w_1 \text{and} w_2 \leq K \\ 0.5 & \text{otherwise} \end{cases}$$

where K is the maximum window size for the co-occurrence and is arbitrarily set to 3 in our experiments.

The statistical approaches may suffer from data sparseness problems especially for named entity terms used in the query, and the proximal clues cannot sufficiently cover all term–query associations. To avoid assigning zero probabilities, our topic association models assign 0.5 to word pairs with no association and 1.0 to words with perfect association.

Note that proximal features using co-occurrence and dependency relationships were used in previous work. For opinion retrieval tasks, Yang et al. (2006) and Zhang and Ye (2008) used the co-occurrence of a query word and a sentiment word within a certain window size. Mullen and Collier (2004) manually annotated named entities in their dataset (i.e. title of the record and name of the artist for music record reviews), and utilized presence and position features in their ML approach.

3.2.3 Word Generation Model

Our word generation model $p(w \mid d)$ evaluates the prominence and the discriminativeness of a word

[3]http://lsa.colorado.edu/, default parameter settings for the semantic space (TASA, 1st year college level) and number of factors (300).

[4]http://cwl-projects.cogsci.rpi.edu/msr/, PMI-IR with the Google Search Engine.

w in a document d. These issues correspond to the core issues of traditional IR tasks. IR models, such as Vector Space (VS), probabilistic models such as BM25, and Language Modeling (LM), albeit in different forms of approach and measure, employ heuristics and formal modeling approaches to effectively evaluate the relevance of a term to a document (Fang et al., 2004). Therefore, we estimate the word generation model with popular IR models' the relevance scores of a document d given w as a query.[5]

$$p(w \mid d) \equiv IR\text{-}SCORE(w, d)$$

In our experiments, we use the Vector Space model with Pivoted Normalization (VS), Probabilistic model (BM25), and Language modeling with Dirichlet Smoothing (LM).

$$VSPN(w,d) = \frac{1 + ln(1 + ln(c(w,d)))}{(1-s) + s \cdot \frac{\mid d \mid}{avgdl}} \cdot ln\frac{N+1}{df(w)}$$

$$BM25(w,d) = ln\frac{N - df(w) + 0.5}{df(w) + 0.5} \cdot \frac{(k_1+1) \cdot c(w,d)}{k_1\left((1-b) + b\frac{|d|}{avgdl}\right) + c(w,d)}$$

$$LMDI(w,d) = ln\left(1 + \frac{c(w,d)}{\mu \cdot c(w,C)}\right) + ln\frac{\mu}{\mid d \mid + \mu}$$

$c(w,d)$ is the frequency of w in d, $\mid d \mid$ is the number of unique terms in d, $avgdl$ is the average $\mid d \mid$ of all documents, N is the number of documents in the collection, $df(w)$ is the number of documents with w, C is the entire collection, and k_1 and b are constants 2.0 and 0.75.

3.3 Data-driven Approach

To verify the effectiveness of our term weighting schemes in experimental settings of the data-driven approach, we carry out a set of simple experiments with ML classifiers. Specifically, we explore the statistical term weighting features of the word generation model with Support Vector machine (SVM), faithfully reproducing previous work as closely as possible (Pang et al., 2002).

Each instance of train and test data is represented as a vector of features. We test various combinations of the term weighting schemes listed below.

- PRESENCE: binary indicator for the presence of a term
- TF: term frequency
- VS.TF: normalized tf as in VS
- BM25.TF: normalized tf as in BM25
- IDF: inverse document frequency
- VS.IDF: normalized idf as in VS
- BM25.IDF: normalized idf as in BM25

[5]With proper assumptions and derivations, $p(w \mid d)$ can be derived to language modeling approaches. Refer to (Zhai and Lafferty, 2004).

4 Experiment

Our experiments consist of an opinion retrieval task and a sentiment classification task. We use MPQA and movie-review corpora in our experiments with an ML classifier. For the opinion retrieval task, we use the two datasets used by TREC blog track and NTCIR MOAT evaluation workshops.

The opinion retrieval task at TREC Blog Track consists of three subtasks: topic retrieval, opinion retrieval, and polarity retrieval. Opinion and polarity retrieval subtasks use the relevant documents retrieved at the topic retrieval stage. On the other hand, the NTCIR MOAT task aims to find opinionated sentences given a set of documents that are already hand-assessed to be relevant to the topic.

4.1 Opinion Retieval Task – TREC Blog Track

4.1.1 Experimental Setting

TREC Blog Track uses the TREC Blog06 corpus (Macdonald and Ounis, 2006). It is a collection of RSS feeds (38.6 GB), permalink documents (88.8GB), and homepages (28.8GB) crawled on the Internet over an eleven week period from December 2005 to February 2006.

Non-relevant content of blog posts such as HTML tags, advertisement, site description, and menu are removed with an effective internal spam removal algorithm (Nam et al., 2009). While our sentiment analysis model uses the entire relevant portion of the blog posts, further stopword removal and stemming is done for the blog retrieval system.

For the relevance retrieval model, we faithfully reproduce the passage-based language model with pseudo-relevance feedback (Lee et al., 2008).

We use in total 100 topics from TREC 2007 and 2008 blog opinion retrieval tasks (07:901-950 and 08:1001-1050). We use the topics from Blog 07 to optimize the parameter for linearly combining the retrieval and opinion models, and use Blog 08 topics as our test data. Topics are extracted only from the Title field, using the Porter stemmer and a stopword list.

Table 1: Performance of opinion retrieval models using Blog 08 topics. The linear combination parameter λ is optimized on Blog 07 topics. † indicates statistical significance at the 1% level over the baseline.

Model	MAP	R-prec	P@10
TOPIC REL.	0.4052	0.4366	0.6440
BASELINE	0.4141	0.4534	0.6440
VS	0.4196	0.4542	0.6600
BM25	0.4235†	**0.4579**	0.6600
LM	0.4158	0.4520	0.6560
PMI	0.4177	0.4538	0.6620
LSA	0.4155	0.4526	0.6480
WP	0.4165	0.4533	**0.6640**
BM25·PMI	0.4238†	0.4575	0.6600
BM25·LSA	0.4237†	0.4578	0.6600
BM25·WP	0.4237†	**0.4579**	0.6600
BM25·PMI·WP	**0.4242**†	0.4574	0.6620
BM25·LSA·WP	0.4238†	0.4576	0.6580

4.1.2 Experimental Result

Retrieval performances using different combinations of term weighting features are presented in Table 1. Using only the word sentiment model is set as our baseline.

First, each feature of the word generation and topic association models are tested; all features of the models improve over the baseline. We observe that the features of our word generation model is more effective than those of the topic association model. Among the features of the word generation model, the most improvement was achieved with $BM25$, improving the MAP by 2.27%.

Features of the topic association model show only moderate improvements over the baseline. We observe that these features generally improve P@10 performance, indicating that they increase the accuracy of the sentiment analysis system. PMI out-performed LSA for all evaluation measures. Among the topic association models, PMI performs the best in MAP and R-prec, while WP achieved the biggest improvement in P@10.

Since BM25 performs the best among the word generation models, its combination with other features was investigated. Combinations of BM25 with the topic association models all improve the performance of the baseline and BM25. This demonstrates that the word generation model and the topic association model are complementary to each other.

The best MAP was achieved with BM25, PMI, and WP (+2.44% over the baseline). We observe that PMI and WP also complement each other.

4.2 Sentiment Analysis Task – NTCIR MOAT

4.2.1 Experimental Setting

Another set of experiments for our opinion analysis model was carried out on the NTCIR-7 MOAT English corpus. The English opinion corpus for NTCIR MOAT consists of newspaper articles from the Mainichi Daily News, Korea Times, Xinhua News, Hong Kong Standard, and the Straits Times. It is a collection of documents manually assessed for relevance to a set of queries from NTCIR-7 Advanced Cross-lingual Information Access (ACLIA) task. The corpus consists of 167 documents, or 4,711 sentences for 14 test topics. Each sentence is manually tagged with opinionatedness, polarity, and relevance to the topic by three annotators from a pool of six annotators.

For preprocessing, no removal or stemming is performed on the data. Each sentence was processed with the Stanford English parser[6] to produce a dependency parse tree. Only the Title fields of the topics were used.

For performance evaluations of opinion and polarity detection, we use precision, recall, and F-measure, the same measure used to report the official results at the NTCIR MOAT workshop. There are lenient and strict evaluations depending on the agreement of the annotators; if two out of three annotators agreed upon an opinion or polarity annotation then it is used during the lenient evaluation, similarly three out of three agreements are used during the strict evaluation. We present the performances using the lenient evaluation only, for the two evaluations generally do not show much difference in relative performance changes.

Since MOAT is a classification task, we use a threshold parameter to draw a boundary between opinionated and non-opinionated sentences. We report the performance of our system using the NTCIR-7 dataset, where the threshold parameter is optimized using the NTCIR-6 dataset.

4.2.2 Experimental Result

We present the performance of our sentiment analysis system in Table 2. As in the experiments with

[6] http://nlp.stanford.edu/software/lex-parser.shtml

Table 2: Performance of the Sentiment Analysis System on NTCIR7 dataset. System parameters are optimized for F-measure using NTCIR6 dataset with lenient evaluations.

Model	Opinionated		
	Precision	Recall	F-Measure
BASELINE	0.305	**0.866**	0.451
VS	0.331	0.807	**0.470**
BM25	0.327	0.795	0.464
LM	0.325	0.794	0.461
LSA	0.315	0.806	0.453
PMI	0.342	0.603	0.436
DTP	0.322	0.778	0.455
VS·LSA	0.335	0.769	0.466
VS·PMI	0.311	0.833	0.453
VS·DTP	0.342	0.745	0.469
VS·LSA·DTP	**0.349**	0.719	**0.470**
VS·PMI·DTP	0.328	0.773	0.461

the TREC dataset, using only the word sentiment model is used as our baseline.

Similarly to the TREC experiments, the features of the word generation model perform exceptionally better than that of the topic association model. The best performing feature of the word generation model is VS, achieving a 4.21% improvement over the baseline's f-measure. Interestingly, this is the tied top performing f-measure over all combinations of our features.

While LSA and DTP show mild improvements, PMI performed worse than baseline, with higher precision but a drop in recall. DTP was the best performing topic association model.

When combining the best performing feature of the word generation model (VS) with the features of the topic association model, LSA, PMI and DTP all performed worse than or as well as the VS in f-measure evaluation. LSA and DTP improves precision slightly, but with a drop in recall. PMI shows the opposite tendency.

The best performing system was achieved using VS, LSA and DTP at both precision and f-measure evaluations.

4.3 Classification task – SVM

4.3.1 Experimental Setting

To test our SVM classifier, we perform the classification task. Movie Review polarity dataset[7] was first introduced by Pang et al. (2002) to test various ML-based methods for sentiment classification. It is a balanced dataset of 700 positive and 700 negative reviews, collected from the Internet Movie Database (IMDb) archive. MPQA Corpus[8] contains 535 newspaper articles manually annotated at sentence and subsentence level for opinions and other private states (Wiebe et al., 2005).

To closely reproduce the experiment with the best performance carried out in (Pang et al., 2002) using SVM, we use unigram with the **presence** feature. We test various combinations of our features applicable to the task. For evaluation, we use ten-fold cross-validation accuracy.

Table 3: Average ten-fold cross-validation accuracies of polarity classification task with SVM.

Features	Accuracy	
	Movie-review	MPQA
PRESENCE	82.6	76.8
TF	71.1	76.5
VS.TF	81.3	76.7
BM25.TF	81.4	**77.9**
IDF	61.6	61.8
VS.IDF	83.6	**77.9**
BM25.IDF	83.6	77.8
VS.TF·VS.IDF	**83.8**	**77.9**
BM25.TF·BM25.IDF	**84.1**	77.7
BM25.TF·VS.IDF	**85.1**	77.7

4.3.2 Experimental Result

We present the sentiment classification performances in Table 3.

As observed by Pang et al. (2002), using the raw **tf** drops the accuracy of the sentiment classification (-13.92%) of movie-review data. Using the raw **idf** feature worsens the accuracy even more (-25.42%). Normalized **tf**-variants show improvements over **tf** but are worse than **presence**. Normalized **idf** features produce slightly better accuracy results than the baseline. Finally, combining any normalized **tf** and **idf** features improved the baseline (high 83% $\sim$ low 85%). The best combination was **BM25.TF·VS.IDF**.

MPQA corpus reveals similar but somewhat uncertain tendency.

[7] http://www.cs.cornell.edu/people/pabo/movie-review-data/

[8] http://www.cs.pitt.edu/mpqa/databaserelease/

4.4 Discussion

Overall, the opinion retrieval and the sentiment analysis models achieve improvements using our proposed features. Especially, the features of the word generation model improve the overall performances drastically. Its effectiveness is also verified with a data-driven approach; the accuracy of a sentiment classifier trained on a polarity dataset was improved by various combinations of normalized tf and idf statistics.

Differences in effectiveness of VS, BM25, and LM come from parameter tuning and corpus differences. For the TREC dataset, BM25 performed better than the other models, and for the NTCIR dataset, VS performed better.

Our features of the topic association model show mild improvement over the baseline performance in general. PMI and LSA, both modeling the semantic associations between words, show different behaviors on the datasets. For the NTCIR dataset, LSA performed better, while PMI is more effective for the TREC dataset. We believe that the explanation lies in the differences between the topics for each dataset. In general, the NTCIR topics are general descriptive words such as "regenerative medicine", "American economy after the 911 terrorist attacks", and "lawsuit brought against Microsoft for monopolistic practices." The TREC topics are more named-entity-like terms such as "Carmax", "Wikipedia primary source", "Jiffy Lube", "Starbucks", and "Windows Vista." We have experimentally shown that LSA is more suited to finding associations between general terms because its training documents are from a general domain.[9] Our PMI measure utilizes a web search engine, which covers a variety of named entity terms.

Though the features of our topic association model, WP and DTP, were evaluated on different datasets, we try our best to conjecture the differences. WP on TREC dataset shows a small improvement of MAP compared to other topic association features, while the precision is improved the most when this feature is used alone. The DTP feature displays similar behavior with precision. It also achieves the best f-measure over other topic association features. DTP achieves higher relative improvement (3.99% F-measure verse 2.32% MAP), and is more effective for improving the performance in combination with LSA and PMI.

[9]TASA Corpus, http://lsa.colorado.edu/spaces.html

5 Conclusion

In this paper, we proposed various term weighting schemes and how such features are modeled in the sentiment analysis task. Our proposed features include corpus statistics, association measures using semantic and local-context proximities. We have empirically shown the effectiveness of the features with our proposed opinion retrieval and sentiment analysis models.

There exists much room for improvement with further experiments with various term weighting methods and datasets. Such methods include, but by no means limited to, semantic similarities between word pairs using lexical resources such as WordNet (Miller, 1995) and data-driven methods with various topic-dependent term weighting schemes on labeled corpus with topics such as MPQA.

Acknowledgments

This work was supported in part by MKE & IITA through IT Leading R&D Support Project and in part by the BK 21 Project in 2009.

References

Kushal Dave, Steve Lawrence, and David M. Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of WWW*, pages 519–528.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th Conference on Language Resources and Evaluation (LREC'06)*, pages 417–422, Geneva, IT.

Hui Fang, Tao Tao, and ChengXiang Zhai. 2004. A formal study of information retrieval heuristics. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 49–56, New York, NY, USA. ACM.

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305.

Michael Gamon. 2004. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Vasileios Hatzivassiloglou and Kathleen R. Mckeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL'97)*, pages 174–181, madrid, ES.

Jaap Kamps, Maarten Marx, Robert J. Mokken, and Maarten De Rijke. 2004. Using wordnet to measure semantic orientation of adjectives. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*, pages 1115–1118, Lisbon, PT.

Taku Kudo and Yuji Matsumoto. 2004. A boosting algorithm for classification of semi-structured text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240, April.

Yeha Lee, Seung-Hoon Na, Jungi Kim, Sang-Hyob Nam, Hun young Jung, and Jong-Hyeok Lee. 2008. Kle at trec 2008 blog track: Blog post and feed retrieval. In *Proceedings of TREC-08*.

Craig Macdonald and Iadh Ounis. 2006. The TREC Blogs06 collection: creating and analysing a blog test collection. Technical Report TR-2006-224, Department of Computer Science, University of Glasgow.

Shotaro Matsumoto, Hiroya Takamura, and Manabu Okumura. 2005. Sentiment classification using word subsequences and dependency sub-trees. In *Proceedings of PAKDD'05, the 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. 2007. Topic sentiment mixture: Modeling facets and opinions in weblogs. In *Proceedings of WWW*, pages 171–180, New York, NY, USA. ACM Press.

George A. Miller. 1995. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41.

Tony Mullen and Nigel Collier. 2004. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 412–418, July. Poster paper.

Sang-Hyob Nam, Seung-Hoon Na, Yeha Lee, and Jong-Hyeok Lee. 2009. Diffpost: Filtering non-relevant content based on content difference between two consecutive blog posts. In *ECIR*.

Vincent Ng, Sajib Dasgupta, and S. M. Niaz Arifin. 2006. Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. In *Proceedings of the COLING/ACL Main Conference Poster Sessions*, pages 611–618, Sydney, Australia, July. Association for Computational Linguistics.

I. Ounis, M. de Rijke, C. Macdonald, G. A. Mishne, and I. Soboroff. 2006. Overview of the trec-2006 blog track. In *Proceedings of TREC-06*, pages 15–27, November.

I. Ounis, C. Macdonald, and I. Soboroff. 2008. Overview of the trec-2008 blog track. In *Proceedings of TREC-08*, pages 15–27, November.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

Yohei Seki, David Kirk Evans, Lun-Wei Ku, Le Sun, Hsin-Hsi Chen, and Noriko Kando. 2008. Overview of multilingual opinion analysis task at ntcir-7. In *Proceedings of The 7th NTCIR Workshop (2007/2008) - Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access*.

Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, and Daniel M. Ogilvie. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. MIT Press, Cambridge, USA.

Peter D. Turney and Michael L. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*, 21(4):315–346.

Peter D. Turney. 2001. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *EMCL '01: Proceedings of the 12th European Conference on Machine Learning*, pages 491–502, London, UK. Springer-Verlag.

Casey Whitelaw, Navendu Garg, and Shlomo Argamon. 2005. Using appraisal groups for sentiment analysis. In *Proceedings of the 14th ACM international conference on Information and knowledge management (CIKM'05)*, pages 625–631, Bremen, DE.

Janyce Wiebe, E. Breck, Christopher Buckley, Claire Cardie, P. Davis, B. Fraser, Diane Litman, D. Pierce, Ellen Riloff, Theresa Wilson, D. Day, and Mark Maybury. 2003. Recognizing and organizing opinions expressed in the world press. In *Proceedings of the 2003 AAAI Spring Symposium on New Directions in Question Answering*.

Janyce M. Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin. 2004. Learning subjective language. *Computational Linguistics*, 30(3):277–308, September.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2/3):164–210.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP'05)*, pages 347–354, Vancouver, CA.

Kiduk Yang, Ning Yu, Alejandro Valerio, and Hui Zhang. 2006. WIDIT in TREC-2006 Blog track. In *Proceedings of TREC*.

Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of 2003 Conference on the Empirical Methods in Natural Language Processing (EMNLP'03)*, pages 129–136, Sapporo, JP.

Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214.

Min Zhang and Xingyao Ye. 2008. A generation model to unify topic relevance and lexicon-based sentiment for opinion retrieval. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 411–418, New York, NY, USA. ACM.

Compiling a Massive, Multilingual Dictionary via Probabilistic Inference

Mausam Stephen Soderland Oren Etzioni
Daniel S. Weld Michael Skinner* Jeff Bilmes
University of Washington, Seattle *Google, Seattle
{mausam,soderlan,etzioni,weld,bilmes}@cs.washington.edu mskinner@google.com

Abstract

Can we automatically compose a large set of Wiktionaries and translation dictionaries to yield a massive, multilingual dictionary whose coverage is substantially greater than that of any of its constituent dictionaries?

The composition of multiple translation dictionaries leads to a transitive inference problem: if word A translates to word B which in turn translates to word C, what is the probability that C is a translation of A? The paper introduces a novel algorithm that solves this problem for 10,000,000 words in more than 1,000 languages. The algorithm yields PANDICTIONARY, a novel multilingual dictionary. PANDICTIONARY contains more than four times as many translations than in the largest Wiktionary at precision 0.90 and over 200,000,000 pairwise translations in over 200,000 language pairs at precision 0.8.

1 Introduction and Motivation

In the era of globalization, inter-lingual communication is becoming increasingly important. Although nearly 7,000 languages are in use today (Gordon, 2005), most language resources are mono-lingual, or bi-lingual.[1] This paper investigates whether Wiktionaries and other translation dictionaries available over the Web can be automatically composed to yield a massive, multilingual dictionary with superior coverage at comparable precision.

We describe the automatic construction of a massive multilingual translation dictionary, called PANDICTIONARY, that could serve as a resource for translation systems operating over a very broad set of language pairs. The most immediate application of PANDICTIONARY is to *lexical translation*—the translation of individual words or simple phrases (*e.g.*, "sweet potato"). Because lexical translation does not require aligned corpora as input, it is feasible for a much broader set of languages than statistical Machine Translation (SMT). Of course, lexical translation cannot replace SMT, but it is useful for several applications including translating search-engine queries, library classifications, meta-data tags,[2] and recent applications like cross-lingual image search (Etzioni et al., 2007), and enhancing multi-lingual Wikipedias (Adar et al., 2009). Furthermore, lexical translation is a valuable component in knowledge-based Machine Translation systems, *e.g.*, (Bond et al., 2005; Carbonell et al., 2006).

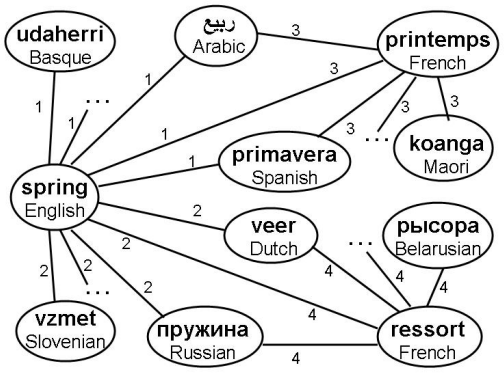


Figure 1: A fragment of the translation graph for two senses of the English word 'spring'. Edges labeled '1' and '3' are for spring in the sense of a season, and '2' and '4' are for the flexible coil sense. The graph shows translation entries from an English dictionary merged with ones from a French dictionary.

PANDICTIONARY currently contains over 200 million pairwise translations in over 200,000 language pairs at precision 0.8. It is constructed from information harvested from 631 online dictionaries and Wiktionaries. This necessitates match-

[1] The English Wiktionary, a lexical resource developed by volunteers over the Internet is one notable exception that contains translations of English words in about 500 languages.

[2] Meta-data tags appear in community Web sites such as `flickr.com` and `del.icio.us`.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 262–270,
Suntec, Singapore, 2-7 August 2009.

ing word senses across multiple, independently-authored dictionaries. Because of the millions of translations in the dictionaries, a feasible solution to this *sense matching* problem has to be scalable; because sense matches are imperfect and uncertain, the solution has to be probabilistic.

The core contribution of this paper is a principled method for probabilistic sense matching to *infer* lexical translations between two languages that do not share a translation dictionary. For example, our algorithm can conclude that Basque word 'udaherri' is a translation of Maori word 'koanga' in Figure 1. Our contributions are as follows:

1. We describe the design and construction of PANDICTIONARY—a novel lexical resource that spans over 200 million pairwise translations in over 200,000 language pairs at 0.8 precision, a four-fold increase when compared to the union of its input translation dictionaries.
2. We introduce SenseUniformPaths, a scalable probabilistic method, based on graph sampling, for inferring lexical translations, which finds 3.5 times more inferred translations at precison 0.9 than the previous best method.
3. We experimentally contrast PANDICTIONARY with the English Wiktionary and show that PANDICTIONARY is from 4.5 to 24 times larger depending on the desired precision.

The remainder of this paper is organized as follows. Section 2 describes our earlier work on sense matching (Etzioni et al., 2007). Section 3 describes how the PANDICTIONARY builds on and improves on their approach. Section 4 reports on our experimental results. Section 5 considers related work on lexical translation. The paper concludes in Section 6 with directions for future work.

2 Building a Translation Graph

In previous work (Etzioni et al., 2007) we introduced an approach to sense matching that is based on translation graphs (see Figure 1 for an example). Each vertex $v \in \mathcal{V}$ in the graph is an ordered pair (w, l) where w is a word in a language l. Undirected edges in the graph denote translations between words: an edge $e \in \mathcal{E}$ between (w_1, l_1) and (w_2, l_2) represents the belief that w_1 and w_2 share at least one word sense.

Construction: The Web hosts a large number of bilingual dictionaries in different languages and several Wiktionaries. Bilingual dictionaries translate words from one language to another, often without distinguishing the intended sense. For example, an Indonesian-English dictionary gives 'light' as a translation of the Indonesian word 'enteng', but does not indicate whether this means illumination, light weight, light color, or the action of lighting fire.

The Wiktionaries (wiktionary.org) are sense-distinguished, multilingual dictionaries created by volunteers collaborating over the Web. A translation graph is constructed by locating these dictionaries, parsing them into a common XML format, and adding the nodes and edges to the graph.

Figure 1 shows a fragment of a translation graph, which was constructed from two sets of translations for the word 'spring' from an English Wiktionary, and two corresponding entries from a French Wiktionary for 'printemps' (spring season) and 'ressort' (flexible spring). Translations of the season 'spring' have edges labeled with sense ID=1, the flexible coil sense has ID=2, translations of 'printemps' have ID=3, and so forth.[3]

For clarity, we show only a few of the actual vertices and edges; *e.g.*, the figure doesn't show the edge (ID=1) between 'udaherri' and 'primavera'.

Inference: In our previous system we had a simple inference procedure over translation graphs, called TRANSGRAPH, to find translations beyond those provided by any source dictionary. TRANSGRAPH searched for paths in the graph between two vertices and estimated the probability that the path maintains the same word sense along all edges in the path, even when the edges come from different dictionaries. For example, there are several paths between 'udaherri' and 'koanga' in Figure 1, but all shift from sense ID 1 to 3. The probability that the two words are translations is equivalent to the probability that IDs 1 and 3 represent the same sense.

TRANSGRAPH used two formulae to estimate these probabilities. One formula estimates the probability that two multi-lingual dictionary entries represent the same word sense, based on the proportion of overlapping translations for the two entries. For example, most of the translations of

[3] Sense-distinguished multi-lingual entries give rise to cliques all of which share a common sense ID.

French 'printemps' are also translations of the season sense of 'spring'. A second formula is based on triangles in the graph (useful for bilingual dictionaries): a clique of 3 nodes with an edge between each pair of nodes. In such cases, there is a high probability that all 3 nodes share a word sense.

Critique: While TRANSGRAPH was the first to present a scalable inference method for lexical translation, it suffers from several drawbacks. Its formulae operate only on local information: pairs of senses that are adjacent in the graph or triangles. It does not incorporate evidence from longer paths when an explicit triangle is not present. Moreover, the probabilities from different paths are combined conservatively (either taking the max over all paths, or using "noisy or" on paths that are completely disjoint, except end points), thus leading to suboptimal precision/recall.

In response to this critique, the next section presents an inference algorithm, called SenseUniformPaths (SP), with substantially improved recall at equivalent precision.

3 Translation Inference Algorithms

In essence, inference over a translation graph amounts to *transitive* sense matching: if word A translates to word B, which translates in turn to word C, what is the probability that C is a translation of A? If B is polysemous then C may not share a sense with A. For example, in Figure 2(a) if A is the French word 'ressort' (the flexible-coil sense of spring) and B is the English word 'spring', then Slovenian word 'vzmet' may or may not be a correct translation of 'ressort' depending on whether the edge (B, C) denotes the flexible-coil sense of spring, the season sense, or another sense. Indeed, given only the knowledge of the path $A - B - C$ we cannot claim anything with certainty regarding A to C.

However, if A, B, and C are on a circuit that starts at A, passes through B and C and returns to A, there is a high probability that all nodes on that circuit share a common word sense, given certain restrictions that we enumerate later. Where TRANSGRAPH used evidence from circuits of length 3, we extend this to paths of arbitrary lengths.

To see how this works, let us begin with the simplest circuit, a triangle of three nodes as shown in Figure 2(b). We can be quite certain that 'vzmet' shares the sense of coil with both 'spring' and 'ressort'. Our reasoning is as follows: even though both 'ressort' and 'spring' are polysemous they share only one sense. For a triangle to form we have two choices – (1) either 'vzmet' means spring coil, or (2) 'vzmet' means *both* the spring season and jurisdiction, but not spring coil. The latter is possible but such a coincidence is very unlikely, which is why a triangle is strong evidence for the three words to share a sense.

As an example of longer paths, our inference algorithms can conclude that in Figure 2(c), both 'molla' and 'vzmet' have the sense coil, even though no explicit triangle is present. To show this, let us define a *translation circuit* as follows:

Definition 1 *A* translation circuit from v_1^* with sense s^* *is a cycle that starts and ends at* v_1^* *with no repeated vertices (other than* v_1^* *at end points). Moreover, the path includes an edge between* v_1^* *and another vertex* v_2^* *that also has sense* s^*.

All vertices on a translation circuit are mutual translations with high probability, as in Figure 2(c). The edge from 'spring' indicates that 'vzmet' means either coil or season, while the edge from 'ressort' indicates that 'molla' means either coil or jurisdiction. The edge from 'vzmet' to 'molla' indicates that they share a sense, which will happen if all nodes share the sense season or if either 'vzmet' has the unlikely combination of coil and jurisdiction (or 'molla' has coil and season).

We also develop a mathematical model of sense-assignment to words that lets us formally prove these insights. For more details on the theory please refer to our extended version. This paper reports on our novel algorithm and experimental results.

These insights suggest a basic version of our algorithm: "given two vertices, v_1^* and v_2^*, that share a sense (say s^*) compute all translation circuits from v_1^* in the sense s^*; mark all vertices in the circuits as translations of the sense s^*".

To implement this algorithm we need to decide whether a vertex lies on a translation circuit, which is trickier than it seems. Notice that knowing that v is connected independently to v_1^* and v_2^* doesn't imply that there exists a translation circuit through v, because both paths may go through a common node, thus violating of the definition of translation circuit. For example, in Figure 2(d) the Catalan word 'ploma' has paths to both spring and ressort, but there is no translation circuit through

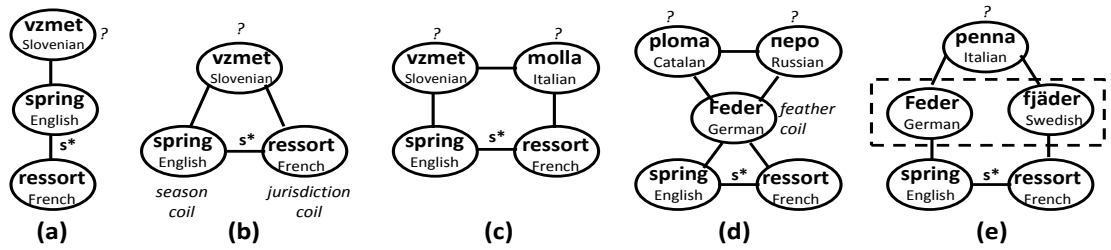


Figure 2: Snippets of translation graphs illustrating various inference scenarios. The nodes in question mark represent the nodes in focus for each illustration. For all cases we are trying to infer translations of the flexible coil sense of spring.

it. Hence, it will not be considered a translation. This example also illustrates potential errors avoided by our algorithm – here, German word 'Feder' mean feather and spring coil, but 'ploma' means feather and not the coil.

An exhaustive search to find translation circuits would be too slow, so we approximate the solution by a random walk scheme. We start the random walk from v_1^* (or v_2^*) and choose random edges without repeating any vertices in the current path. At each step we check if the current node has an edge to v_2^* (or v_1^*). If it does, then all the vertices in the current path form a translation circuit and, thus, are valid translations. We repeat this random walk many times and keep marking the nodes. In our experiments for each inference task we performed a total of 2,000 random walks (N_R in pseudo-code) of max circuit length 7. We chose these parameters based on a development set of 50 inference tasks.

Our first experiments with this basic algorithm resulted in a much higher recall than TRANSGRAPH, albeit, at a significantly lower precision. A closer examination of the results revealed two sources of error – (1) errors in source dictionary data, and (2) correlated sense shifts in translation circuits. Below we add two new features to our algorithm to deal with each of these error sources, respectively.

3.1 Errors in Source Dictionaries

In practice, source dictionaries contain mistakes and errors occur in processing the dictionaries to create the translation graph. Thus, existence of a *single* translation circuit is only limited evidence for a vertex as a translation. We wish to exploit the insight that more translation circuits constitute stronger evidence. However, the different circuits may share some edges, and thus the evidence cannot be simply the number of translation circuits.

We model the errors in dictionaries by assigning a probability less than 1.0 to each edge[4] (p_e in the pseudo-code). We assume that the probability of an edge being erroneous is independent of the rest of the graph. Thus, a translation graph with possible data errors converts into a *distribution* over accurate translation graphs.

Under this distribution, we can use the probability of existence of a translation circuit through a vertex as the probability that the vertex is a translation. This value captures our insights, since a larger number of translation circuits gives a higher probability value.

We sample different graph topologies from our given distribution. Some translation circuits will exist in some of the sampled graphs, but not in others. This, in turn, means that a given vertex v will only be on a circuit for a fraction of the sampled graphs. We take the proportion of samples in which v is on a circuit to be the probability that v is in the translation set. We refer to this algorithm as *Unpruned* SenseUniformPaths (uSP).

3.2 Avoiding Correlated Sense-shifts

The second source of errors are circuits that include a pair of nodes sharing the same polysemy, *i.e.*, having the same pair of senses. A circuit might maintain sense s^* until it reaches a node that has both s^* and a distinct s_i. The next edge may lead to a node with s_i, but not s^*, causing an extraction error. The path later shifts back to sense s^* at a second node that *also* has s^* and s_i. An example for this is illustrated in Figure 2(e), where both the German and Swedish words mean feather and spring coil. Here, Italian 'penna' means only the feather and not the coil.

Two nodes that share the same two senses occur frequently in practice. For example, many languages use the same word for 'heart' (the organ) and center; similarly, it is common for languages to use the same word for 'silver', the metal and the color. These correlations stem from com-

[4]In our experiments we used a flat value of 0.6, chosen by parameter tuning on a development set of 50 inference tasks. In future we can use different values for different dictionaries based on our confidence in their accuracy.

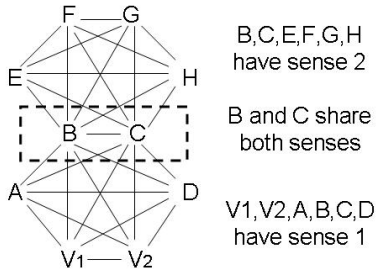


Figure 3: The set {B, C} has a shared ambiguity - each node has both sense 1 (from the lower clique) and sense 2 (from the upper clique). A circuit that contains two nodes from the same ambiguity set with an intervening node not in that set is likely to create translation errors.

mon metaphor and the shared evolutionary roots of some languages.

We are able to avoid circuits with this type of correlated sense-shift by automatically identifying *ambiguity sets*, sets of nodes known to share multiple senses. For instance, in Figure 2(e) 'Feder' and 'fjäder' form an ambiguity set (shown within dashed lines), as they both mean feather and coil.

Definition 2 *An* ambiguity set A *is a set of vertices that all share the same two senses. I.e.,* $\exists s_1, s_2,$ with $s_1 \neq s_2$ s.t. $\forall v \in A,$ *sense*$(v, s_1) \wedge$ *sense*(v, s_2)*, where* $sense(v, s)$ *denotes that* v *has sense* s*.*

To increase the precision of our algorithm we *prune* the circuits that contain two nodes in the same ambiguity set and also have one or more intervening nodes that are not in the ambiguity set. There is a strong likelihood that the intervening nodes will represent a translation error.

Ambiguity sets can be detected from the graph topology as follows. Each clique in the graph represents a set of vertices that share a common word sense. When two cliques intersect in two or more vertices, the intersecting vertices share the word sense of both cliques. This may either mean that both cliques represent the same word sense, or that the intersecting vertices form an ambiguity set. A large overlap between two cliques makes the former case more likely; a small overlap makes it more likely that we have found an ambiguity set.

Figure 3 illustrates one such computation. All nodes of the clique V_1, V_2, A, B, C, D share a word sense, and all nodes of the clique B, C, E, F, G, H also share a word sense. The set $\{B, C\}$ has nodes that have both senses, forming an ambiguity set. We denote the set of ambiguity sets by $\mathcal{A}$ in the pseudo-code.

Having identified these ambiguity sets, we modify our random walk scheme by keeping track of whether we are entering or leaving an ambiguity set. We prune away all paths that enter the same ambiguity set twice. We name the resulting algorithm SenseUniformPaths (SP), summarized at a high level in Algorithm 1.

Comparing Inference Algorithms Our evaluation demonstrated that SP outperforms uSP. Both these algorithms have significantly higher recall than TRANSGRAPH algorithm. The detailed results are presented in Section 4.2. We choose SP as our inference algorithm for all further research, in particular to create PANDICTIONARY.

3.3 Compiling PanDictionary

Our goal is to automatically compile PANDICTIONARY, a sense-distinguished lexical translation resource, where each entry is a distinct word sense. Associated with each word sense is a list of translations in multiple languages.

We use Wiktionary senses as the base senses for PANDICTIONARY. Recall that SP requires two nodes (v_1^* and v_2^*) for inference. We use the Wiktionary source word as v_1^* and automatically pick the second word from the set of Wiktionary translations of that sense by choosing a word that is well connected, and, which does not appear in other senses of v_1^* (*i.e.*, is expected to share only one sense with v_1^*).

We first run SenseUniformPaths to expand the approximately 50,000 senses in the English Wiktionary. We further expand any senses from the other Wiktionaries that are not yet covered by PANDICTIONARY, and add these to PANDICTIONARY. This results in the creation of the world's largest multilingual, sense-distinguished translation resource, PANDICTIONARY. It contains a little over 80,000 senses. Its construction takes about three weeks on a 3.4 GHz processor with a 2 GB memory.

Algorithm 1 S.P.$(G, v_1^*, v_2^*, \mathcal{A})$

parameters N_G: no. of graph samples, N_R: no. of random walks, p_e: prob. of sampling an edge
create N_G versions of G by sampling each edge independently with probability p_e
for all $i = 1..N_G$ **do**
  for all vertices v : $rp[v][i] = 0$
  perform N_R random walks starting at v_1^* (or v_2^*) and pruning any walk that enters (or exits) an ambiguity set in $\mathcal{A}$ twice. All walks that connect to v_2^* (or v_1^*) form a translation circuit.
  for all vertices v **do**
    if(v is on a translation circuit) $rp[v][i] = 1$
return $\frac{\sum_i rp[v][i]}{N_G}$ as the prob. that v is a translation

4 Empirical Evaluation

In our experiments we investigate three key questions: (1) which of the three algorithms (TG, uSP and SP) is superior for translation inference (Section 4.2)? (2) how does the coverage of PANDICTIONARY compare with the largest existing multilingual dictionary, the English Wiktionary (Section 4.3)? (3) what is the benefit of inference over the mere aggregation of 631 dictionaries (Section 4.4)? Additionally, we evaluate the inference algorithm on two other dimensions – variation with the degree of polysemy of source word, and variation with original size of the seed translation set.

4.1 Experimental Methodology

Ideally, we would like to evaluate a random sample of the more than 1,000 languages represented in PANDICTIONARY.[5] However, a high-quality evaluation of translation between two languages requires a person who is fluent in both languages. Such people are hard to find and may not even exist for many language pairs (*e.g.*, Basque and Maori). Thus, our evaluation was guided by our ability to recruit volunteer evaluators. Since we are based in an English speaking country we were able to recruit local volunteers who are fluent in a range of languages and language families, and who are also bilingual in English.[6]

The experiments in Sections 4.2 and 4.3 test whether translations in a PANDICTIONARY have accurate word senses. We provided our evaluators with a random sample of translations into their native language. For each translation we showed the English source word and gloss of the intended sense. For example, a Dutch evaluator was shown the sense 'free (not imprisoned)' together with the Dutch word 'loslopende'. The instructions were to mark a word as correct if it could be used to express the intended sense in a sentence in their native language. For experiments in Section 4.4 we tested precision of *pairwise* translations, by having informants in several pairs of languages discuss whether the words in their respective languages can be used for the same sense.

We use the tags of correct or incorrect to compute the precision: the percentage of correct translations divided by correct plus incorrect translations. We then order the translations by probability and compute the precision at various probability thresholds.

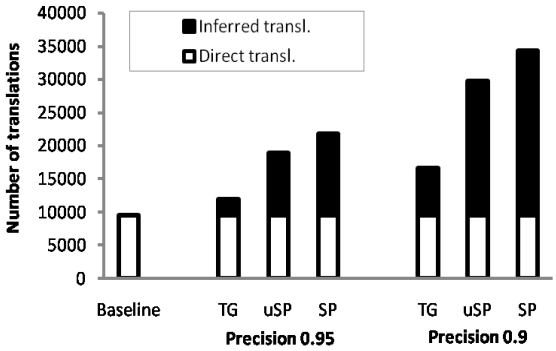


Figure 4: The SenseUniformPaths algorithm (SP) more than doubles the number of correct translations at precision 0.95, compared to a baseline of translations that can be found without inference.

4.2 Comparing Inference Algorithms

Our first evaluation compares our SenseUniformPaths (SP) algorithm (before and after pruning) with TRANSGRAPH on both precision and number of translations.

To carry out this comparison, we randomly sampled 1,000 senses from English Wiktionary and ran the three algorithms over them. We evaluated the results on 7 languages – Chinese, Danish, German, Hindi, Japanese, Russian, and Turkish. Each informant tagged 60 random translations inferred by each algorithm, which resulted in 360-400 tags per algorithm[7]. The precision over these was taken as a surrogate for the precision across all the senses.

We compare the number of translations for each algorithm at comparable precisions. The baseline is the set of translations (for these 1000 senses) found in the source dictionaries without inference, which has a precision 0.95 (as evaluated by our informants).[8]

Our results are shown in Figure 4. At this high precision, SP more than doubles the number of baseline translations, finding 5 times as many inferred translations (in black) as TG.

Indeed, both uSP and SP massively outperform TG. SP is consistently better than uSP, since it performs better for polysemous words, due to its pruning based on ambiguity sets. We conclude

[5]The distribution of words in PANDICTIONARY is highly non-uniform ranging from 182,988 words in English to 6,154 words in Luxembourgish and 189 words in Tuvalu.

[6]The languages used was based on the availability of native speakers. This varied between the different experiments, which were conducted at different times.

[7]Some translations were marked as "Don't know".

[8]Our informants tended to underestimate precision, often marking correct translations in minor senses of a word as incorrect.

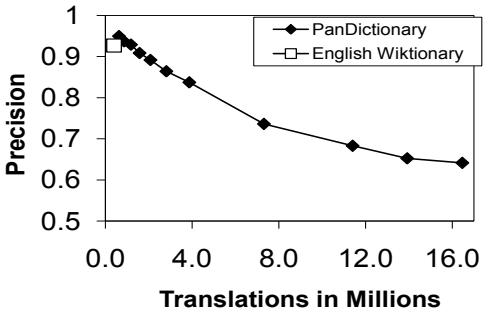


Figure 5: Precision *vs.* coverage curve for PANDICTIONARY. It quadruples the size of the English Wiktionary at precision 0.90, is more than 8 times larger at precision 0.85 and is almost 24 times the size at precision 0.7.

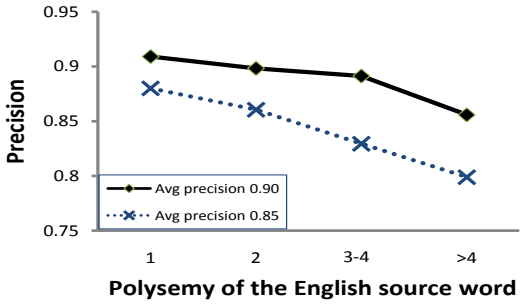


Figure 6: Variation of precision with the degree of polysemy of the source English word. The precision decreases as polysemy increases, still maintaining reasonably high values.

that SP is the best inference algorithm and employ it for PANDICTIONARY construction.

4.3 Comparison with English Wiktionary

We now compare the coverage of PANDICTIONARY with the English Wiktionary at varying levels of precision. The English Wiktionary is the largest Wiktionary with a total of 403,413 translations. It is also more reliable than some other Wiktionaries in making word sense distinctions. In this study we use only the subset of PANDICTIONARY that was computed starting from the English Wiktionary senses. Thus, this subsection under-reports PANDICTIONARY's coverage.

To evaluate a huge resource such as PANDICTIONARY we recruited native speakers of 14 languages – Arabic, Bulgarian, Danish, Dutch, German, Hebrew, Hindi, Indonesian, Japanese, Korean, Spanish, Turkish, Urdu, and Vietnamese. We randomly sampled 200 translations per language, which resulted in about 2,500 tags. Figure 5 shows the total number of translations in PANDICTIONARY in senses from the English Wiktionary. At precision 0.90, PANDICTIONARY has 1.8 million translations, 4.5 times as many as the English Wiktionary.

We also compare the coverage of PANDICTIONARY with that of the English Wiktionary in terms of languages covered. Table 1 reports, for each resource, the number of languages that have a minimum number of distinct words in the resource. PANDICTIONARY has 1.4 times as many languages with at least 1,000 translations at precision 0.90 and more than twice at precision 0.7. These observations reaffirm our faith in the pan-lingual nature of the resource.

PANDICTIONARY's ability to expand the lists of translations provided by the English Wiktionary is most pronounced for senses with a small number of translations. For example, at precision 0.90, senses that originally had 3 to 6 translations are increased 5.3 times in size. The increase is 2.2 times when the original sense size is greater than 20.

For closer analysis we divided the English source words (v_1^*) into different bins based on the number of senses that English Wiktionary lists for them. Figure 6 plots the variation of precision with this degree of polysemy. We find that translation quality decreases as degree of polysemy increases, but this decline is gradual, which suggests that SP algorithm is able to hold its ground well in difficult inference tasks.

4.4 Comparison with All Source Dictionaries

We have shown that PANDICTIONARY has much broader coverage than the English Wiktionary, but how much of this increase is due to the inference algorithm versus the mere aggregation of hundreds of translation dictionaries in PANDICTIONARY?

Since most bilingual dictionaries are not sense-distinguished, we ignore the word senses and count the number of distinct (word1, word2) translation pairs.

We evaluated the precision of word-word translations by a *collaborative tagging* scheme, with two native speakers of different languages, who are both bi-lingual in English. For each suggested translation they discussed the various senses of words in their respective languages and tag a translation correct if they found some sense that is shared by both words. For this study we tagged 7 language pairs: Hindi-Hebrew,

	# languages with distinct words		
	≥ 1000	≥ 100	≥ 1
English Wiktionary	49	107	505
PanDictionary (0.90)	67	146	608
PanDictionary (0.85)	75	175	794
PanDictionary (0.70)	107	607	1066

Table 1: PANDICTIONARY covers substantially more languages than the English Wiktionary.

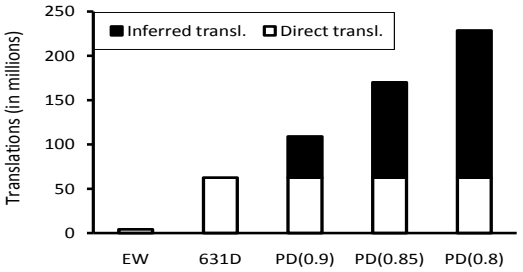


Figure 7: The number of distinct word-word translation pairs from PANDICTIONARY is several times higher than the number of translation pairs in the English Wiktionary (EW) or in all 631 source dictionaries combined (631 D). A majority of PANDICTIONARY translations are inferred by combining entries from multiple dictionaries.

Japanese-Russian, Chinese-Turkish, Japanese-German, Chinese-Russian, Bengali-German, and Hindi-Turkish.

Figure 7 compares the number of word-word translation pairs in the English Wiktionary (EW), in all 631 source dictionaries (631 D), and in PANDICTIONARY at precisions 0.90, 0.85, and 0.80. PANDICTIONARY increases the number of word-word translations by 73% over the source dictionary translations at precision 0.90 and increases it by 2.7 times at precision 0.85. PANDICTIONARY also adds value by identifying the word sense of the translation, which is not given in most of the source dictionaries.

5 Related Work

Because we are considering a relatively new problem (automatically building a panlingual translation resource) there is little work that is directly related to our own. The closest research is our previous work on TRANSGRAPH algorithm (Etzioni et al., 2007). Our current algorithm outperforms the previous state of the art by 3.5 times at precision 0.9 (see Figure 4). Moreover, we compile this in a dictionary format, thus considerably reducing the response time compared to TRANSGRAPH, which performed inference at query time.

There has been considerable research on methods to acquire translation lexicons from either MRDs (Neff and McCord, 1990; Helmreich et al., 1993; Copestake et al., 1994) or from parallel text (Gale and Church, 1991; Fung, 1995; Melamed, 1997; Franz et al., 2001), but this has generally been limited to a small number of languages. Manually engineered dictionaries such as EuroWordNet (Vossen, 1998) are also limited to a relatively small set of languages. There is some recent work on compiling dictionaries from monolingual corpora, which may scale to several language pairs in future (Haghighi et al., 2008).

Little work has been done in combining multiple dictionaries in a way that maintains word senses across dictionaries. Gollins and Sanderson (2001) explored using triangulation between alternate pivot languages in cross-lingual information retrieval. Their triangulation essentially mixes together circuits for all word senses, hence, is unable to achieve high precision.

Dyvik's "semantic mirrors" uses translation paths to tease apart distinct word senses from inputs that are not sense-distinguished (Dyvik, 2004). However, its expensive processing and reliance on parallel corpora would not scale to large numbers of languages. Earlier (Knight and Luk, 1994) discovered senses of Spanish words by matching several English translations to a WordNet synset. This approach applies only to specific kinds of bilingual dictionaries, and also requires a taxonomy of synsets in the target language.

Random walks, graph sampling and Monte Carlo simulations are popular in literature, though, to our knowledge, none have applied these to our specific problems (Henzinger et al., 1999; Andrieu et al., 2003; Karger, 1999).

6 Conclusions

We have described the automatic construction of a unique multilingual translation resource, called PANDICTIONARY, by performing probabilistic inference over the translation graph. Overall, the construction process consists of large scale information extraction over the Web (parsing dictionaries), combining it into a single resource (a translation graph), and then performing automated reasoning over the graph (SenseUniformPaths) to yield a much more extensive and useful knowledge base.

We have shown that PANDICTIONARY has more coverage than any other existing bilingual or multilingual dictionary. Even at the high precision of 0.90, PANDICTIONARY more than quadruples the size of the English Wiktionary, the largest available multilingual resource today.

We plan to make PANDICTIONARY available to the research community, and also to the Wiktionary community in an effort to bolster their efforts. PANDICTIONARY entries can suggest new translations for volunteers to add to Wiktionary entries, particularly if combined with an intelligent editing tool (*e.g.*, (Hoffmann et al., 2009)).

Acknowledgments

This research was supported by a gift from the Utilika Foundation to the Turing Center at University of Washington. We acknowledge Paul Beame, Nilesh Dalvi, Pedro Domingos, Rohit Khandekar, Daniel Lowd, Parag, Jonathan Pool, Hoifung Poon, Vibhor Rastogi, Gyanit Singh for fruitful discussions and insightful comments on the research. We thank the language experts who donated their time and language expertise to evaluate our systems. We also thank the anynomous reviewers of the previous drafts of this paper for their valuable suggestions in improving the evaluation and presentation.

References

E. Adar, M. Skinner, and D. Weld. 2009. Information arbitrage in multi-lingual Wikipedia. In *Procs. of Web Search and Data Mining(WSDM 2009)*.

C. Andrieu, N. De Freitas, A. Doucet, and M. Jordan. 2003. An Introduction to MCMC for Machine Learning. *Machine Learning*, 50:5–43.

F. Bond, S. Oepen, M. Siegel, A. Copestake, and D D. Flickinger. 2005. Open source machine translation with DELPH-IN. In *Open-Source Machine Translation Workshop at MT Summit X*.

J. Carbonell, S. Klein, D. Miller, M. Steinbaum, T. Grassiany, and J. Frey. 2006. Context-based machine translation. In *AMTA*.

A. Copestake, T. Briscoe, P. Vossen, A. Ageno, I. Castellon, F. Ribas, G. Rigau, H. Rodriquez, and A. Samiotou. 1994. Acquisition of lexical translation relations from MRDs. *Machine Translation*, 3(3–4):183–219.

H. Dyvik. 2004. Translation as semantic mirrors: from parallel corpus to WordNet. *Language and Computers*, 49(1):311–326.

O. Etzioni, K. Reiter, S. Soderland, and M. Sammer. 2007. Lexical translation with application to image search on the Web. In *Machine Translation Summit XI*.

M. Franz, S. McCarly, and W. Zhu. 2001. English-Chinese information retrieval at IBM. In *Proceedings of TREC 2001*.

P. Fung. 1995. A pattern matching method for finding noun and proper noun translations from noisy parallel corpora. In *Proceedings of ACL-1995*.

W. Gale and K.W. Church. 1991. A Program for Aligning Sentences in Bilingual Corpora. In *Proceedings of ACL-1991*.

T. Gollins and M. Sanderson. 2001. Improving cross language retrieval with triangulated translation. In *SIGIR*.

Raymond G. Gordon, Jr., editor. 2005. *Ethnologue: Languages of the World (Fifteenth Edition)*. SIL International.

A. Haghighi, P. Liang, T. Berg-Kirkpatrick, and D. Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *ACL*.

S. Helmreich, L. Guthrie, and Y. Wilks. 1993. The use of machine readable dictionaries in the Pangloss project. In *AAAI Spring Symposium on Building Lexicons for Machine Translation*.

Monika R. Henzinger, Allan Heydon, Michael Mitzenmacher, and Marc Najork. 1999. Measuring index quality using random walks on the web. In *WWW*.

R. Hoffmann, S. Amershi, K. Patel, F. Wu, J. Fogarty, and D. S. Weld. 2009. Amplifying community content creation with mixed-initiative information extraction. In *ACM SIGCHI (CHI2009)*.

D. R. Karger. 1999. A randomized fully polynomial approximation scheme for the all-terminal network reliability problem. *SIAM Journal of Computation*, 29(2):492–514.

K. Knight and S. Luk. 1994. Building a large-scale knowledge base for machine translation. In *AAAI*.

I.D. Melamed. 1997. A Word-to-Word Model of Translational Equivalence. In *Proceedings of ACL-1997 and EACL-1997*, pages 490–497.

M. Neff and M. McCord. 1990. Acquiring lexical data from machine-readable dictionary resources for machine translation. In *3rd Intl Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language*.

P. Vossen, editor. 1998. *EuroWordNet: A multilingual database with lexical semantic networds*. Kluwer Academic Publishers.

A Metric-based Framework for Automatic Taxonomy Induction

Hui Yang
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
huiyang@cs.cmu.edu

Jamie Callan
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
callan@cs.cmu.edu

Abstract

This paper presents a novel metric-based framework for the task of automatic taxonomy induction. The framework incrementally clusters terms based on *ontology metric*, a score indicating semantic distance; and transforms the task into a multi-criteria optimization based on minimization of taxonomy structures and modeling of term abstractness. It combines the strengths of both lexico-syntactic patterns and clustering through incorporating heterogeneous features. The flexible design of the framework allows a further study on which features are the best for the task under various conditions. The experiments not only show that our system achieves higher F1-measure than other state-of-the-art systems, but also reveal the interaction between features and various types of relations, as well as the interaction between features and term abstractness.

1 Introduction

Automatic taxonomy induction is an important task in the fields of Natural Language Processing, Knowledge Management, and Semantic Web. It has been receiving increasing attention because semantic taxonomies, such as WordNet (Fellbaum, 1998), play an important role in solving knowledge-rich problems, including question answering (Harabagiu et al., 2003) and textual entailment (Geffet and Dagan, 2005). Nevertheless, most existing taxonomies are manually created at great cost. These taxonomies are rarely complete; it is difficult to include new terms in them from emerging or rapidly changing domains. Moreover, manual taxonomy construction is time-consuming, which may make it unfeasible for specialized domains and personalized tasks. Automatic taxonomy induction is a solution to augment existing resources and to produce new taxonomies for such domains and tasks.

Automatic taxonomy induction can be decomposed into two subtasks: term extraction and relation formation. Since term extraction is relatively easy, relation formation becomes the focus of most research on automatic taxonomy induction. In this paper, we also assume that terms in a taxonomy are given and concentrate on the subtask of relation formation.

Existing work on automatic taxonomy induction has been conducted under a variety of names, such as ontology learning, semantic class learning, semantic relation classification, and relation extraction. The approaches fall into two main categories: *pattern-based* and *clustering-based*. *Pattern-based* approaches define lexical-syntactic patterns for relations, and use these patterns to discover instances of relations. *Clustering-based* approaches hierarchically cluster terms based on similarities of their meanings usually represented by a vector of quantifiable features.

Pattern-based approaches are known for their high accuracy in recognizing instances of relations if the patterns are carefully chosen, either manually (Berland and Charniak, 1999; Kozareva et al., 2008) or via automatic bootstrapping (Hearst, 1992; Widdows and Dorow, 2002; Girju et al., 2003). The approaches, however, suffer from sparse coverage of patterns in a given corpus. Recent studies (Etzioni et al., 2005; Kozareva et al., 2008) show that if the size of a corpus, such as the Web, is nearly unlimited, a pattern has a higher chance to explicitly appear in the corpus. However, corpus size is often not that large; hence the problem still exists. Moreover, since patterns usually extract instances in pairs, the approaches suffer from the problem of inconsistent concept chains after connecting pairs of instances to form taxonomy hierarchies.

Clustering-based approaches have a main advantage that they are able to discover relations

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 271–279,
Suntec, Singapore, 2-7 August 2009.

which do not explicitly appear in text. They also avoid the problem of inconsistent chains by addressing the structure of a taxonomy globally from the outset. Nevertheless, it is generally believed that clustering-based approaches cannot generate relations as accurate as pattern-based approaches. Moreover, their performance is largely influenced by the types of features used. The common types of features include *contextual (*Lin, 1998), *co-occurrence* (Yang and Callan, 2008)*,* and *syntactic dependency* (Pantel and Lin, 2002; Pantel and Ravichandran, 2004). So far there is no systematic study on which features are the best for automatic taxonomy induction under various conditions.

This paper presents a metric-based taxonomy induction framework. It combines the strengths of both pattern-based and clustering-based approaches by incorporating lexico-syntactic patterns as one type of features in a clustering framework. The framework integrates contextual, co-occurrence, syntactic dependency, lexical-syntactic patterns, and other features to learn an *ontology metric*, a score indicating semantic distance, for each pair of terms in a taxonomy; it then incrementally clusters terms based on their ontology metric scores. The incremental clustering is transformed into an optimization problem based on two assumptions: *minimum evolution* and *abstractness*. The flexible design of the framework allows a further study of the interaction between features and relations, as well as that between features and term abstractness.

2 Related Work

There has been a substantial amount of research on automatic taxonomy induction. As we mentioned earlier, two main approaches are *pattern-based* and *clustering-based*.

Pattern-based approaches are the main trend for automatic taxonomy induction. Though suffering from the problems of sparse coverage and inconsistent chains, they are still popular due to their simplicity and high accuracy. They have been applied to extract various types of lexical and semantic relations, including *is-a*, *part-of*, *sibling*, *synonym*, *causal*, and many others.

Pattern-based approaches started from and still pay a great deal of attention to the most common *is-a* relations. Hearst (1992) pioneered using a hand crafted list of hyponym patterns as seeds and employing bootstrapping to discover *is-a* relations. Since then, many approaches (Mann, 2002; Etzioni et al., 2005; Snow et al., 2005) have used Hearst-style patterns in their work on *is-a* relations. For instance, Mann (2002) extracted *is-a* relations for proper nouns by Hearst-style patterns. Pantel et al. (2004) extended *is-a* relation acquisition towards terascale, and automatically identified hypernym patterns by minimal edit distance.

Another common relation is *sibling*, which describes the relation of sharing similar meanings and being members of the same class. Terms in *sibling* relations are also known as *class members* or *similar terms*. Inspired by the conjunction and appositive structures, Riloff and Shepherd (1997), Roark and Charniak (1998) used co-occurrence statistics in local context to discover *sibling* relations. The KnowItAll system (Etzioni et al., 2005) extended the work in (Hearst, 1992) and bootstrapped patterns on the Web to discover siblings; it also ranked and selected the patterns by statistical measures. Widdows and Dorow (2002) combined symmetric patterns and graph link analysis to discover *sibling* relations. Davidov and Rappoport (2006) also used symmetric patterns for this task. Recently, Kozareva et al. (2008) combined a double-anchored hyponym pattern with graph structure to extract siblings.

The third common relation is *part-of*. Berland and Charniak (1999) used two meronym patterns to discover *part-of* relations, and also used statistical measures to rank and select the matching instances. Girju et al. (2003) took a similar approach to Hearst (1992) for *part-of* relations.

Other types of relations that have been studied by pattern-based approaches include question-answer relations (such as *birthdates* and *inventor*) (Ravichandran and Hovy, 2002), synonyms and antonyms (Lin et al., 2003), general purpose analogy (Turney et al., 2003), verb relations (including *similarity, strength, antonym, enablement* and *temporal*) (Chklovski and Pantel, 2004), entailment (Szpektor et al., 2004), and more specific relations, such as *purpose, creation* (Cimiano and Wenderoth, 2007), *LivesIn*, and *EmployedBy* (Bunescu and Mooney , 2007).

The most commonly used technique in pattern-based approaches is *bootstrapping* (Hearst, 1992; Etzioni et al., 2005; Girju et al., 2003; Ravichandran and Hovy, 2002; Pantel and Pennacchiotti, 2006). It utilizes a few man-crafted seed patterns to extract instances from corpora, then extracts new patterns using these instances, and continues the cycle to find new instances and new patterns. It is effective and scalable to large datasets; however, uncontrolled bootstrapping

soon generates undesired instances once a noisy pattern brought into the cycle.

To aid bootstrapping, methods of pattern quality control are widely applied. Statistical measures, such as point-wise mutual information (Etzioni et al., 2005; Pantel and Pennacchiotti, 2006) and conditional probability (Cimiano and Wenderoth, 2007), have been shown to be effective to rank and select patterns and instances. Pattern quality control is also investigated by using WordNet (Girju et al., 2006), graph structures built among terms (Widdows and Dorow, 2002; Kozareva et al., 2008), and pattern clusters (Davidov and Rappoport, 2008).

Clustering-based approaches usually represent word contexts as vectors and cluster words based on similarities of the vectors (Brown et al., 1992; Lin, 1998). Besides contextual features, the vectors can also be represented by verb-noun relations (Pereira et al., 1993), syntactic dependency (Pantel and Ravichandran, 2004; Snow et al., 2005), co-occurrence (Yang and Callan, 2008), conjunction and appositive features (Caraballo, 1999). More work is described in (Buitelaar et al., 2005; Cimiano and Volker, 2005). Clustering-based approaches allow discovery of relations which do not explicitly appear in text. Pantel and Pennacchiotti (2006), however, pointed out that clustering-based approaches generally fail to produce coherent cluster for small corpora. In addition, clustering-based approaches had only applied to solve *is-a* and *sibling* relations.

Many clustering-based approaches face the challenge of appropriately labeling non-leaf clusters. The labeling amplifies the difficulty in creation and evaluation of taxonomies. Agglomerative clustering (Brown et al., 1992; Caraballo, 1999; Rosenfeld and Feldman, 2007; Yang and Callan, 2008) iteratively merges the most similar clusters into bigger clusters, which need to be labeled. Divisive clustering, such as CBC (Clustering By Committee) which constructs cluster centroids by averaging the feature vectors of a subset of carefully chosen cluster members (Pantel and Lin, 2002; Pantel and Ravichandran, 2004), also need to label the parents of split clusters. In this paper, we take an incremental clustering approach, in which terms and relations are added into a taxonomy one at a time, and their parents are from the existing taxonomy. The advantage of the incremental approach is that it eliminates the trouble of inventing cluster labels and concentrates on placing terms in the correct positions in a taxonomy hierarchy.

The work by Snow et al. (2006) is the most similar to ours because they also took an incremental approach to construct taxonomies. In their work, a taxonomy grows based on maximization of conditional probability of relations given evidence; while in our work based on optimization of taxonomy structures and modeling of term abstractness. Moreover, our approach employs heterogeneous features from a wide range; while their approach only used syntactic dependency. We compare system performance between (Snow et al., 2006) and our framework in Section 5.

3 The Features

The features used in this work are indicators of semantic relations between terms. Given two input terms c_x, c_y, a feature is defined as a function generating a single numeric score $h(c_x, c_y) \in \mathbb{R}$ or a vector of numeric scores $h(c_x, c_y) \in \mathbb{R}^n$. The features include *contextual*, *co-occurrence*, *syntactic dependency*, *lexical-syntactic patterns*, and *miscellaneous*.

The first set of features captures *contextual* information of terms. According to Distributional Hypothesis (Harris, 1954), words appearing in similar contexts tend to be similar. Therefore, word meanings can be inferred from and represented by contexts. Based on the hypothesis, we develop the following features: (1) *Global Context KL-Divergence*: The global context of each input term is the search results collected through querying search engines against several corpora (Details in Section 5.1). It is built into a unigram language model without smoothing for each term. This feature function measures the Kullback-Leibler divergence (KL divergence) between the language models associated with the two inputs. (2) *Local Context KL-Divergence:* The local context is the collection of all the left two and the right two words surrounding an input term. Similarly, the local context is built into a unigram language model without smoothing for each term; the feature function outputs KL divergence between the models.

The second set of features is *co-occurrence*. In our work, co-occurrence is measured by point-wise mutual information between two terms:

$$pmi(c_x, c_y) = \log \frac{Count(c_x, c_y)}{Count(c_x)Count(c_y)}$$

where *Count(.)* is defined as the number of documents or sentences containing the term(s); or *n* as in "Results 1-10 of about *n* for *term*" appearing on the first page of Google search results for a term or the concatenation of a term pair. Based

Hypernym Patterns	*Sibling Patterns*
NP_x (,)?and/or other NP_y	NP_x and/or NP_y
such NP_y as NP_x	*Part-of Patterns*
NP_y (,)? such as NP_x	NP_x of NP_y
NP_y (,)? including NP_x	NP_y's NP_x
NP_y (,)? especially NP_x	NP_y has/had/have NP_x
NP_y like NP_x	NP_y is made (up)? of NP_x
NP_y called NP_x	NP_y comprises NP_x
NP_x is a/an NP_y	NP_y consists of NP_x
NP_x , a/an NP_y	

Table 1. Lexico-Syntactic Patterns.

on different definitions of *Count(.),* we have (3) *Document PMI*, (4) *Sentence PMI*, and (5) *Google PMI* as the co-occurrence features.

The third set of features employs *syntactic dependency* analysis. We have (6) *Minipar Syntactic Distance* to measure the average length of the shortest syntactic paths (in the first syntactic parse tree returned by Minipar[1]) between two terms in sentences containing them, (7) *Modifier Overlap, (8) Object Overlap, (9) Subject Overlap, and (10) Verb Overlap* to measure the number of overlaps between modifiers, objects, subjects, and verbs, respectively, for the two terms in sentences containing them. We use Assert[2] to label the semantic roles.

The fourth set of features is *lexical-syntactic patterns*. We have (11) *Hypernym Patterns* based on patterns proposed by (Hearst, 1992) and (Snow et al., 2005), (12) *Sibling Patterns* which are basically conjunctions, and (13) *Part-of Patterns* based on patterns proposed by (Girju et al., 2003) and (Cimiano and Wenderoth, 2007). Table 1 lists all patterns. Each feature function returns a vector of scores for two input terms, one score per pattern. A score is 1 if two terms match a pattern in text, 0 otherwise.

The last set of features is *miscellaneous*. We have (14) *Word Length Difference* to measure the length difference between two terms, and (15) *Definition Overlap* to measure the number of word overlaps between the term definitions obtained by querying Google with *"*define*:term"*.

These heterogeneous features vary from simple statistics to complicated syntactic dependency features, basic word length to comprehensive Web-based contextual features. The flexible design of our learning framework allows us to use all of them, and even allows us to use different sets of them under different conditions, for instance, different types of relations and different abstraction levels. We study the interaction between features and relations and that between features and abstractness in Section 5.

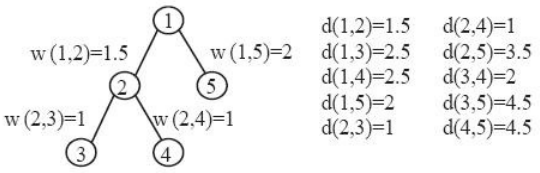


Figure 1. Illustration of Ontology Metric.

4 The Metric-based Framework

This section presents the metric-based framework which incrementally clusters terms to form taxonomies. By minimizing the changes of taxonomy structures and modeling term abstractness at each step, it finds the optimal position for each term in a taxonomy. We first introduce definitions, terminologies and assumptions about taxonomies; then, we formulate automatic taxonomy induction as a multi-criterion optimization and solve it by a greedy algorithm; lastly, we show how to estimate ontology metrics.

4.1 Taxonomies, Ontology Metric, Assumptions, and Information Functions

We define a *taxonomy* T as a data model that represents a set of terms C and a set of relations R between these terms. T can be written as $T(C,R)$. Note that for the subtask of relation formation, we assume that the term set C is given. A *full taxonomy* is a tree containing all the terms in C. A *partial taxonomy* is a tree containing only a subset of terms in C.

In our framework, automatic taxonomy induction is the process to construct a full taxonomy $\hat{T}$ given a set of terms C and an initial partial taxonomy $T^0(S^0, R^0)$, where $S^0 \subseteq C$. Note that T^0 is possibly empty. The process starts from the initial partial taxonomy T^0 and randomly adds terms from C to T^0 one by one, until a full taxonomy is formed, i.e., all terms in C are added.

Ontology Metric

We define an *ontology metric* as a distance measure between two terms (c_x,c_y) in a taxonomy $T(C,R)$. Formally, it is a function $d: C \times C \rightarrow \mathbb{R}_+$, where C is the set of terms in T. An ontology metric d on a taxonomy T with edge weights w for any term pair $(c_x,c_y) \in C$ is the sum of all edge weights along the shortest path between the pair:

$$d_{(T,w)}(c_x, c_y) = \sum_{e_{x,y} \in P(x,y)} w(e_{x,y})$$

[1] http://www.cs.ualberta.ca/lindek/minipar.htm.
[2] http://cemantix.org/assert.

where $P(x, y)$ is the set of edges defining the shortest path from term c_x to c_y . Figure 1 illustrates ontology metrics for a 5-node taxonomy. Section 4.3 presents the details of learning *ontology metrics*.

Information Functions

The amount of information in a taxonomy T is measured and represented by an information function *Info(T)*. An information function is defined as the sum of the ontology metrics among a set of term pairs. The function can be defined over a taxonomy, or on a single level of a taxonomy. For a taxonomy *T(C,R)*, we define its information function as:

$$Info(T) = \sum_{x<y, c_x, c_y \in C} d(c_x, c_y) \quad (1)$$

Similarly, we define the information function for an abstraction level L_i as:

$$Info_i(L_i) = \sum_{x<y, c_x, c_y \in L_i} d(c_x, c_y) \quad (2)$$

where L_i is the subset of terms lying at the i^{th} level of a taxonomy T. For example, in Figure 1, node 1 is at level L_1, node 2 and node 5 level L_2.

Assumptions

Given the above definitions about taxonomies, we make the following assumptions:

Minimum Evolution Assumption. Inspired by the minimum evolution tree selection criterion widely used in phylogeny (Hendy and Penny, 1985), we assume that a good taxonomy not only minimizes the overall semantic distance among the terms but also avoid dramatic changes. Construction of a full taxonomy is proceeded by adding terms one at a time, which yields a series of partial taxonomies. After adding each term, the current taxonomy T^{n+1} from the previous taxonomy T^n is one that introduces the least changes between the information in the two taxonomies:

$$T^{n+1} = \arg\min_{T'} \Delta Info(T^n, T')$$

where the information change function is $\Delta Info(T^a, T^b) = | Info(T^a) - Info(T^b) |$.

Abstractness Assumption. In a taxonomy, concrete concepts usually lay at the bottom of the hierarchy while abstract concepts often occupy the intermediate and top levels. Concrete concepts often represent physical entities, such as "basketball" and "mercury pollution". While abstract concepts, such as "science" and "economy", do not have a physical form thus we must imagine their existence. This obvious difference suggests that there is a need to treat them differently in taxonomy induction. Hence we assume that terms at the same abstraction level have common characteristics and share the same *Info(.)* function. We also assume that terms at different abstraction levels have different characteristics; hence they do not necessarily share the same *Info(.)* function. That is to say, $\forall$ concept $c \in T$, abstraction level $L_i \subset T$, $c \in L_i \Rightarrow c$ uses $Info_i(.)$.

4.2 Problem Formulation

The Minimum Evolution Objective

Based on the minimum evolution assumption, we define the goal of taxonomy induction is to find the optimal full taxonomy $\hat{T}$ such that the information changes are the least since the initial partial taxonomy T^0, i.e., to find:

$$\hat{T} = \arg\min_{T'} \Delta Info(T^0, T') \quad (3)$$

where T' is a full taxonomy, i.e., the set of terms in T' equals C.

To find the optimal solution for Equation (3), $\hat{T}$, we need to find the optimal term set $\hat{C}$ and the optimal relation set $\hat{R}$. Since the optimal term set for a full taxonomy is always C, the only unknown part left is $\hat{R}$. Thus, Equation (3) can be transformed equivalently into:

$$\hat{R} = \arg\min_{R'} \Delta Info(T'(C, R'), T^0(S^0, R^0))$$

Note that in the framework, terms are added incrementally into a taxonomy. Each term insertion yields a new partial taxonomy T. By the minimum evolution assumption, the optimal next partial taxonomy is one gives the least information change. Therefore, the updating function for the set of relations R^{n+1} after a new term z is inserted can be calculated as:

$$\hat{R} = \arg\min_{R'} \Delta Info(T(S^n \cup \{z\}, R'), T(S^n, R^n))$$

By plugging in the definition of the information change function $\Delta Info(.,.)$ in Section 4.1 and Equation (1), the updating function becomes:

$$\hat{R} = \arg\min_{R'} \left| \sum_{c_x, c_y \in S^n \cup \{z\}} d(c_x, c_y) - \sum_{c_x, c_y \in S^n} d(c_x, c_y) \right|$$

The above updating function can be transformed into a minimization problem:

$$\min u$$
$$\text{subject to } u \le \sum_{c_x, c_y \in S^n \cup \{z\}} d(c_x, c_y) - \sum_{c_x, c_y \in S^n} d(c_x, c_y)$$
$$u \le \sum_{c_x, c_y \in S^n} d(c_x, c_y) - \sum_{c_x, c_y \in S^n \cup \{z\}} d(c_x, c_y)$$
$$x < y$$

The minimization follows the minimum evolution assumption; hence we call it the *minimum evolution objective*.

The Abstractness Objective

The *abstractness* assumption suggests that term abstractness should be modeled explicitly by learning separate information functions for terms at different abstraction levels. We approximate an information function by a linear interpolation of some underlying feature functions. Each abstraction level L_i is characterized by its own information function $Info_i(.)$. The least square fit of $Info_i(.)$ is: $\min | Info_i(L_i) - W_i^T H_i |^2$.

By plugging Equation (2) and minimizing over every abstraction level, we have:

$$\min \sum_i (\sum_{c_x, c_y \in L_i} d(c_x, c_y) - \sum_j w_{i,j} h_{i,j}(c_x, c_y))^2$$

where $h_{i,j}(.,.)$ is the j^{th} underlying feature function for term pairs at level L_i, $w_{i,j}$ is the weight for $h_{i,j}(.,.)$. This minimization follows the abstractness assumption; hence we call it the *abstractness objective*.

The Multi-Criterion Optimization Algorithm

We propose that both *minimum evolution* and *abstractness* objectives need to be satisfied. To optimize multiple criteria, the Pareto optimality needs to be satisfied (Boyd and Vandenberghe, 2004). We handle this by introducing $\lambda \in [0,1]$ to control the contribution of each objective. The multi-criterion optimization function is:

$$\begin{aligned} &\min \lambda u + (1-\lambda) v \\ \text{subject to} \quad &u \le \sum_{c_x, c_y \in S^n \cup \{z\}} d(c_x, c_y) - \sum_{c_x, c_y \in S^n} d(c_x, c_y) \\ &u \le \sum_{c_x, c_y \in S^n} d(c_x, c_y) - \sum_{c_x, c_y \in S^n \cup \{z\}} d(c_x, c_y) \\ &v = \sum_i (\sum_{\substack{c_x, c_y \in L_i \\ x < y}} d(c_x, c_y) - \sum_j w_{i,j} h_{i,j}(c_x, c_y))^2 \end{aligned}$$

The above optimization can be solved by a greedy optimization algorithm. At each term insertion step, it produces a new partial taxonomy by adding to the existing partial taxonomy a new term z, and a new set of relations $R(z,.)$. z is attached to every nodes in the existing partial taxonomy; and the algorithm selects the optimal position indicated by $R(z,.)$, which minimizes the multi-criterion objective function. The algorithm is:

foreach $z \in C \setminus S$

$S \rightarrow S \cup \{z\}$;

$R \rightarrow R \cup \{\arg\min_{R_{(z,.)}} (\lambda u + (1-\lambda) v)\}$;

Output $T(S, R)$;

The above algorithm presents a general incremental clustering procedure to construct taxonomies. By minimizing the taxonomy structure changes and modeling term abstractness at each step, it finds the optimal position of each term in the taxonomy hierarchy.

Statistics	*WN/is-a*	*ODP/is-a*	*WN/part-of*
#taxonomies	50	50	50
#terms	1,964	2,210	1,812
Avg #terms	39	44	37
Avg depth	6	6	5

Table 2. Data Statistics.

4.3 Estimating Ontology Metric

Learning a good ontology metric is important for the multi-criterion optimization algorithm. In this work, the estimation and prediction of ontology metric are achieved by ridge regression (Hastie et al., 2001). In the training data, an ontology metric $d(c_x,c_y)$ for a term pair (c_x,c_y) is generated by assuming every edge weight as 1 and summing up all the edge weights along the shortest path from c_x to c_y. We assume that there are some underlying feature functions which measure the semantic distance from term c_x to c_y. A weighted combination of these functions approximates the ontology metric for (c_x,c_y):

$$d(x, y) = \sum_j w_j h_j(c_x, c_y)$$

where w_j is the j^{th} weight for $h_j(c_x, c_y)$, the j^{th} feature function. The feature functions are generated as mentioned in Section 3.

5 Experiments

5.1 Data

The gold standards used in the evaluation are hypernym taxonomies extracted from WordNet and ODP (Open Directory Project), and meronym taxonomies extracted from WordNet. In WordNet taxonomy extraction, we only use the word senses within a particular taxonomy to ensure no ambiguity. In ODP taxonomy extraction, we parse the topic lines, such as "Topic r:id=`Top/Arts/Movies'", in the XML databases to obtain relations, such as *is_a(movies, arts)*. In total, there are 100 hypernym taxonomies, 50 each extracted from WordNet[3] and ODP[4], and 50 meronym taxonomies from WordNet[5]. Table 2

[3] WordNet hypernym taxonomies are from 12 topics: *gathering, professional, people, building, place, milk, meal, water, beverage, alcohol, dish,* and *herb.*

[4] ODP hypernym taxonomies are from 16 topics: *computers, robotics, intranet, mobile computing, database, operating system, linux, tex, software, computer science, data communication, algorithms, data formats, security multimedia,* and *artificial intelligence.*

[5] WordNet meronym taxonomies are from 15 topics: *bed*, car, *building*, *lamp*, *earth*, *television*, *body, drama, theatre, water, airplane, piano, book, computer,* and *watch.*

WordNet/*is-a*			
System	*Precision*	*Recall*	*F1-measure*
HE	**0.85**	0.32	0.46
GI	n/a	n/a	n/a
PR	0.75	0.73	0.74
ME	0.82	**0.79**	**0.82**
ODP/*is-a*			
System	*Precision*	*Recall*	*F1-measure*
HE	0.31	0.29	0.30
GI	n/a	n/a	n/a
PR	0.60	**0.72**	0.65
ME	**0.64**	0.70	**0.67**
WordNet/*part-of*			
System	*Precision*	*Recall*	*F1-measure*
HE	n/a	n/a	n/a
GI	**0.75**	0.25	0.38
PR	0.68	0.52	0.59
ME	0.69	**0.55**	**0.61**

Table 3. System Performance.

summarizes the data statistics.

We also use two Web-based auxiliary datasets to generate features mentioned in Section 3:

- *Wikipedia corpus.* The entire Wikipedia corpus is downloaded and indexed by Indri[6]. The top 100 documents returned by Indri are the global context of a term when querying with the term.
- *Google corpus*. A collection of the top 1000 documents by querying Google using each term, and each term pair. Each top 1000 documents are the global context of a query term.

Both corpora are split into sentences and are used to generate contextual, co-occurrence, syntactic dependency and lexico-syntactic pattern features.

5.2 Methodology

We evaluate the quality of automatic generated taxonomies by comparing them with the gold standards in terms of precision, recall and F1-measure. F1-measure is calculated as *2*P*R/(P+R)*, where *P* is *precision*, the percentage of correctly returned relations out of the total returned relations, *R* is *recall*, the percentage of correctly returned relations out of the total relations in the gold standard.

Leave-one-out cross validation is used to average the system performance across different training and test datasets. For each 50 datasets from WordNet hypernyms, WordNet meronyms or ODP hypernyms, we randomly pick 49 of them to generate training data, and test on the remaining dataset. We repeat the process for 50 times, with different training and test sets at each time, and report the averaged precision, recall and F1-measure across all 50 runs.

We also group the fifteen features in Section 3 into six sets: contextual, co-concurrence, patterns, syntactic dependency, word length difference and definition. Each set is turned on one by one for experiments in Section 5.4 and 5.5.

Feature	*is-a*	*sibling*	*part-of*	***Benefited Relations***
Contextual	0.21	**0.42**	0.12	*sibling*
Co-occur.	**0.48**	**0.41**	**0.28**	*All*
Patterns	**0.46**	**0.41**	**0.30**	*All*
Syntactic	0.22	**0.36**	0.12	*sibling*
Word Leng.	0.16	0.16	0.15	*All but limited*
Definition	0.12	0.18	0.10	*Sibling but limited*
Best Features	Co-occur., patterns	Contextual, co-occur., patterns	Co-occur., patterns	

Table 4. F1-measure for Features vs. Relations: WordNet.

5.3 Performance of Taxonomy Induction

In this section, we compare the following automatic taxonomy induction systems: *HE*, the system by Hearst (1992) with 6 hypernym patterns; *GI*, the system by Girju et al. (2003) with 3 meronym patterns; *PR*, the probabilistic framework by Snow et al. (2006); and *ME*, the metric-based framework proposed in this paper. To have a fair comparison, for *PR*, we estimate the conditional probability of a relation given the evidence $P(R_{ij}|E_{ij})$, as in (Snow et al. 2006), by using the same set of features as in *ME*.

Table 3 shows precision, recall, and F1-measure of each system for WordNet hypernyms (*is-a*), WordNet meronyms (*part-of*) and ODP hypernyms (*is-a*). Bold font indicates the best performance in a column. Note that *HE* is not applicable to *part-of*, so is *GI* to *is-a*.

Table 3 shows that systems using heterogeneous features (*PR* and *ME*) achieve higher F1-measure than systems only using patterns (*HE* and *GI*) with a significant absolute gain of >30%. Generally speaking, pattern-based systems show higher precision and lower recall, while systems using heterogeneous features show lower precision and higher recall. However, when considering both precision and recall, using heterogeneous features is more effective than just using patterns. The proposed system *ME* consistently produces the best F1-measure for all three tasks.

The performance of the systems for ODP/*is-a* is worse than that for WordNet/*is-a*. This may be because there is more noise in ODP than in

[6] http://www.lemurproject.org/indri/.

Feature	L_2	L_3	L_4	L_5	L_6
Contextual	0.29	0.31	**0.35**	**0.36**	**0.36**
Co-occurrence	**0.47**	**0.56**	**0.45**	**0.41**	**0.41**
Patterns	**0.47**	**0.44**	**0.42**	**0.39**	**0.40**
Syntactic	0.31	0.28	**0.36**	**0.38**	**0.39**
Word Length	0.16	0.16	0.16	0.16	0.16
Definition	0.12	0.12	0.12	0.12	0.12

Table 5. F1-measure for Features vs. Abstractness: WordNet/*is-a*.

Feature	L_2	L_3	L_4	L_5	L_6
Contextual	**0.30**	**0.30**	**0.33**	**0.29**	**0.29**
Co-occurrence	**0.34**	**0.36**	**0.34**	**0.31**	**0.31**
Patterns	0.23	0.25	**0.30**	**0.28**	**0.28**
Syntactic	0.18	0.18	0.23	**0.27**	**0.27**
Word Length	0.15	0.15	0.15	0.14	0.14
Definition	0.13	0.13	0.13	0.12	0.12

Table 6. F1-measure for Features vs. Abstractness: ODP/*is-a*.

WordNet. For example, under *artificial intelligence*, ODP has *neural networks*, *natural language* and *academic departments*. Clearly, *academic departments* is not a hyponym of *artificial intelligence*. The noise in ODP interferes with the learning process, thus hurts the performance.

5.4 Features vs. Relations

This section studies the impact of different sets of features on different types of relations. Table 4 shows F1-measure of using each set of features alone on taxonomy induction for WordNet *is-a*, *sibling*, and *part-of* relations. Bold font means a feature set gives a major contribution to the task of automatic taxonomy induction for a particular type of relation.

Table 4 shows that different relations favor different sets of features. Both co-occurrence and lexico-syntactic patterns work well for all three types of relations. It is interesting to see that simple co-occurrence statistics work as good as lexico-syntactic patterns. Contextual features work well for *sibling* relations, but not for *is-a* and *part-of*. Syntactic features also work well for *sibling,* but not for *is-a* and *part-of*. The similar behavior of contextual and syntactic features may be because that four out of five syntactic features (*Modifier, Subject, Object,* and *Verb overlaps*) are just surrounding context for a term.

Comparing the *is-a* and *part-of* columns in Table 4 and the *ME* rows in Table 3, we notice a significant difference in F1-measure. It indicates that combination of heterogeneous features gives more rise to the system performance than a single set of features does.

5.5 Features vs. Abstractness

This section studies the impact of different sets of features on terms at different abstraction levels. In the experiments, F1-measure is evaluated for terms at each level of a taxonomy, not the whole taxonomy. Table 5 and 6 demonstrate F1-measure of using each set of features alone on each abstraction levels. Columns 2-6 are indices of the levels in a taxonomy. The larger the indices are, the lower the levels. Higher levels contain abstract terms, while lower levels contain concrete terms. L_1 is ignored here since it only contains a single term, the root. Bold font indicates good performance in a column.

Both tables show that abstract terms and concrete terms favor different sets of features. In particular, contextual, co-occurrence, pattern, and syntactic features work well for terms at L_4-L_6, i.e., concrete terms; co-occurrence works well for terms at L_2-L_3, i.e., abstract terms. This difference indicates that terms at different abstraction levels have different characteristics; it confirms our *abstractness assumption* in Section 4.1.

We also observe that for abstract terms in WordNet, patterns work better than contextual features; while for abstract terms in ODP, the conclusion is the opposite. This may be because that WordNet has a richer vocabulary and a more rigid definition of hypernyms, and hence *is-a* relations in WordNet are recognized more effectively by using lexico-syntactic patterns; while ODP contains more noise, and hence it favors features requiring less rigidity, such as the contextual features generated from the Web.

6 Conclusions

This paper presents a novel metric-based taxonomy induction framework combining the strengths of lexico-syntactic patterns and clustering. The framework incrementally clusters terms and transforms automatic taxonomy induction into a multi-criteria optimization based on minimization of taxonomy structures and modeling of term abstractness. The experiments show that our framework is effective; it achieves higher F1-measure than three state-of-the-art systems. The paper also studies which features are the best for different types of relations and for terms at different abstraction levels.

Most prior work uses a single rule or feature function for automatic taxonomy induction at all levels of abstraction. Our work is a more general framework which allows a wider range of features and different metric functions at different abstraction levels. This more general framework has the potential to learn more complex taxonomies than previous approaches.

Acknowledgements

This research was supported by NSF grant IIS-0704210. Any opinions, findings, conclusions, or recommendations expressed in this paper are of the authors, and do not necessarily reflect those of the sponsor.

References

M. Berland and E. Charniak. 1999. Finding parts in very large corpora. ACL'99.

S. Boyd and L. Vandenberghe. 2004. Convex optimization. In Cambridge University Press, 2004.

P. Brown, V. D. Pietra, P. deSouza, J. Lai, and R. Mercer. 1992. Class-based ngram models for natural language. Computational Linguistics, 18(4):468–479.

P. Buitelaar, P. Cimiano, and B. Magnini. 2005. Ontology Learning from Text: Methods, Evaluation and Applications. Volume 123 Frontiers in Artificial Intelligence and Applications.

R. Bunescu and R. Mooney. 2007. Learning to Extract Relations from the Web using Minimal Supervision. ACL'07.

S. Caraballo. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. ACL'99.

T. Chklovski and P. Pantel. 2004. VerbOcean: mining the web for fine-grained semantic verb relations. EMNLP '04.

P. Cimiano and J. Volker. 2005. Towards large-scale, open-domain and ontology-based named entity classification. RANLP'07.

P. Cimiano and J. Wenderoth. 2007. Automatic Acquisition of Ranked Qualia Structures from the Web. ACL'07.

D. Davidov and A. Rappoport. 2006. Efficient Unsupervised Discovery of Word Categories Using Symmetric Patterns and High Frequency Words. ACL'06.

D. Davidov and A. Rappoport. 2008. Classification of Semantic Relationships between Nominals Using Pattern Clusters. ACL'08.

D. Downey, O. Etzioni, and S. Soderland. 2005. A Probabilistic model of redundancy in information extraction. IJCAI'05.

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: an experimental study. Artificial Intelligence, 165(1):91–134.

C. Fellbuam. 1998. WordNet: An Electronic Lexical Database. MIT Press. 1998.

M. Geffet and I. Dagan. 2005. The Distributional Inclusion Hypotheses and Lexical Entailment. ACL'05.

R. Girju, A. Badulescu, and D. Moldovan. 2003. Learning Semantic Constraints for the Automatic Discovery of Part-Whole Relations. HLT'03.

R. Girju, A. Badulescu, and D. Moldovan. 2006. Automatic Discovery of Part-Whole Relations. Computational Linguistics, 32(1): 83-135.

Z. Harris. 1985. Distributional structure. In Word, 10(23): 146-162s, 1954.

T. Hastie, R. Tibshirani and J. Friedman. 2001. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer-Verlag, 2001.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. COLING'92.

M. D. Hendy and D. Penny. 1982. Branch and bound algorithms to determine minimal evolutionary trees. Mathematical Biosciences 59: 277-290.

Z. Kozareva, E. Riloff, and E. Hovy. 2008. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs. ACL'08.

D. Lin, 1998. Automatic retrieval and clustering of similar words. COLING'98.

D. Lin, S. Zhao, L. Qin, and M. Zhou. 2003. Identifying Synonyms among Distributionally Similar Words. IJCAI'03.

G. S. Mann. 2002. Fine-Grained Proper Noun Ontologies for Question Answering. In Proceedings of SemaNet' 02: Building and Using Semantic Networks, Taipei.

P. Pantel and D Lin. 2002. Discovering word senses from text. SIGKDD'02.

P. Pantel and D. Ravichandran. 2004. Automatically labeling semantic classes. HLT/NAACL'04.

P. Pantel, D. Ravichandran, and E. Hovy. 2004. Towards terascale knowledge acquisition. COLING'04.

P. Pantel and M. Pennacchiotti. 2006. Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. ACL'06.

F. Pereira, N. Tishby, and L. Lee. 1993. Distributional clustering of English words. ACL'93.

D. Ravichandran and E. Hovy. 2002. Learning surface text patterns for a question answering system. ACL'02.

E. Riloff and J. Shepherd. 1997. A corpus-based approach for building semantic lexicons. EMNLP'97.

B. Roark and E. Charniak. 1998. Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction. ACL/COLING'98.

R. Snow, D. Jurafsky, and A. Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. NIPS'05.

R. Snow, D. Jurafsky, and A. Y. Ng. 2006. Semantic Taxonomy Induction from Heterogeneous Evidence. ACL'06.

B. Rosenfeld and R. Feldman. 2007. Clustering for unsupervised relation identification. CIKM'07.

P. Turney, M. Littman, J. Bigham, and V. Shnayder. 2003. Combining independent modules to solve multiple-choice synonym and analogy problems. RANLP'03.

S. M. Harabagiu, S. J. Maiorano and M. A. Pasca. 2003. Open-Domain Textual Question Answering Techniques. Natural Language Engineering 9 (3): 1-38, 2003.

I. Szpektor, H. Tanev, I. Dagan, and B. Coppola. 2004. Scaling web-based acquisition of entailment relations. EMNLP'04.

D. Widdows and B. Dorow. 2002. A graph model for unsupervised Lexical acquisition. COLING '02.

H. Yang and J. Callan. 2008. Learning the Distance Metric in a Personal Ontology. Workshop on Ontologies and Information Systems for the Semantic Web of CIKM'08.

Learning with Annotation Noise

Eyal Beigman
Olin Business School
Washington University in St. Louis
beigman@wustl.edu

Beata Beigman Klebanov
Kellogg School of Management
Northwestern University
beata@northwestern.edu

Abstract

It is usually assumed that the kind of noise existing in annotated data is random classification noise. Yet there is evidence that differences between annotators are not always random attention slips but could result from different biases towards the classification categories, at least for the harder-to-decide cases. Under an annotation generation model that takes this into account, there is a hazard that some of the training instances are actually hard cases with unreliable annotations. We show that these are relatively unproblematic for an algorithm operating under the 0-1 loss model, whereas for the commonly used voted perceptron algorithm, hard training cases could result in incorrect prediction on the *uncontroversial* cases at test time.

1 Introduction

It is assumed, often tacitly, that the kind of noise existing in human-annotated datasets used in computational linguistics is random classification noise (Kearns, 1993; Angluin and Laird, 1988), resulting from annotator attention slips randomly distributed across instances. For example, Osborne (2002) evaluates noise tolerance of shallow parsers, with random classification noise taken to be "crudely approximating annotation errors." It has been shown, both theoretically and empirically, that this type of noise is tolerated well by the commonly used machine learning algorithms (Cohen, 1997; Blum et al., 1996; Osborne, 2002; Reidsma and Carletta, 2008).

Yet this might be overly optimistic. Reidsma and op den Akker (2008) show that apparent differences between annotators are not random slips of attention but rather result from different biases annotators might have towards the classification categories. When training data comes from one annotator and test data from another, the first annotator's biases are sometimes systematic enough for a machine learner to pick them up, with detrimental results for the algorithm's performance on the test data. A small subset of doubly annotated data (for inter-annotator agreement check) and large chunks of singly annotated data (for training algorithms) is not uncommon in computational linguistics datasets; such a setup is prone to problems if annotators are differently biased.[1]

Annotator bias is consistent with a number of noise models. For example, it could be that an annotator's bias is exercised on each and every instance, making his preferred category likelier for any instance than in another person's annotations. Another possibility, recently explored by Beigman Klebanov and Beigman (2009), is that some items are really quite clear-cut for an annotator with any bias, belonging squarely within one particular category. However, some instances – termed **hard cases** therein – are harder to decide upon, and this is where various preferences and biases come into play. In a metaphor annotation study reported by Beigman Klebanov et al. (2008), certain markups received overwhelming annotator support when people were asked to validate annotations after a certain time delay. Other instances saw opinions split; moreover, Beigman Klebanov et al. (2008) observed cases where people retracted their own earlier annotations.

To start accounting for such annotator behavior, Beigman Klebanov and Beigman (2009) proposed a model where instances are either **easy**, and then all annotators agree on them, or **hard**, and then each annotator flips his or her own coin to de-

[1] The different biases might not amount to much in the small doubly annotated subset, resulting in acceptable inter-annotator agreement; yet when enacted throughout a large number of instances they can be detrimental from a machine learner's perspective.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 280–287,
Suntec, Singapore, 2-7 August 2009.

cide on a label (each annotator can have a different "coin" reflecting his or her biases). For annotations generated under such a model, there is a danger of hard instances posing as easy – an observed agreement between annotators being a result of all coins coming up heads by chance. They therefore define the expected proportion of hard instances in agreed items as **annotation noise**. They provide an example from the literature where an annotation noise rate of about 15% is likely.

The question addressed in this article is: How problematic is learning from training data with annotation noise? Specifically, we are interested in estimating the degree to which performance on easy instances at test time can be hurt by the presence of hard instances in training data.

Definition 1 *The* **hard case bias***, τ, is the portion of easy instances in the test data that are misclassified as a result of hard instances in the training data.*

This article proceeds as follows. First, we show that a machine learner operating under a 0-1 loss minimization principle could sustain a hard case bias of $\theta(\frac{1}{\sqrt{N}})$ in the worst case. Thus, while annotation noise is hazardous for small datasets, it is better tolerated in larger ones. However, 0-1 loss minimization is computationally intractable for large datasets (Feldman et al., 2006; Guruswami and Raghavendra, 2006); substitute loss functions are often used in practice. While their tolerance to random classification noise is as good as for 0-1 loss, their tolerance to annotation noise is worse. For example, the perceptron family of algorithms handle random classification noise well (Cohen, 1997). We show in section 3.4 that the widely used Freund and Schapire (1999) voted perceptron algorithm could face a constant hard case bias when confronted with annotation noise in training data, irrespective of the size of the dataset. Finally, we discuss the implications of our findings for the practice of annotation studies and for data utilization in machine learning.

2 0-1 Loss

Let a sample be a sequence $x_1, \ldots, x_N$ drawn uniformly from the d-dimensional discrete cube $I_d = \{-1, 1\}^d$ with corresponding labels $y_1, \ldots, y_N \in \{-1, 1\}$. Suppose further that the learning algorithm operates by finding a hyperplane (w, ψ), $w \in \mathbb{R}^d, \psi \in \mathbb{R}$, that minimizes the empirical error $L(w, \psi) = \sum_{j=1\ldots N}[y_j - \text{sgn}(\sum_{i=1\ldots d} x_j^i w^i - \psi)]^2$. Let there be H hard cases, such that the annotation noise is $\gamma = \frac{H}{N}$.[2]

Theorem 1 *In the worst case configuration of instances a hard case bias of $\tau = \theta(\frac{1}{\sqrt{N}})$ cannot be ruled out with constant confidence.*

Idea of the proof: We prove by explicit construction of an adversarial case. Suppose there is a plane that perfectly separates the easy instances. The $\theta(N)$ hard instances will be concentrated in a band parallel to the separating plane, that is near enough to the plane so as to trap only about $\theta(\sqrt{N})$ easy instances between the plane and the band (see figure 1 for an illustration). For a random labeling of the hard instances, the central limit theorem shows there is positive probability that there would be an imbalance between $+1$ and -1 labels in favor of -1s on the scale of $\sqrt{N}$, which, with appropriate constants, would lead to the movement of the empirically minimal separation plane to the right of the hard case band, misclassifying the trapped easy cases.

Proof: Let $v = v(x) = \sum_{i=1\ldots d} x^i$ denote the sum of the coordinates of an instance in I_d and take $\lambda_e = \sqrt{d} \cdot F^{-1}(\sqrt{\gamma} \cdot 2^{-\frac{d}{2}} + \frac{1}{2})$ and $\lambda_h = \sqrt{d} \cdot F^{-1}(\gamma + \sqrt{\gamma} \cdot 2^{-\frac{d}{2}} + \frac{1}{2})$, where $F(t)$ is the cumulative distribution function of the normal distribution. Suppose further that instances x_j such that $\lambda_e < v_j < \lambda_h$ are all and only hard instances; their labels are coinflips. All other instances are easy, and labeled $y = y(x) = \text{sgn}(v)$. In this case, the hyperplane $\frac{1}{\sqrt{d}}(1 \ldots 1)$ is the true separation plane for the easy instances, with $\psi = 0$. Figure 1 shows this configuration.

According to the central limit theorem, for d, N large, the distribution of v is well approximated by $\mathcal{N}(0, \sqrt{d})$. If $N = c_1 \cdot 2^d$, for some $0 < c_1 < 4$, the second application of the central limit theorem ensures that, with high probability, about $\gamma N = c_1 \gamma 2^d$ items would fall between λ_e and λ_h (all hard), and $\sqrt{\gamma} \cdot 2^{-\frac{d}{2}} N = c_1 \sqrt{\gamma 2^d}$ would fall between 0 and λ_e (all easy, all labeled $+1$).

Let Z be the sum of labels of the hard cases, $Z = \sum_{i=1\ldots H} y_i$. Applying the central limit theorem a third time, for large N, Z will, with a high probability, be distributed approximately as

[2] In Beigman Klebanov and Beigman (2009), annotation noise is defined as percentage of hard instances in the agreed annotations; this implies noise measurement on multiply annotated material. When there is just one annotator, no distinction between easy vs hard instances can be made; in this sense, all hard instances are posing as easy.

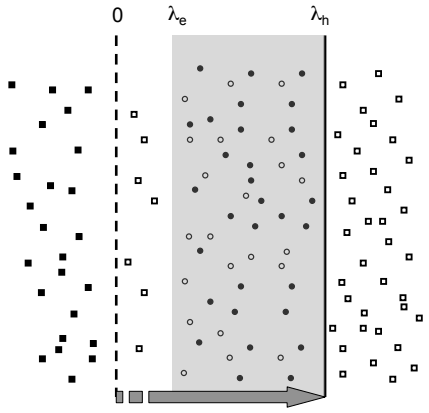


Figure 1: The adversarial case for 0-1 loss. Squares correspond to easy instances, circles – to hard ones. Filled squares and circles are labeled -1, empty ones are labeled $+1$.

$\mathcal{N}(0, \sqrt{\gamma N})$. This implies that a value as low as -2σ cannot be ruled out with high (say 95%) confidence. Thus, an imbalance of up to $2\sqrt{\gamma N}$, or of $2\sqrt{c_1\gamma 2^d}$, in favor of -1s is possible.

There are between 0 and λ_h about $2\sqrt{c}_1\sqrt{\gamma 2^d}$ more -1 hard instances than $+1$ hard instances, as opposed to $c_1\sqrt{\gamma 2^d}$ easy instances that are all $+1$. As long as $c_1 < 2\sqrt{c}_1$, i.e. $c_1 < 4$, the empirically minimal threshold would move to λ_h, resulting in a hard case bias of $\tau = \frac{\sqrt{\gamma}\sqrt{c_1 2^d}}{(1-\gamma)\cdot c_1 2^d} = \theta(\frac{1}{\sqrt{N}})$.

To see that this is the worst case scenario, we note that 0-1 loss sustained on $\theta(N)$ hard cases is the order of magnitude of the possible imbalance between -1 and $+1$ random labels, which is $\theta(\sqrt{N})$. For hard case loss to outweigh the loss on the misclassified easy instances, there cannot be more than $\theta(\sqrt{N})$ of the latter $\square$

Note that the proof requires that $N = \theta(2^d)$ namely, that asymptotically the sample includes a fixed portion of the instances. If the sample is asymptotically smaller, then λ_e will have to be adjusted such that $\lambda_e = \sqrt{d} \cdot F^{-1}(\theta(\frac{1}{\sqrt{N}}) + \frac{1}{2})$.

According to theorem 1, for a 10K dataset with 15% hard case rate, a hard case bias of about 1% cannot be ruled out with 95% confidence.

Theorem 1 suggests that annotation noise as defined here is qualitatively different from more malicious types of noise analyzed in the agnostic learning framework (Kearns and Li, 1988; Haussler, 1992; Kearns et al., 1994), where an adversary can not only choose the placement of the hard cases, but also their labels. In worst case, the 0-1 loss model would sustain a constant rate of error due to malicious noise, whereas annotation noise is tolerated quite well in large datasets.

3 Voted Perceptron

Freund and Schapire (1999) describe the *voted perceptron*. This algorithm and its many variants are widely used in the computational linguistics community (Collins, 2002a; Collins and Duffy, 2002; Collins, 2002b; Collins and Roark, 2004; Henderson and Titov, 2005; Viola and Narasimhan, 2005; Cohen et al., 2004; Carreras et al., 2005; Shen and Joshi, 2005; Ciaramita and Johnson, 2003). In this section, we show that the voted perceptron can be vulnerable to annotation noise. The algorithm is shown below.

Algorithm 1 Voted Perceptron

Training
Input: a labeled training set $(x_1, y_1), \ldots, (x_N, y_N)$
Output: a list of perceptrons $w_1, \ldots, w_N$

Initialize: $t \leftarrow 0$; $w_1 \leftarrow 0$; $\psi_1 \leftarrow 0$
for $t = 1 \ldots N$ **do**
  $\hat{y}_t \leftarrow \text{sign}(\langle w_t, x_t\rangle + \psi_t)$
  $w_{t+1} \leftarrow w_t + \frac{y_t - \hat{y}_t}{2} \cdot x_t$
  $\psi_{t+1} \leftarrow \psi_t + \frac{y_t - \hat{y}_t}{2} \cdot \langle w_t, x_t\rangle$
end for

Forecasting
Input: a list of perceptrons $w_1, \ldots, w_N$
  an unlabeled instance x
Output: A forecasted label y

$\hat{y} \leftarrow \sum_{t=1}^{N} \text{sign}(\langle w_t, x_t\rangle + \psi_t)$
$y \leftarrow \text{sign}(\hat{y})$

The voted perceptron algorithm is a refinement of the perceptron algorithm (Rosenblatt, 1962; Minsky and Papert, 1969). Perceptron is a dynamic algorithm; starting with an initial hyperplane w_0, it passes repeatedly through the labeled sample. Whenever an instance is misclassified by w_t, the hyperplane is modified to adapt to the instance. The algorithm terminates once it has passed through the sample without making any classification mistakes. The algorithm terminates iff the sample can be separated by a hyperplane, and in this case the algorithm finds a separating hyperplane. Novikoff (1962) gives a bound on the number of iterations the algorithm goes through before termination, when the sample is separable by a margin.

The perceptron algorithm is vulnerable to noise, as even a little noise could make the sample inseparable. In this case the algorithm would cycle indefinitely never meeting termination conditions, w_t would obtain values within a certain dynamic range but would not converge. In such setting, imposing a stopping time would be equivalent to drawing a random vector from the dynamic range.

Freund and Schapire (1999) extend the perceptron to inseparable samples with their voted perceptron algorithm and give theoretical generalization bounds for its performance. The basic idea underlying the algorithm is that if the dynamic range of the perceptron is not too large then w_t would classify most instances correctly most of the time (for most values of t). Thus, for a sample $x_1, \ldots, x_N$ the new algorithm would keep track of $w_0, \ldots, w_N$, and for an unlabeled instance x it would forecast the classification most prominent amongst these hyperplanes.

The bounds given by Freund and Schapire (1999) depend on the *hinge loss* of the dataset. In section 3.2 we construct a difficult setting for this algorithm. To prove that voted perceptron would suffer from a constant hard case bias in this setting using the exact dynamics of the perceptron is beyond the scope of this article. Instead, in section 3.3 we provide a lower bound on the hinge loss for a simplified model of the perceptron algorithm dynamics, which we argue would be a good approximation to the true dynamics in the setting we constructed. For this simplified model, we show that the hinge loss is large, and the bounds in Freund and Schapire (1999) cannot rule out a constant level of error regardless of the size of the dataset. In section 3.4 we study the dynamics of the model and prove that $\tau = \theta(1)$ for the adversarial setting.

3.1 Hinge Loss

Definition 2 *The* **hinge loss** *of a labeled instance (x, y) with respect to hyperplane (w, ψ) and margin $\delta > 0$ is given by $\zeta = \zeta(\psi, \delta) = \max(0, \delta - y \cdot (\langle w, x \rangle - \psi))$.*

ζ measures the distance of an instance from being classified correctly with a δ margin. Figure 2 shows examples of hinge loss for various data points.

Theorem 2 (Freund and Schapire (1999)) *After one pass on the sample, the probability that the voted perceptron algorithm does not predict correctly the label of a test instance x_{N+1} is bounded by $\frac{2}{N+1}\mathbb{E}_{N+1}\left[\frac{d+D}{\delta}\right]^2$ where $D = D(w, \psi, \delta) = \sqrt{\sum_{i=1}^{N} \zeta_i^2}$.*

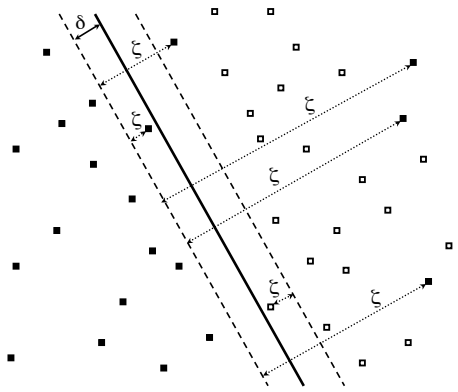


Figure 2: Hinge loss ζ for various data points incurred by the separator with margin δ.

This result is used to explain the convergence of weighted or voted perceptron algorithms (Collins, 2002a). It is useful as long as the expected value of D is not too large. We show that in an adversarial setting of the annotation noise D is large, hence these bounds are trivial.

3.2 Adversarial Annotation Noise

Let a sample be a sequence $x_1, \ldots, x_N$ drawn uniformly from I_d with $y_1, \ldots, y_N \in \{-1, 1\}$. Easy cases are labeled $y = y(x) = \text{sgn}(v)$ as before, with $v = v(x) = \sum_{i=1\ldots d} x^i$. The true separation plane for the easy instances is $w^* = \frac{1}{\sqrt{d}}(1 \ldots 1)$, $\psi^* = 0$. Suppose hard cases are those where $v(x) > c_1\sqrt{d}$, where c_1 is chosen so that the hard instances account for γN of all instances.[3] Figure 3 shows this setting.

3.3 Lower Bound on Hinge Loss

In the simplified case, we assume that the algorithm starts training with the hyperplane $w_0 = w^* = \frac{1}{\sqrt{d}}(1 \ldots 1)$, and keeps it throughout the training, only updating ψ. In reality, each hard instance can be decomposed into a component that is parallel to w^*, and a component that is orthogonal to it. The expected contribution of the orthogonal

[3] See the proof of 0-1 case for a similar construction using the central limit theorem.

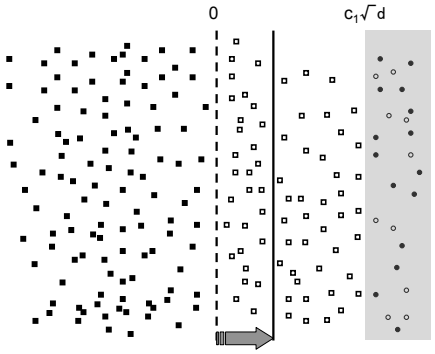


Figure 3: An adversarial case of annotation noise for the voted perceptron algorithm.

component to the algorithm's update will be positive due to the systematic positioning of the hard cases, while the contributions of the parallel components are expected to cancel out due to the symmetry of the hard cases around the main diagonal that is orthogonal to w^*. Thus, while w_t will not necessarily parallel w^*, it will be close to parallel for most $t > 0$. The simplified case is thus a good approximation of the real case, and the bound we obtain is expected to hold for the real case as well.

For any initial value $\psi_0 < 0$ all misclassified instances are labeled -1 and classified as $+1$, hence the update will increase ψ_0, and reach 0 soon enough. We can therefore assume that $\psi_t \geq 0$ for any $t > t_0$ where $t_0 \ll N$.

Lemma 3 *For any $t > t_0$, there exist $\alpha = \alpha(\gamma, T) > 0$ such that $\mathbb{E}(\zeta^2) \geq \alpha \cdot \delta$.*

Proof: For $\psi \geq 0$ there are two main sources of hinge loss: easy $+1$ instances that are classified as -1, and hard -1 instances classified as $+1$. These correspond to the two components of the following sum (the inequality is due to disregarding the loss incurred by a correct classification with too wide a margin):

$$\begin{aligned}\mathbb{E}(\zeta^2) \quad \geq \quad & \sum_{l=0}^{[\psi]} \frac{1}{2^d}\binom{d}{l}\left(\frac{\psi}{\sqrt{d}} - \frac{l}{\sqrt{d}} + \delta\right)^2 \\ & + \frac{1}{2}\sum_{l=c_1\sqrt{d}}^{d} \frac{1}{2^d}\binom{d}{l}\left(\frac{l}{\sqrt{d}} - \frac{\psi}{\sqrt{d}} + \delta\right)^2\end{aligned}$$

Let $0 < T < c_1$ be a parameter. For $\psi > T\sqrt{d}$, misclassified easy instances dominate the loss:

$$\begin{aligned}\mathbb{E}(\zeta^2) \quad \geq \quad & \sum_{l=0}^{[\psi]} \frac{1}{2^d}\binom{d}{l}\left(\frac{\psi}{\sqrt{d}} - \frac{l}{\sqrt{d}} + \delta\right)^2 \\ \geq \quad & \sum_{l=0}^{[T\sqrt{d}]} \frac{1}{2^d}\binom{d}{l}\left(\frac{T\sqrt{d}}{\sqrt{d}} - \frac{l}{\sqrt{d}} + \delta\right)^2 \\ \geq \quad & \sum_{l=0}^{T\sqrt{d}} \frac{1}{2^d}\binom{d}{l}\left(T - \frac{l}{\sqrt{d}} + \delta\right)^2 \\ \geq \quad & \frac{1}{\sqrt{2\pi}}\int_0^T (T + \delta - t)^2 e^{-t^2/2} dt = H_T(\delta)\end{aligned}$$

The last inequality follows from a normal approximation of the binomial distribution (see, for example, Feller (1968)).

For $0 \leq \psi \leq T\sqrt{d}$, misclassified hard cases dominate:

$$\begin{aligned}\mathbb{E}(\zeta^2) \quad \geq \quad & \frac{1}{2}\sum_{l=c_1\sqrt{d}}^{d} \frac{1}{2^d}\binom{d}{l}\left(\frac{l}{\sqrt{d}} - \frac{\psi}{\sqrt{d}} + \delta\right)^2 \\ \geq \quad & \frac{1}{2}\sum_{l=c_1\sqrt{d}}^{d} \frac{1}{2^d}\binom{d}{l}\left(\frac{l}{\sqrt{d}} - \frac{T\sqrt{d}}{\sqrt{d}} + \delta\right)^2 \\ \geq \quad & \frac{1}{2}\cdot\frac{1}{\sqrt{2\pi}}\int_{\Phi^{-1}(\gamma)}^{\infty} (t - T + \delta)^2 e^{-t^2/2} dt \\ = \quad & H_\gamma(\delta)\end{aligned}$$

where $\Phi^{-1}(\gamma)$ is the inverse of the normal distribution density.

Thus $\mathbb{E}(\zeta^2) \geq \min\{H_T(\delta), H_\gamma(\delta)\}$, and there exists $\alpha = \alpha(\gamma, T) > 0$ such that $\min\{H_T(\delta), H_\gamma(\delta)\} \geq \alpha \cdot \delta$ □

Corollary 4 *The bound in theorem 2 does not converge to zero for large N.*

We recall that Freund and Schapire (1999) bound is proportional to $D^2 = \sum_{i=1}^{N} \zeta_i^2$. It follows from lemma 3 that $D^2 = \theta(N)$, hence the bound is ineffective.

3.4 Lower Bound on τ for Voted Perceptron Under Simplified Dynamics

Corollary 4 does not give an estimate on the hard case bias. Indeed, it could be that $w_t = w^*$ for almost every t. There would still be significant hinge in this case, but the hard case bias for the voted forecast would be zero. To assess the hard case bias we need a model of perceptron dynamics that would account for the history of hyperplanes $w_0, \ldots, w_N$ the perceptron goes through on

a sample $x_1, \ldots, x_N$. The key simplification in our model is assuming that w_t parallels w^* for all t, hence the next hyperplane depends only on the offset ψ_t. This is a one dimensional Markov random walk governed by the distribution

$$\mathbb{P}(\psi_{t+1} - \psi_t = r | \psi_t) = \mathbb{P}(x | \frac{y_t - \hat{y}_t}{2} \cdot \langle w^*, x \rangle = r)$$

In general $-d \leq \psi_t \leq d$ but as mentioned before lemma 3, we may assume $\psi_t > 0$.

Lemma 5 *There exists $c > 0$ such that with a high probability $\psi_t > c \cdot \sqrt{d}$ for most $0 \leq t \leq N$.*

Proof: Let $c_0 = F^{-1}(\frac{\gamma}{2} + \frac{1}{2})$; $c_1 = F^{-1}(1-\gamma)$. We designate the intervals $I_0 = [0, c_0 \cdot \sqrt{d}]$; $I_1 = [c_0 \cdot \sqrt{d}, c_1 \cdot \sqrt{d}]$ and $I_2 = [c_1 \cdot \sqrt{d}, d]$ and define $A_i = \{x : v(x) \in I_i\}$ for $i = 0, 1, 2$. Note that the constants c_0 and c_1 are chosen so that $\mathbb{P}(A_0) = \frac{\gamma}{2}$ and $\mathbb{P}(A_2) = \gamma$. It follows from the construction in section 3.2 that A_0 and A_1 are easy instances and A_2 are hard. Given a sample $x_1, \ldots, x_N$, a misclassification of $x_t \in A_0$ by ψ_t could only happen when an easy $+1$ instance is classified as -1. Thus the algorithm would shift ψ_t to the left by no more than $|v_t - \psi_t|$ since $v_t = \langle w^*, x_t \rangle$. This shows that $\psi_t \in I_0$ implies $\psi_{t+1} \in I_0$. In the same manner, it is easy to verify that if $\psi_t \in I_j$ and $x_t \in A_k$ then $\psi_{t+1} \in I_k$, unless $j = 0$ and $k = 1$, in which case $\psi_{t+1} \in I_0$ because $x_t \in A_1$ would be classified correctly by $\psi_t \in I_0$.

We construct a Markov chain with three states $a_0 = 0$, $a_1 = c_0 \cdot \sqrt{d}$ and $a_2 = c_1 \cdot \sqrt{d}$ governed by the following transition distribution:

$$\begin{pmatrix} 1 - \frac{\gamma}{2} & 0 & \frac{\gamma}{2} \\ \frac{\gamma}{2} & 1 - \gamma & \frac{\gamma}{2} \\ \frac{\gamma}{2} & \frac{1}{2} - \frac{3\gamma}{2} & \frac{1}{2} + \gamma \end{pmatrix}$$

Let X_t be the state at time t. The principal eigenvector of the transition matrix $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ gives the stationary probability distribution of X_t. Thus $X_t \in \{a_1, a_2\}$ with probability $\frac{2}{3}$. Since the transition distribution of X_t mirrors that of ψ_t, and since a_j are at the leftmost borders of I_j, respectively, it follows that $X_t \leq \psi_t$ for all t, thus $X_t \in \{a_1, a_2\}$ implies $\psi_t \in I_1 \cup I_2$. It follows that $\psi_t > c_0 \cdot \sqrt{d}$ with probability $\frac{2}{3}$, and the lemma follows from the law of large numbers □

Corollary 6 *With high probability $\tau = \theta(1)$.*

Proof: Lemma 5 shows that for a sample $x_1, \ldots, x_N$ with high probability ψ_t is most of the time to the right of $c \cdot \sqrt{d}$. Consequently for any x in the band $0 \leq v \leq c \cdot \sqrt{d}$ we get $\text{sign}(\langle w^*, x \rangle + \psi_t) = -1$ for most t hence by definition, the voted perceptron would classify such an instance as -1, although it is in fact a $+1$ easy instance. Since there are $\theta(N)$ misclassified easy instances, $\tau = \theta(1)$ □

4 Discussion

In this article we show that training with annotation noise can be detrimental for test-time results on easy, uncontroversial instances; we termed this phenomenon *hard case bias*. Although under the 0-1 loss model annotation noise can be tolerated for larger datasets (theorem 1), minimizing such loss becomes intractable for larger datasets. Freund and Schapire (1999) voted perceptron algorithm and its variants are widely used in computational linguistics practice; our results show that it could suffer a constant rate of hard case bias irrespective of the size of the dataset (section 3.4).

How can hard case bias be reduced? One possibility is removing as many hard cases as one can not only from the test data, as suggested in Beigman Klebanov and Beigman (2009), but from the training data as well. Adding the second annotator is expected to detect about half the hard cases, as they would surface as disagreements between the annotators. Subsequently, a machine learner can be told to ignore those cases during training, reducing the risk of hard case bias. While this is certainly a daunting task, it is possible that for annotation studies that do not require expert annotators and extensive annotator training, the newly available access to a large pool of inexpensive annotators, such as the Amazon Mechanical Turk scheme (Snow et al., 2008),[4] or embedding the task in an online game played by volunteers (Poesio et al., 2008; von Ahn, 2006) could provide some solutions.

Reidsma and op den Akker (2008) suggest a different option. When non-overlapping parts of the dataset are annotated by different annotators, each classifier can be trained to reflect the opinion (albeit biased) of a specific annotator, using different parts of the datasets. Such "subjective machines" can be applied to a new set of data; an item that causes disagreement between classifiers is then extrapolated to be a case of potential disagreement between the humans they replicate, i.e.

[4] http://aws.amazon.com/mturk/

a hard case. Our results suggest that, regardless of the success of such an extrapolation scheme in detecting hard cases, it could erroneously invalidate easy cases: Each classifier would presumably suffer from a certain hard case bias, i.e. classify incorrectly things that are in fact uncontroversial for any human annotator. If each such classifier has a different hard case bias, some inter-classifier disagreements would occur on easy cases. Depending on the distribution of those easy cases in the feature space, this could invalidate valuable cases. If the situation depicted in figure 1 corresponds to the pattern learned by one of the classifiers, it would lead to marking the easy cases closest to the real separation boundary (those between 0 and λ_e) as hard, and hence unsuitable for learning, eliminating the most informative material from the training data.

Reidsma and Carletta (2008) recently showed by simulation that different types of annotator behavior have different impact on the outcomes of machine learning from the annotated data. Our results provide a theoretical analysis that points in the same direction: While random classification noise is tolerable, other types of noise – such as annotation noise handled here – are more problematic. It is therefore important to develop models of annotator behavior and of the resulting imperfections of the annotated datasets, in order to diagnose the potential learning problem and suggest mitigation strategies.

References

Dana Angluin and Philip Laird. 1988. Learning from Noisy Examples. *Machine Learning*, 2(4):343–370.

Beata Beigman Klebanov and Eyal Beigman. 2009. From Annotator Agreement to Noise Models. *Computational Linguistics, accepted for publication*.

Beata Beigman Klebanov, Eyal Beigman, and Daniel Diermeier. 2008. Analyzing Disagreements. In *COLING 2008 Workshop on Human Judgments in Computational Linguistics*, pages 2–7, Manchester, UK.

Avrim Blum, Alan Frieze, Ravi Kannan, and Santosh Vempala. 1996. A Polynomial-Time Algorithm for Learning Noisy Linear Threshold Functions. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 330–338, Burlington, Vermont, USA.

Xavier Carreras, Llúis Màrquez, and Jorge Castro. 2005. Filtering-Ranking Perceptron Learning for Partial Parsing. *Machine Learning*, 60(1):41–71.

Massimiliano Ciaramita and Mark Johnson. 2003. Supersense Tagging of Unknown Nouns in WordNet. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, pages 168–175, Sapporo, Japan.

William Cohen, Vitor Carvalho, and Tom Mitchell. 2004. Learning to Classify Email into "Speech Acts". In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, pages 309–316, Barcelona, Spain.

Edith Cohen. 1997. Learning Noisy Perceptrons by a Perceptron in Polynomial Time. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 514–523, Miami Beach, Florida, USA.

Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 263–370, Philadelphia, USA.

Michael Collins and Brian Roark. 2004. Incremental Parsing with the Perceptron Algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 111–118, Barcelona, Spain.

Michael Collins. 2002a. Discriminative Training Methods for Hidden Markov Hodels: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, pages 1–8, Philadelphia, USA.

Michael Collins. 2002b. Ranking Algorithms for Named Entity Extraction: Boosting and the Voted Perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 489–496, Philadelphia, USA.

Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Ponnuswami. 2006. New Results for Learning Noisy Parities and Halfspaces. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 563–574, Los Alamitos, CA, USA.

William Feller. 1968. *An Introduction to Probability Theory and Its Application*, volume 1. Wiley, New York, 3rd edition.

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

Venkatesan Guruswami and Prasad Raghavendra. 2006. Hardness of Learning Halfspaces with Noise. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 543–552, Los Alamitos, CA, USA.

David Haussler. 1992. Decision Theoretic Generalizations of the PAC Model for Neural Net and other Learning Applications. *Information and Computation*, 100(1):78–150.

James Henderson and Ivan Titov. 2005. Data-Defined Kernels for Parse Reranking Derived from Probabilistic Models. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 181–188, Ann Arbor, Michigan, USA.

Michael Kearns and Ming Li. 1988. Learning in the Presence of Malicious Errors. In *Proceedings of the 20th Annual ACM symposium on Theory of Computing*, pages 267–280, Chicago, USA.

Michael Kearns, Robert Schapire, and Linda Sellie. 1994. Toward Efficient Agnostic Learning. *Machine Learning*, 17(2):115–141.

Michael Kearns. 1993. Efficient Noise-Tolerant Learning from Statistical Queries. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 392–401, San Diego, CA, USA.

Marvin Minsky and Seymour Papert. 1969. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, Mass.

A. B. Novikoff. 1962. On convergence proofs on perceptrons. *Symposium on the Mathematical Theory of Automata*, 12:615–622.

Miles Osborne. 2002. Shallow Parsing Using Noisy and Non-Stationary Training Material. *Journal of Machine Learning Research*, 2:695–719.

Massimo Poesio, Udo Kruschwitz, and Chamberlain Jon. 2008. ANAWIKI: Creating Anaphorically Annotated Resources through Web Cooperation. In *Proceedings of the 6th International Language Resources and Evaluation Conference*, Marrakech, Morocco.

Dennis Reidsma and Jean Carletta. 2008. Reliability measurement without limit. *Computational Linguistics*, 34(3):319–326.

Dennis Reidsma and Rieks op den Akker. 2008. Exploiting Subjective Annotations. In *COLING 2008 Workshop on Human Judgments in Computational Linguistics*, pages 8–16, Manchester, UK.

Frank Rosenblatt. 1962. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington, D.C.

Libin Shen and Aravind Joshi. 2005. Incremental LTAG Parsing. In *Proceedings of the Human Language Technology Conference and Empirical Methods in Natural Language Processing Conference*, pages 811–818, Vancouver, British Columbia, Canada.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and Fast – But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, pages 254–263, Honolulu, Hawaii.

Paul Viola and Mukund Narasimhan. 2005. Learning to Extract Information from Semi-Structured Text Using a Discriminative Context Free Grammar. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 330–337, Salvador, Brazil.

Luis von Ahn. 2006. Games with a purpose. *Computer*, 39(6):92–94.

Abstraction and Generalisation in Semantic Role Labels: PropBank, VerbNet or both?

Paola Merlo
Linguistics Department
University of Geneva
5 Rue de Candolle, 1204 Geneva
Switzerland
Paola.Merlo@unige.ch

Lonneke Van Der Plas
Linguistics Department
University of Geneva
5 Rue de Candolle, 1204 Geneva
Switzerland
Lonneke.VanDerPlas@unige.ch

Abstract

Semantic role labels are the representation of the grammatically relevant aspects of a sentence meaning. Capturing the nature and the number of semantic roles in a sentence is therefore fundamental to correctly describing the interface between grammar and meaning. In this paper, we compare two annotation schemes, PropBank and VerbNet, in a task-independent, general way, analysing how well they fare in capturing the linguistic generalisations that are known to hold for semantic role labels, and consequently how well they grammaticalise aspects of meaning. We show that VerbNet is more verb-specific and better able to generalise to new semantic role instances, while PropBank better captures some of the structural constraints among roles. We conclude that these two resources should be used together, as they are complementary.

1 Introduction

Most current approaches to language analysis assume that the structure of a sentence depends on the lexical semantics of the verb and of other predicates in the sentence. It is also assumed that only certain aspects of a sentence meaning are grammaticalised. Semantic role labels are the representation of the grammatically relevant aspects of a sentence meaning.

Capturing the nature and the number of semantic roles in a sentence is therefore fundamental to correctly describe the interface between grammar and meaning, and it is of paramount importance for all natural language processing (NLP) applications that attempt to extract meaning representations from analysed text, such as question-answering systems or even machine translation.

The role of theories of semantic role lists is to obtain a set of semantic roles that can apply to any argument of any verb, to provide an unambiguous identifier of the grammatical roles of the participants in the event described by the sentence (Dowty, 1991). Starting from the first proposals (Gruber, 1965; Fillmore, 1968; Jackendoff, 1972), several approaches have been put forth, ranging from a combination of very few roles to lists of very fine-grained specificity. (See Levin and Rappaport Hovav (2005) for an exhaustive review).

In NLP, several proposals have been put forth in recent years and adopted in the annotation of large samples of text (Baker et al., 1998; Palmer et al., 2005; Kipper, 2005; Loper et al., 2007). The annotated PropBank corpus, and therefore implicitly its role labels inventory, has been largely adopted in NLP because of its exhaustiveness and because it is coupled with syntactic annotation, properties that make it very attractive for the automatic learning of these roles and their further applications to NLP tasks. However, the labelling choices made by PropBank have recently come under scrutiny (Zapirain et al., 2008; Loper et al., 2007; Yi et al., 2007).

The annotation of PropBank labels has been conceived in a two-tiered fashion. A first tier assigns abstract labels such as ARG0 or ARG1, while a separate annotation records the second-tier, verb-sense specific meaning of these labels. Labels ARG0 or ARG1 are assigned to the most prominent argument in the sentence (ARG1 for unaccusative verbs and ARG0 for all other verbs). The other labels are assigned in the order of prominence. So, while the same high-level labels are used across verbs, they could have different meanings for different verb senses. Researchers have usually concentrated on the high-level annotation, but as indicated in Yi et al. (2007), there is reason to think that these labels do not generalise across verbs, nor to unseen verbs or to novel verb

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 288–296,
Suntec, Singapore, 2-7 August 2009.

senses. Because the meaning of the role annotation is verb-specific, there is also reason to think that it fragments the data and creates data sparseness, making automatic learning from examples more difficult. These short-comings are more apparent in the annotation of less prominent and less frequent roles, marked by the ARG2 to ARG5 labels.

Zapirain et al. (2008), Loper et al. (2007) and Yi et al. (2007) investigated the ability of the PropBank role inventory to generalise compared to the annotation in another semantic role list, proposed in the electronic dictionary VerbNet. VerbNet labels are assigned in a verb-class specific way and have been devised to be more similar to the inventories of thematic role lists usually proposed by linguists. The results in these papers are conflicting.

While Loper et al. (2007) and Yi et al. (2007) show that augmenting PropBank labels with VerbNet labels increases generalisation of the less frequent labels, such as ARG2, to new verbs and new domains, they also show that PropBank labels perform better overall, in a semantic role labelling task. Confirming this latter result, Zapirain et al. (2008) find that PropBank role labels are more robust than VerbNet labels in predicting new verb usages, unseen verbs, and they port better to new domains.

The apparent contradiction of these results can be due to several confounding factors in the experiments. First, the argument labels for which the VerbNet improvement was found are infrequent, and might therefore not have influenced the overall results enough to counterbalance new errors introduced by the finer-grained annotation scheme; second, the learning methods in both these experimental settings are largely based on syntactic information, thereby confounding learning and generalisation due to syntax — which would favour the more syntactically-driven PropBank annotation — with learning due to greater generality of the semantic role annotation; finally, task-specific learning-based experiments do not guarantee that the learners be sufficiently powerful to make use of the full generality of the semantic role labels.

In this paper, we compare the two annotation schemes, analysing how well they fare in capturing the linguistic generalisations that are known to hold for semantic role labels, and consequently how well they grammaticalise aspects of meaning. Because the well-attested strong correlation between syntactic structure and semantic role labels (Levin and Rappaport Hovav, 2005; Merlo and Stevenson, 2001) could intervene as a confounding factor in this analysis, we expressly limit our investigation to data analyses and statistical measures that do not exploit syntactic properties or parsing techniques. The conclusions reached this way are not task-specific and are therefore widely applicable.

To preview, based on results in section 3, we conclude that PropBank is easier to learn, but VerbNet is more informative in general, it generalises better to new role instances and its labels are more strongly correlated to specific verbs. In section 4, we show that VerbNet labels provide finer-grained specificity. PropBank labels are more concentrated on a few VerbNet labels at higher frequency. This is not true at low frequency, where VerbNet provides disambiguations to overloaded PropBank variables. Practically, these two sets of results indicate that both annotation schemes could be useful in different circumstances, and at different frequency bands. In section 5, we report results indicating that PropBank role sets are high-level abstractions of VerbNet role sets and that VerbNet role sets are more verb and class-specific. In section 6, we show that PropBank more closely captures the thematic hierarchy and is more correlated to grammatical functions, hence potentially more useful for semantic role labelling, for learners whose features are based on the syntactic tree. Finally, in section 7, we summarise some previous results, and we provide new statistical evidence to argue that VerbNet labels are more general across verbs. These conclusions are reached by task-independent statistical analyses. The data and the measures used to reach these conclusions are discussed in the next section.

2 Materials and Method

In data analysis and inferential statistics, careful preparation of the data and choice of the appropriate statistical measures are key. We illustrate the data and the measures used here.

2.1 Data and Semantic Role Annotation

Proposition Bank (Palmer et al., 2005) adds Levin's style predicate-argument annotation and indication of verbs' alternations to the syntactic structures of the Penn Treebank (Marcus et al.,

1993).

It defines a limited role typology. Roles are specified for each verb individually. Verbal predicates in the Penn Treebank (PTB) receive a label REL and their arguments are annotated with abstract semantic role labels A0-A5 or AA for those complements of the predicative verb that are considered arguments, while those complements of the verb labelled with a semantic functional label in the original PTB receive the composite semantic role label AM-X, where X stands for labels such as LOC, TMP or ADV, for locative, temporal and adverbial modifiers respectively. PropBank uses two levels of granularity in its annotation, at least conceptually. Arguments receiving labels A0-A5 or AA do not express consistent semantic roles and are specific to a verb, while arguments receiving an AM-X label are supposed to be adjuncts and the respective roles they express are consistent across all verbs. However, among argument labels, A0 and A1 are assigned attempting to capture Proto-Agent and Proto-Patient properties (Dowty, 1991). They are, therefore, more valid across verbs and verb instances than the A2-A5 labels. Numerical results in Yi et al. (2007) show that 85% of A0 occurrences translate into Agent roles and more than 45% instances of A1 map into Patient and Patient-like roles, using a VerbNet labelling scheme. This is also confirmed by our counts, as illustrated in Tables 3 and 4 and discussed in Section 4 below.

VerbNet is a lexical resource for English verbs, yielding argumental and thematic information (Kipper, 2005). VerbNet resembles WordNet in spirit, it provides a verbal lexicon tying verbal semantics (theta-roles and selectional restrictions) to verbal distributional syntax. VerbNet defines 23 thematic roles that are valid across verbs. The list of thematic roles can be seen in the first column of Table 4.

For some of our comparisons below to be valid, we will need to reduce the inventory of labels of VerbNet to the same number of labels in PropBank. Following previous work (Loper et al., 2007), we define equivalence classes of VerbNet labels. We will refer to these classes as VerbNet *groups*. The groups we define are illustrated in Figure 1. Notice also that all our comparisons, like previous work, will be limited to the obligatory arguments in PropBank, the A0 to A5, AA arguments, to be comparable to VerbNet. VerbNet is a lexicon and by definition it does not list optional modifiers (the arguments labelled AM-X in PropBank).

In order to support the joint use of both these resources and their comparison, SemLink has been developed (Loper et al., 2007). SemLink[1] provides mappings from PropBank to VerbNet for the WSJ portion of the Penn Treebank. The mapping have been annotated automatically by a two-stage process: a lexical mapping and an instance classifier (Loper et al., 2007). The results were hand-corrected. In addition to semantic roles for both PropBank and VerbNet, SemLink contains information about verbs, their senses and their VerbNet classes which are extensions of Levin's classes.

The annotations in SemLink 1.1. are not complete. In the analyses presented here, we have only considered occurrences of semantic roles for which both a PropBank and a VerbNet label is available in the data (roughly 45% of the PropBank semantic roles have a VerbNet semantic role).[2] Furthermore, we perform our analyses on training and development data only. This means that we left section 23 of the Wall Street Journal out. The analyses are done on the basis of 106,459 semantic role pairs.

For the analysis concerning the correlation between semantic roles and syntactic dependencies in Section 6, we merged the SemLink data with the non-projectivised gold data of the CoNNL 2008 shared task on syntactic and semantic dependency parsing (Surdeanu et al., 2008). Only those dependencies that bear both a syntactic and a semantic label have been counted for test and development set. We have discarded discontinous arguments. Analyses are based on 68,268 dependencies in total.

2.2 Measures

In the following sections, we will use simple proportions, entropy, joint entropy, conditional entropy, mutual information, and a normalised form of mutual information which measures correlation between nominal attributes called symmetric uncertainty (Witten and Frank, 2005, 291). These are all widely used measures (Manning and Schuetze, 1999), excepted perhaps the last one. We briefly describe it here.

[1] (`http://verbs.colorado.edu/semlink/`)

[2] In some cases SemLink allows for multiple annotations. In those cases we selected the first annotation.

AGENT: Agent, Agent1
PATIENT: Patient
GOAL: Recipient, Destination, Location, Source, Material, Beneficiary, Goal
EXTENT: Extent, Asset, Value
PREDATTR: Predicate, Attribute, Theme, Theme1, Theme2, Topic, Stimulus, Proposition
PRODUCT: Patient2, Product, Patient1
INSTRCAUSE: Instrument, Cause, Experiencer, Actor2, Actor, Actor1

Figure 1: VerbNet Groups

Given a random variable X, the entropy H(X) describes our uncertainty about the value of X, and hence it quantifies the information contained in a message trasmitted by this variable. Given two random variables X,Y, the joint entropy H(X,Y) describes our uncertainty about the value of the pair (X,Y). *Symmetric uncertainty* is a normalised measure of the information redundancy between the distributions of two random variables. It calculates the ratio between the joint entropy of the two random variables if they are not independent and the joint entropy if the two random variables were independent (which is the sum of their individual entropies). This measure is calculated as follows.

$$U(A,B) = 2\frac{H(A) + H(B) - H(A,B)}{H(A) + H(B)}$$

where $H(X) = -\Sigma_{x \in X}\ p(x)logp(x)$ and $H(X,Y) = -\Sigma_{x \in X, y \in Y}\ p(x,y)logp(x,y)$.

Symmetric uncertainty lies between 0 and 1. A higher value for symmetric uncertainty indicates that the two random variables are more highly associated (more redundant), while lower values indicate that the two random variables approach independence.

We use these measures to evaluate how well two semantic role inventories capture well-known distributional generalisations. We discuss several of these generalisations in the following sections.

3 Amount of Information in Semantic Roles Inventory

Most proposals of semantic role inventories agree on the fact that the number of roles should be small to be valid generally. [3]

[3] With the notable exception of FrameNet, which is developing a large number of labels organised hierarchically and interpreted frame-specifically (Ruppenhofer et al., 2006).

Task	PropBank ERR	VerbNet ERR
Role generalisation	62 (82−52/48)	66 (77−33/67)
No verbal features	48 (76−52/48)	43 (58−33/67)
Unseen predicates	50 (75−52/48)	37 (62−33/67)

Table 2: Percent Error rate reduction (ERR) across role labelling sets in three tasks in Zapirain et al. (2008). ERR= (result − baseline / 100% − baseline)

PropBank and VerbNet clearly differ in the level of granularity of the semantic roles that have been assigned to the arguments. PropBank makes fewer distinctions than VerbNet, with 7 core argument labels compared to VerbNet's 23. More important than the size of the inventory, however, is the fact that PropBank has a much more skewed distribution than VerbNet, illustrated in Table 1. Consequently, the distribution of PropBank labels has an entropy of 1.37 bits, and even when the VerbNet labels are reduced to 7 equivalence classes the distribution has an entropy of 2.06 bits. VerbNet therefore conveys more information, but it is also more difficult to learn, as it is more uncertain. An uninformed PropBank learner that simply assigned the most frequent label would be correct 52% of the times by always assigning an A1 label, while for VerbNet would be correct only 33% of the times assigning Agent.

This simple fact might cast new light on some of the comparative conclusions of previous work. In some interesting experiments, Zapirain et al. (2008) test generalising abilities of VerbNet and PropBank comparatively to new role instances in general (their Table 1, line CoNLL setting, column F1 core), and also on unknown verbs and in the absence of verbal features. They find that a learner based on VerbNet has worse learning performance. They interpret this result as indicating that VerbNet labels are less general and more dependent on knowledge of specific verbs. However, a comparison that takes into consideration the differential baseline is able to factor the difficulty of the task out of the results for the overall performance. A simple baseline for a classifier is based on a majority class assignment (see our Table 1). We use the performance results reported in Zapirain et al. (2008) and calculate the reduction in error rate based on this differential baseline for the two annotation schemes. We compare only the results for the core labels in PropBank as those

PropBank		VerbNet									
A0	38.8	Agent	32.8	Cause	1.9	Source	0.9	Asset	0.3	Goal	0.00
A1	51.7	Theme	26.3	Product	1.6	Actor1	0.8	Material	0.2	Agent1	0.00
A2	9.0	Topic	11.5	Extent	1.3	Theme2	0.8	Beneficiary	0.2		
A3	0.5	Patient	5.8	Destination	1.2	Theme1	0.8	Proposition	0.1		
A4	0.0	Experiencer	4.2	Patient1	1.2	Attribute	0.7	Value	0.1		
A5	0.0	Predicate	2.3	Location	1.0	Patient2	0.5	Instrument	0.1		
AA	0.0	Recipient	2.2	Stimulus	0.9	Actor2	0.3	Actor	0.0		

Table 1: Distribution of PropBank core labels and VerbNet labels.

are the ones that correspond to VerbNet.[4] We find more mixed results than previously reported. VerbNet has better role generalising ability overall as its reduction in error rate is greater than PropBank (first line of Table 2), but it is more degraded by lack of verb information (second and third lines of Table 2). The importance of verb information for VerbNet is confirmed by information-theoretic measures. While the entropy of VerbNet labels is higher than that of PropBank labels (2.06 bits vs. 1.37 bits), as seen before, the conditional entropy of respective PropBank and VerbNet distributions given the verb is very similar, but higher for PropBank (1.11 vs 1.03 bits), thereby indicating that the verb provides much more information in association with VerbNet labels. The mutual information of the PropBank labels and the verbs is only 0.26 bits, while it is 1.03 bits for VerbNet. These results are expected if we recall the two-tiered logic that inspired PropBank annotation, where the abstract labels are less related to verbs than labels in VerbNet.

These results lead us to our first conclusion: while PropBank is easier to learn, VerbNet is more informative in general, it generalises better to new role instances, and its labels are more strongly correlated to specific verbs. It is therefore advisable to use both annotations: VerbNet labels if the verb is available, reverting to PropBank labels if no lexical information is known.

4 Equivalence Classes of Semantic Roles

An observation that holds for all semantic role labelling schemes is that certain labels seem to be more similar than others, based on their ability to occur in the same syntactic environment and to be expressed by the same function words. For example, Agent and Instrumental Cause are often subjects (of verbs selecting animate and inanimate subjects respectively); Patients/Themes can be direct objects of transitive verbs and subjects of change of state verbs; Goal and Beneficiary can be passivised and undergo the dative alternation; Instrument and Comitative are expressed by the same preposition in many languages (see Levin and Rappaport Hovav (2005).) However, most annotation schemes in NLP and linguistics assume that semantic role labels are atomic. It is therefore hard to explain why labels do not appear to be equidistant in meaning, but rather to form equivalence classes in certain contexts. [5]

While both role inventories under scrutiny here use atomic labels, their joint distribution shows interesting relations. The proportion counts are shown in Table 3 and 4.

If we read these tables column-wise, thereby taking the more linguistically-inspired labels in VerbNet to be the reference labels, we observe that the labels in PropBank are especially concentrated on those labels that linguistically would be considered similar. Specifically, in Table 3 A0 mostly groups together Agents and Instrumental Causes; A1 mostly refers to Themes and Patients; while A2 refers to Goals and Themes. If we

[4] We assume that our majority class can roughly correspond to Zapirain et al. (2008)'s data. Notice however that both sampling methods used to collect the counts are likely to slightly overestimate frequent labels. Zapirain et al. (2008) sample only complete propositions. It is reasonable to assume that higher numbered PropBank roles (A3, A4, A5) are more difficult to define. It would therefore more often happen that these labels are not annotated than it happens that A0, A1, A2, the frequent labels, are not annotated. This reasoning is confirmed by counts on our corpus, which indicate that incomplete propositions include a higher proportion of low frequency labels and a lower proportion of high frequency labels that the overall distribution. However, our method is also likely to overestimate frequent labels, since we count all labels, even those in incomplete propositions. By the same reasoning, we will find more frequent labels than the underlying real distribution of a complete annotation.

[5] Clearly, VerbNet annotators recognise the need to express these similarities since they use variants of the same label in many cases. Because the labels are atomic however, the distance between Agent and Patient is the same as Patient and Patient1 and the intended greater similarity of certain labels is lost to a learning device. As discussed at length in the linguistic literature, features bundles instead of atomic labels would be the mechanism to capture the differential distance of labels in the inventory (Levin and Rappaport Hovav, 2005).

	A0	A1	A2	A3	A4	A5	AA
Agent	32.6	0.2	-	-	-	-	-
Patient	0.0	5.8	-	-	-	-	-
Goal	0.0	1.5	4.0	0.2	0.0	0.0	-
Extent	-	0.2	1.3	0.2	-	-	-
PredAttr	1.2	39.3	2.9	0.0	-	-	0.0
Product	0.1	2.7	0.6	-	0.0	-	-
InstrCause	4.8	2.2	0.3	0.1	-	-	-

Table 3: Distribution of PropBank by VerbNet group labels according to SemLink. Counts indicated as 0.0 approximate zero by rounding, while a - sign indicates that no occurrences were found.

read these tables row-wise, thereby concentrating on the grouping of PropBank labels provided by VerbNet labels, we see that low frequency PropBank labels are more evenly spread across VerbNet labels than the frequent labels, and it is more difficult to identify a dominant label than for high-frequency labels. Because PropBank groups together VerbNet labels at high frequency, while VerbNet labels make different distinctions at lower frequencies, the distribution of PropBank is much more skewed than VerbNet, yielding the differences in distributions and entropy discussed in the previous section.

We can draw, then, a second conclusion: while VerbNet is finer-grained than PropBank, the two classifications are not in contradiction with each other. VerbNet greater specificity can be used in different ways depending on the frequency of the label. Practically, PropBank labels could provide a strong generalisation to a VerbNet annotation at high-frequency. VerbNet labels, on the other hand, can act as disambiguators of overloaded variables in PropBank. This conclusion was also reached by Loper et al. (2007). Thus, both annotation schemes could be useful in different circumstances and at different frequency bands.

5 The Combinatorics of Semantic Roles

Semantic roles exhibit paradigmatic generalisations — generalisations across similar semantic roles in the inventory — (which we saw in section 4.) They also show syntagmatic generalisations, generalisations that concern the context. One kind of context is provided by what other roles they can occur with. It has often been observed that certain semantic roles sets are possible, while others are not; among the possible sets, certain are much more frequent than others (Levin and Rappaport Hovav, 2005). Some linguistically-inspired

	A0	A1	A2	A3	A4	A5	AA
Actor	0.0	-	-	-	-	-	-
Actor1	0.8	-	-	-	-	-	-
Actor2	-	0.3	0.1	-	-	-	-
Agent1	0.0	-	-	-	-	-	-
Agent	32.6	0.2	-	-	-	-	-
Asset	-	0.1	0.0	0.2	-	-	-
Attribute	-	0.1	0.7	-	-	-	-
Beneficiary	-	0.0	0.1	0.1	0.0	-	-
Cause	0.7	1.1	0.1	0.1	-	-	-
Destination	-	0.4	0.8	0.0	-	-	-
Experiencer	3.3	0.9	0.1	-	-	-	-
Extent	-	-	1.3	-	-	-	-
Goal	-	-	-	-	0.0	-	-
Instrument	-	-	0.1	0.0	-	-	-
Location	0.0	0.4	0.6	0.0	-	0.0	-
Material	-	0.1	0.1	0.0	-	-	-
Patient	0.0	5.8	-	-	-	-	-
Patient1	0.1	1.1	-	-	-	-	-
Patient2	-	0.1	0.5	-	-	-	-
Predicate	-	1.2	1.1	0.0	-	-	-
Product	0.0	1.5	0.1	-	0.0	-	-
Proposition	-	0.0	0.1	-	-	-	-
Recipient	-	0.3	2.0	-	0.0	-	-
Source	-	0.3	0.5	0.1	-	-	-
Stimulus	-	1.0	-	-	-	-	-
Theme	0.8	25.1	0.5	0.0	-	-	0.0
Theme1	0.4	0.4	0.0	0.0	-	-	-
Theme2	0.1	0.4	0.3	-	-	-	-
Topic	-	11.2	0.3	-	-	-	-
Value	-	0.1	-	-	-	-	-

Table 4: Distribution of PropBank by original VerbNet labels according to SemLink. Counts indicated as 0.0 approximate zero by rounding, while a - sign indicates that no occurrences were found.

semantic role labelling techniques do attempt to model these dependencies directly (Toutanova et al., 2008; Merlo and Musillo, 2008).

Both annotation schemes impose tight constraints on co-occurrence of roles, independently of any verb information, with 62 role sets for PropBank and 116 role combinations for VerbNet, fewer than possible. Among the observed role sets, some are more frequent than expected under an assumption of independence between roles. For example, in PropBank, propositions comprising A0, A1 roles are observed 85% of the time, while they would be expected to occur only in 20% of the cases. In VerbNet the difference is also great between the 62% observed Agent, PredAttr propositions and the 14% expected.

Constraints on possible role sets are the expression of structural constraints among roles inherited from syntax, which we discuss in the next section, but also of the underlying event structure of the verb. Because of this relation, we expect a strong correlation between role sets and their associated

	A0,A1	A0,A2	A1,A2
Agent, Theme	11650	109	4
Agent, Topic	8572	14	0
Agent, Patient	1873	0	0
Experiencer, Theme	1591	0	15
Agent, Product	993	1	0
Agent, Predicate	960	64	0
Experiencer, Stimulus	843	0	0
Experiencer, Cause	756	0	2

Table 5: Sample of role sets correspondences

verb, as well as role sets and verb classes for both annotation schemes. However, PropBank roles are associated based on the meaning of the verb, but also based on their positional prominence in the tree, and so we can expect their relation to the actual verb entry to be weaker.

We measure here simply the correlation as indicated by the symmetric uncertainty of the joint distribution of role sets by verbs and of role sets by verb classes, for each of the two annotation schemes. We find that the correlation between PropBank role sets and verb classes is weaker than the correlation between VerbNet role sets and verb classes, as expected (PropBank: U=0.21 vs VerbNet: U=0.46). We also find that correlation between PropBank role sets and verbs is weaker than the correlation between VerbNet role sets and verbs (PropBank: U=0.23 vs VerbNet U=0.43). Notice that this result holds for VerbNet role label groups, and is therefore not a side-effect of a different size in role inventory. This result confirms our findings reported in Table 2, which showed a larger degradation of VerbNet labels in the absence of verb information.

If we analyse the data, we see that many role sets that form one single set in PropBank are split into several sets in VerbNet, with those roles that are different being roles that in PropBank form a group. So, for example, a role list (A0, A1) in PropBank will corresponds to 14 different lists in VerbNet (when using the groups). The three most frequent VerbNet role sets describe 86% of the cases: (Agent, Predattr) 71%, (InstrCause, PredAttr) 9%, and (Agent, Patient) 6% . Using the original VerbNet labels – a very small sample of the most frequent ones is reported in Table 5 — we find 39 different sets. Conversely, we see that VerbNet sets corresponds to few PropBank sets, even for high frequency.

The third conclusion we can draw then is twofold. First, while VerbNet labels have been assigned to be valid across verbs, as confirmed by their ability to enter in many combinations, these combinations are more verb and class-specific than combinations in PropBank. Second, the fine-grained, coarse-grained correspondence of annotations between VerbNet and PropBank that was illustrated by the results in Section 4 is also borne out when we look at role sets: PropBank role sets appear to be high-level abstractions of VerbNet role sets.

6 Semantic Roles and Grammatical Functions: the Thematic Hierarchy

A different kind of context-dependence is provided by thematic hierarchies. It is a well-attested fact that lexical semantic properties described by semantic roles and grammatical functions appear to be distributed according to prominence scales (Levin and Rappaport Hovav, 2005). Semantic roles are organized according to the thematic hierarchy (one proposal among many is Agent $>$ Experiencer$>$ Goal/Source/Location$>$ Patient (Grimshaw, 1990)). This hierarchy captures the fact that the options for the structural realisation of a particular argument do not depend only on its role, but also on the roles of other arguments. For example in psychological verbs, the position of the Experiencer as a syntactic subject or object depends on whether the other role in the sentence is a Stimulus, hence lower in the hierarchy, as in the psychological verbs of the *fear* class or an Agent/Cause as in the *frighten* class. Two prominence scales can combine by matching elements harmonically, higher elements with higher elements and lower with lower (Aissen, 2003). Grammatical functions are also distributed according to a prominence scale. Thus, we find that most subjects are Agents, most objects are Patients or Themes, and most indirect objects are Goals, for example.

The semantic role inventory, thus, should show a certain correlation with the inventory of grammatical functions. However, perfect correlation is clearly not expected as in this case the two levels of representation would be linguistically and computationally redundant. Because PropBank was annotated according to argument prominence, we expect to see that PropBank reflects relationships between syntax and semantic role labels more strongly than VerbNet. Comparing syntactic dependency labels to their corresponding PropBank or VerbNet groups labels (groups are used to elim-

inate the confound of different inventory sizes), we find that the joint entropy of PropBank and dependency labels is 2.61 bits while the joint entropy of VerbNet and dependency labels is 3.32 bits. The symmetric uncertainty of PropBank and dependency labels is 0.49, while the symmetric uncertainty of VerbNet and dependency labels is 0.39.

On the basis of these correlations, we can confirm previous findings: PropBank more closely captures the thematic hierarchy and is more correlated to grammatical functions, hence potentially more useful for semantic role labelling, for learners whose features are based on the syntactic tree. VerbNet, however, provides a level of annotation that is more independent of syntactic information, a property that might be useful in several applications, such as machine translation, where syntactic information might be too language-specific.

7 Generality of Semantic Roles

Semantic roles are not meant to be domain-specific, but rather to encode aspects of our conceptualisation of the world. A semantic role inventory that wants to be linguistically perspicuous and also practically useful in several tasks needs to reflect our grammatical representation of events. VerbNet is believed to be superior in this respect to PropBank, as it attempts to be less verb-specific and to be portable across classes. Previous results (Loper et al., 2007; Zapirain et al., 2008) appear to indicate that this is not the case because a labeller has better performance with PropBank labels than with VerbNet labels. But these results are task-specific, and they were obtained in the context of parsing. Since we know that PropBank is more closely related to grammatical function and syntactic annotation than VerbNet, as indicated above in Section 6, then these results could simply indicate that parsing predicts PropBank labels better because they are more closely related to syntactic labels, and not because the semantic roles inventory is more general.

Several of the findings in the previous sections shed light on the generality of the semantic roles in the two inventories. Results in Section 3 show that previous results can be reinterpreted as indicating that VerbNet labels generalise better to new roles.

We attempt here to determine the generality of the "meaning" of a role label without recourse to a task-specific experiment. It is often claimed in the literature that semantic roles are better described by feature bundles. In particular, the features sentience and volition have been shown to be useful in distinguishing Proto-Agents from Proto-Patients (Dowty, 1991). These features can be assumed to be correlated to animacy. Animacy has indeed been shown to be a reliable indicator of semantic role differences (Merlo and Stevenson, 2001). Personal pronouns in English grammaticalise animacy. We extract all the occurrences of the unambiguously animate pronouns (*I, you, he, she, us, we, me, us, him*) and the unambiguously inanimate pronoun *it*, for each semantic role label, in PropBank and VerbNet. We find occurrences for three semantic role labels in PropBank and six in VerbNet. We reduce the VerbNet groups to five by merging Patient roles with PredAttr roles to avoid artificial variation among very similar roles. An analysis of variance of the distributions of the pronous yields a significant effect of animacy for VerbNet ($F(4)=5.62$, $p< 0.05$), but no significant effect for PropBank ($F(2)=4.94$, $p=0.11$). This result is a preliminary indication that VerbNet labels might capture basic components of meaning more clearly than PropBank labels, and that they might therefore be more general.

8 Conclusions

In this paper, we have proposed a task-independent, general method to analyse annotation schemes. The method is based on information-theoretic measures and comparison with attested linguistic generalisations, to evaluate how well semantic role inventories and annotations capture grammaticalised aspects of meaning. We show that VerbNet is more verb-specific and better able to generalise to new semantic roles, while PropBank, because of its relation to syntax, better captures some of the structural constraints among roles. Future work will investigate another basic property of semantic role labelling schemes: cross-linguistic validity.

Acknowledgements

We thank James Henderson and Ivan Titov for useful comments. The research leading to these results has received partial funding from the EU FP7 programme (FP7/2007-2013) under grant agreement number 216594 (CLASSIC project: www.classic-project.org).

References

Judith Aissen. 2003. Differential object marking: Iconicity vs. economy. *Natural Language and Linguistic Theory*, 21:435–483.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics (ACL-COLING'98)*, pages 86–90, Montreal, Canada.

David Dowty. 1991. Thematic proto-roles and argument selection. *Language*, 67(3):547–619.

Charles Fillmore. 1968. The case for case. In Emmon Bach and Harms, editors, *Universals in Linguistic Theory*, pages 1–88. Holt, Rinehart, and Winston.

Jane Grimshaw. 1990. *Argument Structure*. MIT Press.

Jeffrey Gruber. 1965. *Studies in Lexical Relation*. MIT Press, Cambridge, MA.

Ray Jackendoff. 1972. *Semantic Interpretation in Generative Grammar*. MIT Press, Cambridge, MA.

Karin Kipper. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, University of Pennsylvania.

Beth Levin and Malka Rappaport Hovav. 2005. *Argument Realization*. Cambridge University Press, Cambridge, UK.

Edward Loper, Szu ting Yi, and Martha Palmer. 2007. Combining lexical resources: Mapping between PropBank and VerbNet. In *Proceedings of the IWCS*.

Christopher Manning and Hinrich Schuetze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Mitch Marcus, Beatrice Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

Paola Merlo and Gabriele Musillo. 2008. Semantic parsing for high-precision semantic role labelling. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CONLL-08)*, pages 1–8, Manchester, UK.

Paola Merlo and Suzanne Stevenson. 2001. Automatic verb classification based on statistical distributions of argument structure. *Computational Linguistics*, 27(3):373–408.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31:71–105.

Josef Ruppenhofer, Michael Ellsworth, Miriam Petruck, Christopher Johnson, and Jan Scheffczyk. 2006. Framenet ii: Theory and practice. Technical report, Berkeley,CA.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*, pages 159–177.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2).

Ian Witten and Eibe Frank. 2005. *Data Mining*. Elsevier.

Szu-ting Yi, Edward Loper, and Martha Palmer. 2007. Can semantic roles generalize across genres? In *Proceedings of the Human Language Technologies 2007 (NAACL-HLT'07)*, pages 548–555, Rochester, New York, April.

Beñat Zapirain, Eneko Agirre, and Lluís Màrquez. 2008. Robustness and generalization of role sets: PropBank vs. VerbNet. In *Proceedings of ACL-08: HLT*, pages 550–558, Columbus, Ohio, June.

Robust Machine Translation Evaluation with Entailment Features*

Sebastian Padó
Stuttgart University
pado@ims.uni-stuttgart.de

Michel Galley, Dan Jurafsky, Chris Manning
Stanford University
{mgalley,jurafsky,manning}@stanford.edu

Abstract

Existing evaluation metrics for machine translation lack crucial *robustness*: their correlations with human quality judgments vary considerably across languages and genres. We believe that the main reason is their inability to properly capture *meaning*: A good translation candidate *means* the same thing as the reference translation, regardless of formulation. We propose a metric that evaluates MT output based on a rich set of features motivated by *textual entailment*, such as lexical-semantic (in-)compatibility and argument structure overlap. We compare this metric against a combination metric of four state-of-the-art scores (BLEU, NIST, TER, and METEOR) in two different settings. The combination metric outperforms the individual scores, but is bested by the entailment-based metric. Combining the entailment and traditional features yields further improvements.

1 Introduction

Constant evaluation is vital to the progress of machine translation (MT). Since human evaluation is costly and difficult to do reliably, a major focus of research has been on *automatic* measures of MT quality, pioneered by BLEU (Papineni et al., 2002) and NIST (Doddington, 2002). BLEU and NIST measure MT quality by using the strong correlation between human judgments and the degree of n-gram overlap between a system hypothesis translation and one or more reference translations. The resulting scores are cheap and objective.

However, studies such as Callison-Burch et al. (2006) have identified a number of problems with BLEU and related n-gram-based scores: (1) BLEU-like metrics are unreliable at the level of individual sentences due to data sparsity; (2) BLEU metrics can be "gamed" by permuting word order; (3) for some corpora and languages, the correlation to human ratings is very low even at the system level; (4) scores are biased towards statistical MT; (5) the quality gap between MT and human translations is not reflected in equally large BLEU differences.

*This paper is based on work funded by the Defense Advanced Research Projects Agency through IBM. The content does not necessarily reflect the views of the U.S. Government, and no official endorsement should be inferred.

This is problematic, but not surprising: The metrics treat *any* divergence from the reference as a negative, while (computational) linguistics has long dealt with linguistic variation that preserves the meaning, usually called *paraphrase*, such as:

(1) **HYP:** However, this was declared terrorism by observers and witnesses.
REF: Nevertheless, commentators as well as eyewitnesses are terming it terrorism.

A number of metrics have been designed to account for paraphrase, either by making the matching more intelligent (TER, Snover et al. (2006)), or by using linguistic evidence, mostly lexical similarity (METEOR, Banerjee and Lavie (2005); MaxSim, Chan and Ng (2008)), or syntactic overlap (Owczarzak et al. (2008); Liu and Gildea (2005)). Unfortunately, each metrics tend to concentrate on one particular type of linguistic information, none of which always correlates well with human judgments.

Our paper proposes two strategies. We first explore the combination of traditional scores into a more robust ensemble metric with linear regression. Our second, more fundamental, strategy replaces the use of loose surrogates of translation quality with a model that attempts to comprehensively assess *meaning equivalence* between references and MT hypotheses. We operationalize meaning equivalence by bidirectional *textual entailment* (RTE, Dagan et al. (2005)), and thus predict the quality of MT hypotheses with a rich RTE feature set. The entailment-based model goes beyond existing word-level "semantic" metrics such as METEOR by integrating phrasal and compositional aspects of meaning equivalence, such as multiword paraphrases, (in-)correct argument and modification relations, and (dis-)allowed phrase reorderings. We demonstrate that the resulting metric beats both individual and combined traditional MT metrics. The complementary features of both metric types can be combined into a joint, superior metric.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 297–305,
Suntec, Singapore, 2-7 August 2009.

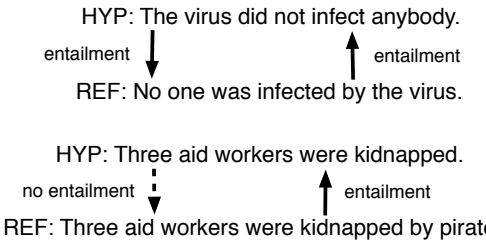


Figure 1: Entailment status between an MT system hypothesis and a reference translation for equivalent (top) and non-equivalent (bottom) translations.

2 Regression-based MT Quality Prediction

Current MT metrics tend to focus on a single dimension of linguistic information. Since the importance of these dimensions tends not to be stable across language pairs, genres, and systems, performance of these metrics varies substantially. A simple strategy to overcome this problem could be to combine the judgments of different metrics. For example, Paul et al. (2007) train binary classifiers on a feature set formed by a number of MT metrics. We follow a similar idea, but use a regularized linear regression to directly predict human ratings.

Feature combination via regression is a supervised approach that requires labeled data. As we show in Section 5, this data is available, and the resulting model generalizes well from relatively small amounts of training data.

3 Textual Entailment vs. MT Evaluation

Our novel approach to MT evaluation exploits the similarity between MT evaluation and textual entailment (TE). TE was introduced by Dagan et al. (2005) as a concept that corresponds more closely to "common sense" reasoning patterns than classical, strict logical entailment. Textual entailment is defined informally as a relation between two natural language sentences (a premise P and a hypothesis H) that holds if "a human reading P would infer that H is most likely true". Knowledge about entailment is beneficial for NLP tasks such as Question Answering (Harabagiu and Hickl, 2006).

The relation between textual entailment and MT evaluation is shown in Figure 1. Perfect MT output and the reference translation entail each other (top). Translation problems that impact semantic equivalence, e.g., deletion or addition of material, can break entailment in one or both directions (bottom).

On the modelling level, there is common ground between RTE and MT evaluation: Both have to distinguish between valid and invalid variation to determine whether two texts convey the same information or not. For example, to recognize the bidirectional entailment in Ex. (1), RTE must account for the following reformulations: synonymy (*However/Nevertheless*), more general semantic relatedness (*observers/commentators*), phrasal replacements (*and/as well as*), and an active/passive alternation that implies structural change (*is declared/are terming*). This leads us to our main hypothesis: RTE features are designed to distinguish meaning-preserving variation from true divergence and are thus *also* good predictors in MT evaluation. However, while the original RTE task is asymmetric, MT evaluation needs to determine meaning equivalence, which is a symmetric relation. We do this by checking for entailment in *both* directions (see Figure 1). Operationally, this ensures we detect translations which either delete or insert material.

Clearly, there are also differences between the two tasks. An important one is that RTE assumes the well-formedness of the two sentences. This is not generally true in MT, and could lead to degraded linguistic analyses. However, entailment relations are more sensitive to the contribution of individual words (MacCartney and Manning, 2008). In Example 2, the modal modifiers break the entailment between two otherwise identical sentences:

(2) **HYP:** Peter is *certainly* from Lincolnshire.
REF: Peter is *possibly* from Lincolnshire.

This means that the prediction of TE hinges on correct semantic analysis and is sensitive to misanalyses. In contrast, human MT judgments behave robustly. Translations that involve individual errors, like (2), are judged lower than perfect ones, but usually not crucially so, since most aspects are still rendered correctly. We thus expect even noisy RTE features to be predictive for translation quality. This allows us to use an off-the-shelf RTE system to obtain features, and to combine them using a regression model as described in Section 2.

3.1 The Stanford Entailment Recognizer

The Stanford Entailment Recognizer (MacCartney et al., 2006) is a stochastic model that computes match and mismatch features for each premise-hypothesis pair. The three stages of the system are shown in Figure 2. The system first uses a robust broad-coverage PCFG parser and a deterministic constituent-dependency converter to construct linguistic representations of the premise and

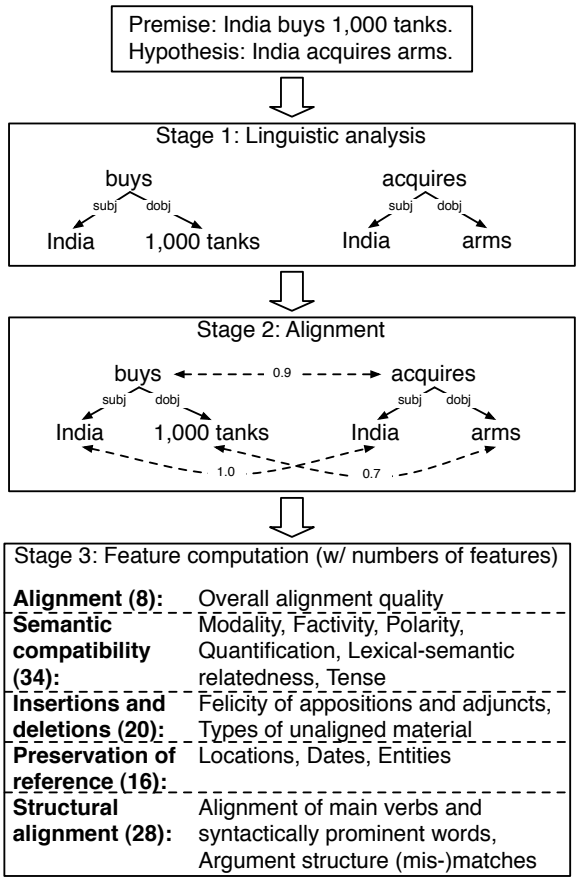


Figure 2: The Stanford Entailment Recognizer

the hypothesis. The results are typed dependency graphs that contain a node for each word and labeled edges representing the grammatical relations between words. Named entities are identified, and contiguous collocations grouped. Next, it identifies the highest-scoring alignment from each node in the hypothesis graph to a single node in the premise graph, or to null. It uses a locally decomposable scoring function: The score of an alignment is the sum of the local word and edge alignment scores. The computation of these scores make extensive use of about ten lexical similarity resources, including WordNet, InfoMap, and Dekang Lin's thesaurus. Since the search space is exponential in the hypothesis length, the system uses stochastic (rather than exhaustive) search based on Gibbs sampling (see de Marneffe et al. (2007)).

Entailment features. In the third stage, the system produces roughly 100 features for each aligned premise-hypothesis pair. A small number of them are real-valued (mostly quality scores), but most are binary implementations of small linguistic theories whose activation indicates syntactic and semantic (mis-)matches of different types. Figure 2 groups the features into five classes. *Alignment* features measure the overall quality of the alignment as given by the lexical resources. *Semantic compatibility* features check to what extent the aligned material has the same meaning and preserves semantic dimensions such as modality and factivity, taking a limited amount of context into account. *Insertion/deletion* features explicitly address material that remains unaligned and assess its felicity. *Reference* features ascertain that the two sentences actually refer to the same events and participants. Finally, *structural* features add structural considerations by ensuring that argument structure is preserved in the translation. See MacCartney et al. (2006) for details on the features, and Sections 5 and 6 for examples of feature firings.

Efficiency considerations. The use of deep linguistic analysis makes our entailment-based metric considerably more heavyweight than traditional MT metrics. The average total runtime per sentence pair is 5 seconds on an AMD 2.6GHz Opteron core – efficient enough to perform regular evaluations on development and test sets. We are currently investigating caching and optimizations that will enable the use of our metric for MT parameter tuning in a Minimum Error Rate Training setup (Och, 2003).

4 Experimental Evaluation

4.1 Experiments

Traditionally, human ratings for MT quality have been collected in the form of *absolute scores* on a five- or seven-point Likert scale, but low reliability numbers for this type of annotation have raised concerns (Callison-Burch et al., 2008). An alternative that has been adopted by the yearly WMT evaluation shared tasks since 2008 is the collection of *pairwise preference judgments* between pairs of MT hypotheses which can be elicited (somewhat) more reliably. We demonstrate that our approach works well for both types of annotation and different corpora. Experiment 1 models absolute scores on Asian newswire, and Experiment 2 pairwise preferences on European speech and news data.

4.2 Evaluation

We evaluate the output of our models both on the sentence and on the system level. At the sentence level, we can correlate predictions in Experiment 1 directly with human judgments with Spearman's ρ,

a non-parametric rank correlation coefficient appropriate for non-normally distributed data. In Experiment 2, the predictions cannot be pooled between sentences. Instead of correlation, we compute "consistency" (i.e., accuracy) with human preferences.

System-level predictions are computed in both experiments from sentence-level predictions, as the ratio of sentences for which each system provided the best translation (Callison-Burch et al., 2008). We extend this procedure slightly because real-valued predictions cannot predict ties, while human raters decide for a significant portion of sentences (as much as 80% in absolute score annotation) to "tie" two systems for first place. To simulate this behavior, we compute "tie-aware" predictions as the percentage of sentences where the system's hypothesis was assigned a score *better or at most* ε *worse than the best system*. ε is set to match the frequency of ties in the training data.

Finally, the predictions are again correlated with human judgments using Spearman's ρ. "Tie awareness" makes a considerable practical difference, improving correlation figures by 5–10 points.[1]

4.3 Baseline Metrics

We consider four baselines. They are small regression models as described in Section 2 over component scores of four widely used MT metrics. To alleviate possible nonlinearity, we add all features in linear and log space. Each baselines carries the name of the underlying metric plus the suffix *-R*.[2]

BLEUR includes the following 18 sentence-level scores: BLEU-*n* and *n*-gram precision scores ($1 \leq n \leq 4$); BLEU brevity penalty (BP); BLEU score divided by BP. To counteract BLEU's brittleness at the sentence level, we also smooth BLEU-*n* and *n*-gram precision as in Lin and Och (2004).

NISTR consists of 16 features. NIST-*n* scores ($1 \leq n \leq 10$) and information-weighted *n*-gram precision scores ($1 \leq n \leq 4$); NIST brevity penalty (BP); and NIST score divided by BP.

TERR includes 50 features. We start with the standard TER score and the number of each of the four edit operations. Since the default uniform cost does not always correlate well with human judgment, we duplicate these features for 9 non-uniform edit costs. We find it effective to set insertion cost close to 0, as a way of enabling surface variation, and indeed the new TERp metric uses a similarly low default insertion cost (Snover et al., 2009).

METEORR consists of METEOR v0.7.

4.4 Combination Metrics

The following three regression models implement the methods discussed in Sections 2 and 3.

MTR combines the 85 features of the four baseline models. It uses no entailment features.

RTER uses the 70 entailment features described in Section 3.1, but no MTR features.

MT+RTER uses all MTR and RTER features, combining matching and entailment evidence.[3]

5 Expt. 1: Predicting Absolute Scores

Data. Our first experiment evaluates the models we have proposed on a corpus with traditional annotation on a seven-point scale, namely the NIST OpenMT 2008 corpus.[4] The corpus contains translations of newswire text into English from three source languages (Arabic (Ar), Chinese (Ch), Urdu (Ur)). Each language consists of 1500–2800 sentence pairs produced by 7–15 MT systems.

We use a "round robin" scheme. We optimize the weights of our regression models on two languages and then predict the human scores on the third language. This gauges performance of our models when training and test data come from the same genre, but from different languages, which we believe to be a setup of practical interest. For each test set, we set the system-level tie parameter ε so that the relative frequency of ties was equal to the training set (65–80%). Hypotheses generally had to receive scores within $0.3 - 0.5$ points to tie.

Results. Table 1 shows the results. We first concentrate on the upper half (sentence-level results). The predictions of all models correlate highly significantly with human judgments, but we still see robustness issues for the individual MT metrics.

[1] Due to space constraints, we only show results for "tie-aware" predictions. See Padó et al. (2009) for a discussion.

[2] The regression models can simulate the behaviour of each component by setting the weights appropriately, but are strictly more powerful. A possible danger is that the parameters overfit on the training set. We therefore verified that the three non-trivial "baseline" regression models indeed confer a benefit over the default component combination scores: BLEU-1 (which outperformed BLEU-4 in the MetricsMATR 2008 evaluation), NIST-4, and TER (with all costs set to 1). We found higher robustness and improved correlations for the regression models. An exception is BLEU-1 and NIST-4 on Expt. 1 (Ar, Ch), which perform 0.5–1 point better at the sentence level.

[3] Software for RTER and MT+RTER is available from `http://nlp.stanford.edu/software/mteval.shtml`.

[4] Available from `http://www.nist.gov`.

Evaluation	Data		Metrics						
	train	test	BLEUR	METEORR	NISTR	TERR	MTR	RTER	MT+RTER
Sentence-level	Ar+Ch	Ur	49.9	49.1	49.5	50.1	50.1	54.5	**55.6**
	Ar+Ur	Ch	53.9	61.1	53.1	50.3	57.3	58.0	**62.7**
	Ch+Ur	Ar	52.5	60.1	50.4	54.5	55.2	59.9	**61.1**
System-level	Ar+Ch	Ur	73.9	68.4	50.0	90.0*	**92.7***	77.4*	81.0*
	Ar+Ur	Ch	38.5	44.3	40.0	**59.0***	51.8*	47.7	57.3*
	Ch+Ur	Ar	59.7*	**86.3***	61.9*	42.1	48.1	59.7*	61.7*

Table 1: Expt. 1: Spearman's ρ for correlation between human absolute scores and model predictions on NIST OpenMT 2008. Sentence level: All correlations are highly significant. System level: *: $p<0.05$.

METEORR achieves the best correlation for Chinese and Arabic, but fails for Urdu, apparently the most difficult language. TERR shows the best result for Urdu, but does worse than METEORR for Arabic and even worse than BLEUR for Chinese. The MTR combination metric alleviates this problem to some extent by improving the "worst-case" performance on Urdu to the level of the best individual metric. The entailment-based RTER system outperforms MTR on each language. It particularly improves on MTR's correlation on Urdu. Even though METEORR still does somewhat better than MTR and RTER, we consider this an important confirmation for the usefulness of entailment features in MT evaluation, and for their robustness.[5]

In addition, the combined model MT+RTER is best for all three languages, outperforming METEORR for each language pair. It performs considerably better than either MTR or RTER. This is a second result: the types of evidence provided by MTR and RTER appear to be *complementary* and can be combined into a superior model.

On the system level (bottom half of Table 1), there is high variance due to the small number of predictions per language, and many predictions are not significantly correlated with human judgments. BLEUR, METEORR, and NISTR significantly predict one language each (all Arabic); TERR, MTR, and RTER predict two languages. MT+RTER is the only model that shows significance for all three languages. This result supports the conclusions we have drawn from the sentence-level analysis.

Further analysis. We decided to conduct a thorough analysis of the Urdu dataset, the most difficult source language for all metrics. We start with a feature ablation study. Removing any feature group from RTER results in drops in correlation of at least three points. The largest drops occur for the structural ($\delta = -11$) and insertion/deletion ($\delta = -8$) features. Thus, all feature groups appear to contribute to the good correlation of RTER. However, there are big differences in the generality of the feature groups: in isolation, the insertion/deletion features achieve almost no correlation, and need to be complemented by more robust features.

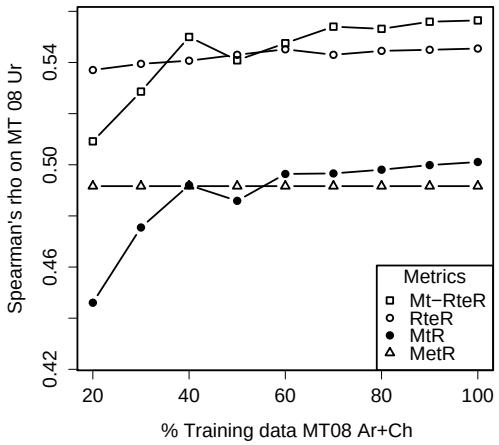


Figure 3: Experiment 1: Learning curve (Urdu).

Next, we analyze the role of training data. Figure 3 shows Urdu average correlations for models trained on increasing subsets of the training data (10% increments, 10 random draws per step; Ar and Ch show similar patterns.) METEORR does not improve, which is to be expected given the model definition. RTER has a rather flat learning curve that climbs to within 2 points of the final correlation value for 20% of the training set (about 400 sentence pairs). Apparently, entailment features do not require a large training set, presumably because most features of RTER are binary. The remaining two models, MTR and MT+RTER, show clearer benefit from more data. With 20% of the total data, they climb to within 5 points of their final performance, but keep slowly improving further.

[5]These results are substantially better than the performance our metric showed in the MetricsMATR 2008 challenge. Beyond general enhancement of our model, we attribute the less good MetricsMATR 2008 results to an infelicitous choice of training data for the submission, coupled with the large amount of ASR output in the test data, whose disfluencies represent an additional layer of problems for deep approaches.

REF: I shall face that fact today. **HYP:** Today I will face this reality. [doc WL-34-174270-7483871, sent 4, system1]	Gold: 6 METEORR: 2.8 RTER: **6.1** • Only function words unaligned (*will*, *this*) • Alignment *fact/reality*: hypernymy is ok in upward monotone context
REF: What does BBC's Haroon Rasheed say after a visit to Lal Masjid Jamia Hafsa complex? There are no underground tunnels in Lal Masjid or Jamia Hafsa. The presence of the foreigners could not be confirmed as well. What became of the extremists like Abuzar? **HYP:** *BBC Haroon Rasheed Lal Masjid, Jamia Hafsa after* his *visit to* Auob Medical *Complex says Lal Masjid* and seminary in under a land mine, *not also been confirmed the presence of foreigners could not be, such as Abu* by the *extremist?* [doc WL-12-174261-7457007, sent 2, system2]	Gold: 1 METEORR: 4.5 RTER: **1.2** • Hypothesis root node unaligned • Missing alignments for subjects • Important entities in hypothesis cannot be aligned • Reference, hypothesis differ in polarity

Table 2: Expt. 1: Reference translations and MT output (Urdu). Scores are out of 7 (higher is better).

Finally, we provide a qualitative comparison of RTER's performance against the best baseline metric, METEORR. Since the computation of RTER takes considerably more resources than METEORR, it is interesting to compare the predictions of RTER against METEORR. Table 2 shows two classes of examples with apparent improvements.

The first example (top) shows a good translation that is erroneously assigned a low score by METEORR because (a) it cannot align *fact* and *reality* (METEORR aligns only synonyms) and (b) it punishes the change of word order through its "penalty" term. RTER correctly assigns a high score. The features show that this prediction results from two semantic judgments. The first is that the lack of alignments for two function words is unproblematic; the second is that the alignment between *fact* and *reality*, which is established on the basis of WordNet similarity, is indeed licensed in the current context. More generally, we find that RTER is able to account for more valid variation in good translations because (a) it judges the validity of alignments dependent on context; (b) it incorporates more semantic similarities; and (c) it weighs mismatches according to the word's status.

The second example (bottom) shows a very bad translation that is scored highly by METEORR, since almost all of the reference words appear either literally or as synonyms in the hypothesis (marked in italics). In combination with METEORR's concentration on recall, this is sufficient to yield a moderately high score. In the case of RTER, a number of mismatch features have fired. They indicate problems with the structural well-formedness of the MT output as well as semantic incompatibility between hypothesis and reference (argument structure and reference mismatches).

6 Expt. 2: Predicting Pairwise Preferences

In this experiment, we predict human pairwise preference judgments (cf. Section 4). We reuse the linear regression framework from Section 2 and predict pairwise preferences by predicting two absolute scores (as before) and comparing them.[6]

Data. This experiment uses the 2006–2008 corpora of the Workshop on Statistical Machine Translation (WMT).[7] It consists of data from EUROPARL (Koehn, 2005) and various news commentaries, with five source languages (French, German, Spanish, Czech, and Hungarian). As training set, we use the portions of WMT 2006 and 2007 that are annotated with absolute scores on a five-point scale (around 14,000 sentences produced by 40 systems). The test set is formed by the WMT 2008 relative rank annotation task. As in Experiment 1, we set ε so that the incidence of ties in the training and test set is equal (60%).

Results. Table 4 shows the results. The left result column shows consistency, i.e., the accuracy on human pairwise preference judgments.[8] The pattern of results matches our observations in Expt. 1: Among individual metrics, METEORR and TERR do better than BLEUR and NISTR. MTR and RTER outperform individual metrics. The best result by a wide margin, 52.5%, is shown by MT+RTER.

[6] We also experimented with a logistic regression model that predicts binary preferences directly. Its performance is comparable; see Padó et al. (2009) for details.

[7] Available from `http://www.statmt.org/`.

[8] The random baseline is not 50%, but, according to our experiments, 39.8%. This has two reasons: (1) the judgments include contradictory and tie annotations that cannot be predicted correctly (raw inter-annotator agreement on WMT 2008 was 58%); (2) metrics have to submit a total order over the translations for each sentence, which introduces transitivity constraints. For details, see Callison-Burch et al. (2008).

Segment	MTR	RTER	MT+RTER	Gold
REF: Scottish NHS boards need to improve criminal records checks for employees outside Europe, a watchdog has said. HYP: The Scottish health ministry should improve the controls on extra-community employees to check whether they have criminal precedents, said the monitoring committee. [1357, lium-systran]	Rank: 3	Rank: **1**	Rank: 2	Rank: 1
REF: Arguments, bullying and fights between the pupils have extended to the relations between their parents. HYP: Disputes, chicane and fights between the pupils transposed in relations between the parents. [686, rbmt4]	Rank: **5**	Rank: 2	Rank: 4	Rank: 5

Table 3: Expt. 2: Reference translations and MT output (French). Ranks are out of five (smaller is better).

Feature set	Consistency (%)	System-level correlation (ρ)
BLEUR	49.6	69.3
METEORR	51.1	72.6
NISTR	50.2	70.4
TERR	51.2	72.5
MTR	51.5	73.1
RTER	51.8	**78.3**
MT+RTER	**52.5**	75.8
WMT 08 (worst)	44	37
WMT 08 (best)	56	83

Table 4: Expt. 2: Prediction of pairwise preferences on the WMT 2008 dataset.

The right column shows Spearman's ρ for the correlation between human judgments and tie-aware system-level predictions. All metrics predict system scores highly significantly, partly due to the larger number of systems compared (87 systems). Again, we see better results for METEORR and TERR than for BLEUR and NISTR, and the individual metrics do worse than the combination models. Among the latter, the order is: MTR (worst), MT+RTER, and RTER (best at 78.3).

WMT 2009. We submitted the Expt. 2 RTER metric to the WMT 2009 shared MT evaluation task (Padó et al., 2009). The results provide further validation for our results and our general approach. At the system level, RTER made third place (avg. correlation $\rho = 0.79$), trailing the two top metrics closely ($\rho = 0.80$, $\rho = 0.83$) and making the best predictions for Hungarian. It also obtained the second-best consistency score (53%, best: 54%).

Metric comparison. The pairwise preference annotation of WMT 2008 gives us the opportunity to compare the MTR and RTER models by computing consistency separately on the "top" (highest-ranked) and "bottom" (lowest-ranked) hypotheses for each reference. RTER performs about 1.5 percent better on the top than on the bottom hypotheses. The MTR model shows the inverse behavior, performing 2 percent worse on the top hypotheses. This matches well with our intuitions: We see some noise-induced degradation for the entailment features, but not much. In contrast, surface-based features are better at detecting bad translations than at discriminating among good ones.

Table 3 further illustrates the difference between the top models on two example sentences. In the top example, RTER makes a more accurate prediction than MTR. The human rater's favorite translation deviates considerably from the reference in lexical choice, syntactic structure, and word order, for which it is punished by MTR (rank 3/5). In contrast, RTER determines correctly that the propositional content of the reference is almost completely preserved (rank 1). In the bottom example, RTER's prediction is less accurate. This sentence was rated as bad by the judge, presumably due to the inappropriate main verb translation. Together with the subject mismatch, MTR correctly predicts a low score (rank 5/5). RTER's attention to semantic overlap leads to an incorrect high score (rank 2/5).

Feature Weights. Finally, we make two observations about feature weights in the RTER model.

First, the model has learned high weights not only for the overall alignment score (which behaves most similarly to traditional metrics), but also for a number of binary syntacto-semantic match and mismatch features. This confirms that these features systematically confer the benefit we have shown anecdotally in Table 2. Features with a consistently negative effect include dropping adjuncts, unaligned or poorly aligned root nodes, incompatible modality between the main clauses, person and location mismatches (as opposed to general mismatches) and wrongly handled passives. Con-

versely, higher scores result from factors such as high alignment score, matching embeddings under factive verbs, and matches between appositions.

Second, good MT evaluation feature weights are not good weights for RTE. Some differences, particularly for structural features, are caused by the low grammaticality of MT data. For example, the feature that fires for mismatches between dependents of predicates is unreliable on the WMT data. Other differences do reflect more fundamental differences between the two tasks (cf. Section 3). For example, RTE puts high weights onto quantifier and polarity features, both of which have the potential of influencing entailment decisions, but are (at least currently) unimportant for MT evaluation.

7 Related Work

Researchers have exploited various resources to enable the matching between words or *n*-grams that are semantically close but not identical. Banerjee and Lavie (2005) and Chan and Ng (2008) use WordNet, and Zhou et al. (2006) and Kauchak and Barzilay (2006) exploit large collections of automatically-extracted paraphrases. These approaches reduce the risk that a good translation is rated poorly due to lexical deviation, but do not address the problem that a translation may contain many long matches while lacking coherence and grammaticality (cf. the bottom example in Table 2).

Thus, incorporation of syntactic knowledge has been the focus of another line of research. Amigó et al. (2006) use the degree of overlap between the dependency trees of reference and hypothesis as a predictor of translation quality. Similar ideas have been applied by Owczarzak et al. (2008) to LFG parses, and by Liu and Gildea (2005) to features derived from phrase-structure tress. This approach has also been successful for the related task of summarization evaluation (Hovy et al., 2006).

The most comparable work to ours is Giménez and Márquez (2008). Our results agree on the crucial point that the use of a wide range of linguistic knowledge in MT evaluation is desirable and important. However, Giménez and Márquez advocate the use of a bottom-up development process that builds on a set of "heterogeneous", independent metrics each of which measures overlap with respect to one linguistic level. In contrast, our aim is to provide a "top-down", integrated motivation for the features we integrate through the textual entailment recognition paradigm.

8 Conclusion and Outlook

In this paper, we have explored a strategy for the evaluation of MT output that aims at comprehensively assessing the *meaning equivalence* between reference and hypothesis. To do so, we exploit the common ground between MT evaluation and the Recognition of Textual Entailment (RTE), both of which have to distinguish valid from invalid linguistic variation. Conceputalizing MT evaluation as an entailment problem motivates the use of a rich feature set that covers, unlike almost all earlier metrics, a wide range of linguistic levels, including lexical, syntactic, and compositional phenomena.

We have used an off-the-shelf RTE system to compute these features, and demonstrated that a regression model over these features can outperform an ensemble of traditional MT metrics in two experiments on different datasets. Even though the features build on deep linguistic analysis, they are robust enough to be used in a real-world setting, at least on written text. A limited amount of training data is sufficient, and the weights generalize well.

Our data analysis has confirmed that each of the feature groups contributes to the overall success of the RTE metric, and that its gains come from its better success at abstracting away from valid variation (such as word order or lexical substitution), while still detecting major semantic divergences. We have also clarified the relationship between MT evaluation and textual entailment: The majority of phenomena (but not all) that are relevant for RTE are also informative for MT evaluation.

The focus of this study was on the use of an existing RTE infrastructure for MT evaluation. Future work will have to assess the effectiveness of individual features and investigate ways to customize RTE systems for the MT evaluation task. An interesting aspect that we could not follow up on in this paper is that entailment features are linguistically interpretable (cf. Fig. 2) and may find use in uncovering systematic shortcomings of MT systems.

A limitation of our current metric is that it is language-dependent and relies on NLP tools in the target language that are still unavailable for many languages, such as reliable parsers. To some extent, of course, this problem holds as well for state-of-the-art MT systems. Nevertheless, it must be an important focus of future research to develop robust meaning-based metrics for other languages that can cash in the promise that we have shown for evaluating translation into English.

References

Enrique Amigó, Jesús Giménez, Julio Gonzalo, and Lluís Màrquez. 2006. MT Evaluation: Human-like vs. human acceptable. In *Proceedings of COLING/ACL 2006*, pages 17–24, Sydney, Australia.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures*, pages 65–72, Ann Arbor, MI.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of BLEU in machine translation research. In *Proceedings of EACL*, pages 249–256, Trento, Italy.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2008. Further meta-evaluation of machine translation. In *Proceedings of the ACL Workshop on Statistical Machine Translation*, pages 70–106, Columbus, OH.

Yee Seng Chan and Hwee Tou Ng. 2008. MAXSIM: A maximum similarity metric for machine translation evaluation. In *Proceedings of ACL-08: HLT*, pages 55–62, Columbus, Ohio, June.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, Southampton, UK.

Marie-Catherine de Marneffe, Trond Grenager, Bill MacCartney, Daniel Cer, Daniel Ramage, Chloé Kiddon, and Christopher D. Manning. 2007. Aligning semantic graphs for textual inference and machine reading. In *Proceedings of the AAAI Spring Symposium*, Stanford, CA.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram cooccurrence statistics. In *Proceedings of HLT*, pages 128–132, San Diego, CA.

Jesús Giménez and Lluís Márquez. 2008. Heterogeneous automatic MT evaluation through non-parametric metric combinations. In *Proceedings of IJCNLP*, pages 319–326, Hyderabad, India.

Sanda Harabagiu and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proceedings of ACL*, pages 905–912, Sydney, Australia.

Eduard Hovy, Chin-Yew Lin, Liang Zhou, and Junichi Fukumoto. 2006. Automated summarization evaluation with basic elements. In *Proceedings of LREC*, Genoa, Italy.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proceedings of HLT-NAACL*, pages 455–462.

Phillip Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the MT Summit X*, Phuket, Thailand.

Chin-Yew Lin and Franz Josef Och. 2004. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of COLING*, pages 501–507, Geneva, Switzerland.

Ding Liu and Daniel Gildea. 2005. Syntactic features for evaluation of machine translation. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures*, pages 25–32, Ann Arbor, MI.

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of COLING*, pages 521–528, Manchester, UK.

Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D. Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of NAACL*, pages 41–48, New York City, NY.

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

Karolina Owczarzak, Josef van Genabith, and Andy Way. 2008. Evaluating machine translation with LFG dependencies. *Machine Translation*, 21(2):95–119.

Sebastian Padó, Michel Galley, Dan Jurafsky, and Christopher D. Manning. 2009. Textual entailment features for machine translation evaluation. In *Proceedings of the EACL Workshop on Statistical Machine Translation*, pages 37–41, Athens, Greece.

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

Michael Paul, Andrew Finch, and Eiichiro Sumita. 2007. Reducing human assessment of machine translation quality to binary classifiers. In *Proceedings of TMI*, pages 154–162, Skövde, Sweden.

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

Matthew Snover, Nitin Madnani, Bonnie J. Dorr, and Richard Schwartz. 2009. Fluency, adequacy, or HTER? Exploring different human judgments with a tunable MT metric. In *Proceedings of the EACL Workshop on Statistical Machine Translation*, pages 259–268, Athens, Greece.

Liang Zhou, Chin-Yew Lin, and Eduard Hovy. 2006. Re-evaluating machine translation results with paraphrase support. In *Proceedings of EMNLP*, pages 77–84, Sydney, Australia.

The Contribution of Linguistic Features to Automatic Machine Translation Evaluation

Enrique Amigó[1] **Jesús Giménez**[2] **Julio Gonzalo** [1] **Felisa Verdejo**[1]
[1]UNED, Madrid
{enrique,julio,felisa}@lsi.uned.es
[2]UPC, Barcelona
jgimenez@lsi.upc.edu

Abstract

A number of approaches to Automatic MT Evaluation based on deep linguistic knowledge have been suggested. However, n-gram based metrics are still today the dominant approach. The main reason is that the advantages of employing deeper linguistic information have not been clarified yet. In this work, we propose a novel approach for meta-evaluation of MT evaluation metrics, since correlation cofficient against human judges do not reveal details about the advantages and disadvantages of particular metrics. We then use this approach to investigate the benefits of introducing linguistic features into evaluation metrics. Overall, our experiments show that (i) both lexical and linguistic metrics present complementary advantages and (ii) combining both kinds of metrics yields the most robust meta-evaluation performance.

1 Introduction

Automatic evaluation methods based on similarity to human references have substantially accelerated the development cycle of many NLP tasks, such as Machine Translation, Automatic Summarization, Sentence Compression and Language Generation. These automatic evaluation metrics allow developers to optimize their systems without the need for expensive human assessments for each of their possible system configurations. However, estimating the system output quality according to its similarity to human references is not a trivial task. The main problem is that many NLP tasks are open/subjective; therefore, different humans may generate different outputs, all of them equally valid. Thus, language variability is an issue.

In order to tackle language variability in the context of Machine Translation, a considerable effort has also been made to include deeper linguistic information in automatic evaluation metrics, both syntactic and semantic (see Section 2 for details). However, the most commonly used metrics are still based on n-gram matching. The reason is that the advantages of employing higher linguistic processing levels have not been clarified yet.

The main goal of our work is to analyze to what extent deep linguistic features can contribute to the automatic evaluation of translation quality. For that purpose, we compare – using four different test beds – the performance of 16 n-gram based metrics, 48 linguistic metrics and one combined metric from the state of the art.

Analyzing the reliability of evaluation metrics requires meta-evaluation criteria. In this respect, we identify important drawbacks of the standard meta-evaluation methods based on correlation with human judgements. In order to overcome these drawbacks, we then introduce six novel meta-evaluation criteria which represent different metric reliability dimensions. Our analysis indicates that: (i) both lexical and linguistic metrics have complementary advantages and different drawbacks; (ii) combining both kinds of metrics is a more effective and robust evaluation method across all meta-evaluation criteria.

In addition, we also perform a qualitative analysis of one hundred sentences that were incorrectly evaluated by state-of-the-art metrics. The analysis confirms that deep linguistic techniques are necessary to avoid the most common types of error.

Section 2 examines the state of the art Section 3 describes the test beds and metrics considered in our experiments. In Section 4 the correlation between human assessors and metrics is computed, with a discussion of its drawbacks. In Section 5 different quality aspects of metrics are analysed. Conclusions are drawn in the last section.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 306–314,
Suntec, Singapore, 2-7 August 2009.

2 Previous Work on Machine Translation Meta-Evaluation

Insofar as automatic evaluation metrics for machine translation have been proposed, different meta-evaluation frameworks have been gradually introduced. For instance, Papineni et al. (2001) introduced the BLEU metric and evaluated its reliability in terms of Pearson correlation with human assessments for adequacy and fluency judgements. With the aim of overcoming some of the deficiencies of BLEU, Doddington (2002) introduced the NIST metric. Metric reliability was also estimated in terms of correlation with human assessments, but over different document sources and for a varying number of references and segment sizes. Melamed et al. (2003) argued, at the time of introducing the GTM metric, that Pearson correlation coefficients can be affected by scale properties, and suggested, in order to avoid this effect, to use the non-parametric Spearman correlation coefficients instead.

Lin and Och (2004) experimented, unlike previous works, with a wide set of metrics, including NIST, WER (Nießen et al., 2000), PER (Tillmann et al., 1997), and variants of ROUGE, BLEU and GTM. They computed both Pearson and Spearman correlation, obtaining similar results in both cases. In a different work, Banerjee and Lavie (2005) argued that the measured reliability of metrics can be due to averaging effects but might not be robust across translations. In order to address this issue, they computed the translation-by-translation correlation with human judgements (i.e., correlation at the segment level).

All that metrics were based on n-gram overlap. But there is also extensive research focused on including linguistic knowledge in metrics (Owczarzak et al., 2006; Reeder et al., 2001; Liu and Gildea, 2005; Amigó et al., 2006; Mehay and Brew, 2007; Giménez and Màrquez, 2007; Owczarzak et al., 2007; Popovic and Ney, 2007; Giménez and Màrquez, 2008b) among others. In all these cases, metrics were also evaluated by means of correlation with human judgements.

In a different research line, several authors have suggested approaching automatic evaluation through the combination of individual metric scores. Among the most relevant let us cite research by Kulesza and Shieber (2004), Albrecht and Hwa (2007). But finding optimal metric combinations requires a meta-evaluation criterion. Most approaches again rely on correlation with human judgements. However, some of them measured the reliability of metric combinations in terms of their ability to discriminate between human translations and automatic ones (*human likeness*) (Amigó et al., 2005). .

In this work, we present a novel approach to meta-evaluation which is distinguished by the use of additional easily interpretable meta-evaluation criteria oriented to measure different aspects of metric reliability. We then apply this approach to find out about the advantages and challenges of including linguistic features in meta-evaluation criteria.

3 Metrics and Test Beds

3.1 Metric Set

For our study, we have compiled a rich set of metric variants at three linguistic levels: lexical, syntactic, and semantic. In all cases, translation quality is measured by comparing automatic translations against a set of human references.

At the lexical level, we have included several standard metrics, based on different similarity assumptions: edit distance (WER, PER and TER), lexical precision (BLEU and NIST), lexical recall (ROUGE), and F-measure (GTM and METEOR). At the syntactic level, we have used several families of metrics based on dependency parsing (DP) and constituency trees (CP). At the semantic level, we have included three different families which operate using named entities (NE), semantic roles (SR), and discourse representations (DR). A detailed description of these metrics can be found in (Giménez and Màrquez, 2007).

Finally, we have also considered ULC, which is a very simple approach to metric combination based on the unnormalized arithmetic mean of metric scores, as described by Giménez and Màrquez (2008a). ULC considers a subset of metrics which operate at several linguistic levels. This approach has proven very effective in recent evaluation campaigns. Metric computation has been carried out using the IQ*MT* Framework for Automatic MT Evaluation (Giménez, 2007)[1]. The simplicity of this approach (with no training of the metric weighting scheme) ensures that the potential advantages detected in our experiments are not due to overfitting effects.

[1] `http://www.lsi.upc.edu/~nlp/IQMT`

	2004		2005	
	AE	**CE**	**AE**	**CE**
#references	5	5	5	4
#systems$_{\text{assessed}}$	5	10	5+1	5
#cases$_{\text{assessed}}$	347	447	266	272

Table 1: NIST 2004/2005 MT Evaluation Campaigns. Test bed description

3.2 Test Beds

We use the test beds from the 2004 and 2005 NIST MT Evaluation Campaigns (Le and Przybocki, 2005)[2]. Both campaigns include two different translations exercises: Arabic-to-English ('AE') and Chinese-to-English ('CE'). Human assessments of adequacy and fluency, on a 1-5 scale, are available for a subset of sentences, each evaluated by two different human judges. A brief numerical description of these test beds is available in Table 1. The corpus AE05 includes, apart from five automatic systems, one human-aided system that is only used in our last experiment.

4 Correlation with Human Judgements

4.1 Correlation at the Segment vs. System Levels

Let us first analyze the correlation with human judgements for linguistic vs. n-gram based metrics. Figure 1 shows the correlation obtained by each automatic evaluation metric at system level (horizontal axis) versus segment level (vertical axis) in our test beds. Linguistic metrics are represented by grey plots, and black plots represent metrics based on n-gram overlap.

The most remarkable aspect is that there exists a certain trade-off between correlation at segment versus system level. In fact, this graph produces a negative Pearson correlation coefficient between system and segment levels of 0.44. In other words, depending on how the correlation is computed, the relative predictive power of metrics can swap. Therefore, we need additional meta-evaluation criteria in order to clarify the behavior of linguistic metrics as compared to n-gram based metrics.

However, there are some exceptions. Some metrics achieve high correlation at both levels. The first one is ULC (the circle in the plot), which combines both kind of metrics in a heuristic way (see Section 3.1). The metric nearest to ULC is

[2] http://www.nist.gov/speech/tests/mt

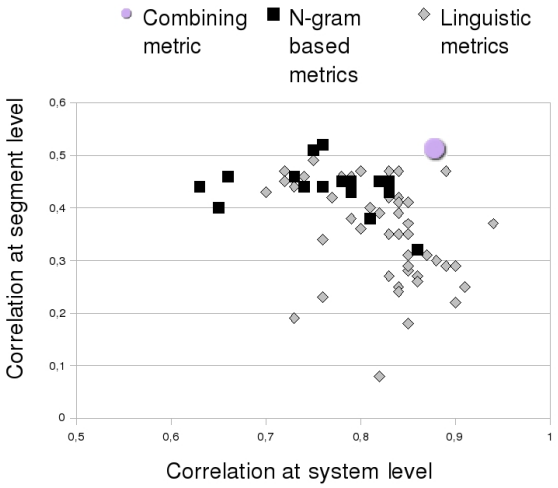


Figure 1: Averaged Pearson correlation at system vs. segment level over all test beds.

DP-O_r-$\star$, which computes lexical overlapping but on dependency relationships. These results are a first evidence of the advantages of combining metrics at several linguistic processing levels.

4.2 Drawbacks of Correlation-based Meta-evaluation

Although correlation with human judgements is considered the standard meta-evaluation criterion, it presents serious drawbacks. With respect to correlation at system level, the main problem is that the relative performance of different metrics changes almost randomly between testbeds. One of the reasons is that the number of assessed systems per testbed is usually low, and then correlation has a small number of samples to be estimated with. Usually, the correlation at system level is computed over no more than a few systems.

For instance, Table 2 shows the best 10 metrics in CE05 according to their correlation with human judges at the system level, and then the ranking they obtain in the AE05 testbed. There are substantial swaps between both rankings. Indeed, the Pearson correlation of both ranks is only 0.26. This result supports the intuition in (Banerjee and Lavie, 2005) that correlation at segment level is necessary to ensure the reliability of metrics in different situations.

However, the correlation values of metrics at segment level have also drawbacks related to their interpretability. Most metrics achieve a Pearson coefficient lower than 0.5. Figure 2 shows two possible relationships between human and metric

CE 2005		AE 2005	
SP-pNIST-5	0,88	SR-Mrv-_	0,95
SP-iobNIST-5	0,87	SP-pNIST-5	0,92
DR-STM-4	0,76	DR-Orp-_-b	0,91
DR-Orp-_	0,75	DP-HWC-r-4	0,89
DR-Orp-_-b	0,74	SP-cNIST-5	0,89
DP-HWC-r-4	0,73	SP-iobNIST-5	0,86
SP-cNIST-5	0,72	SR-Orv	0,82
SR-Mrv-_	0,71	SR-Orv-b	0,81
DR-STM-5	0,69	DR-STM-5	0,81
SR-Orv	0,67	DR-Orp-_	0,8
SR-Or-b	0,67	DR-STM-4	0,77
SR-Orv-b	0,67	SR-Or-b	0,73

Table 2: Metrics rankings according to correlation with human judgements using CE05 vs. AE05

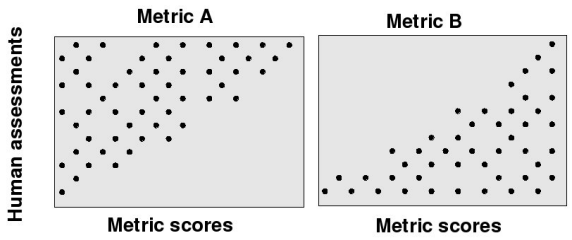


Figure 2: Human judgements and scores of two hypothetical metrics with Pearson correlation 0.5

produced scores. Both hypothetical metrics A and B would achieve a 0.5 correlation. In the case of Metric A, a high score implies a high human assessed quality, but not the reverse. This is the tendency hypothesized by Culy and Riehemann (2003). In the case of Metric B, the high scored translations can achieve both low or high quality according to human judges but low scores ensure low quality. Therefore, the same Pearson coefficient may hide very different behaviours. In this work, we tackle these drawbacks by defining more specific meta-evaluation criteria.

5 Alternatives to Correlation-based Meta-evaluation

We have seen that correlation with human judgements has serious limitations for metric evaluation. Therefore, we have focused on other aspects of metric reliability that have revealed differences between n-gram and linguistic based metrics:

1. Is the metric able to accurately reveal improvements between two systems?

2. Can we trust the metric when it says that a translation is very good or very bad?

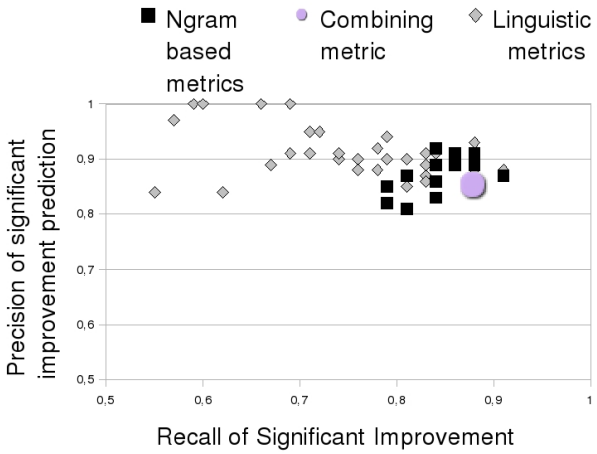


Figure 3: SIP versus SIR

3. Are metrics able to identify good translations which are dissimilar from the models?

We now discuss each of these aspects separately.

5.1 Ability of metrics to Reveal System Improvements

We now investigate to what extent a significant system improvement according to the metric implies a significant improvement according to human assessors, and viceversa. In other words: are the metrics able to detect any quality improvement? Is a metric score improvement a strong evidence of quality increase? Knowing that a metric has a 0.8 Pearson correlation at the system level or 0.5 at the segment level does not provide a direct answer to this question.

In order to tackle this issue, we compare metrics versus human assessments in terms of precision and recall over statistically significant improvements within all system pairs in the test beds. First, Table 3 shows the amount of significant improvements over human judgements according to the Wilcoxon statistical significant test ($\alpha \leq 0.025$). For instance, the testbed CE2004 consists of 10 systems, i.e. 45 system pairs; from these, in 40 cases (rightmost column) one of the systems significantly improves the other.

Now we would like to know, for every metric, if the pairs which are significantly different according to human judges are also the pairs which are significantly different according to the metric.

Based on these data, we define two meta-metrics: *Significant Improvement Precision* (SIP) and *Significant Improvement Recall* (SIR). SIP

	Systems	System pairs	Sig. imp.
CE$_{2004}$	10	45	40
AE$_{2004}$	5	10	8
CE$_{2005}$	5	10	4
AE$_{2005}$	5	10	6
Total	25	75	58

Table 3: System pairs with a significant difference according to human judgements (Wilcoxon test)

(precision) represents the reliability of improvements detected by metrics. SIR (recall) represents to what extent the metric is able to cover the significant improvements detected by humans. Let I_h be the set of significant improvements detected by human assessors and I_m the set detected by the metric m. Then:

$$\text{SIP} = \frac{|I_h \cap I_m|}{|I_m|} \qquad \text{SIR} = \frac{|I_h \cap I_m|}{|I_h|}$$

Figure 3 shows the SIR and SIP values obtained for each metric. Linguistic metrics achieve higher precision values but at the cost of an important recall decrease. Given that linguistic metrics require matching translation with references at additional linguistic levels, the significant improvements detected are more reliable (higher precision or SIP), but at the cost of recall over real significant improvements (lower SIR).

This result supports the behaviour predicted in (Giménez and Màrquez, 2009). Although linguistic metrics were motivated by the idea of modeling linguistic variability, the practical effect is that current linguistic metrics introduce additional restrictions (such as dependency tree overlap, for instance) for accepting automatic translations. Then they reward precision at the cost of recall in the evaluation process, and this explains the high correlation with human judgements at system level with respect to segment level.

All n-gram based metrics achieve SIP and SIR values between 0.8 and 0.9. This result suggests that n-gram based metrics are reasonably reliable for this purpose. Note that the combined metric, ULC (the circle in the figure), achieves results comparable to n-gram based metrics with this test[3]. That is, combining linguistic and n-gram based metrics preserves the good behavior of n-gram based metrics in this test.

[3]Notice that we just have 75 significant improvement samples, so small differences in SIP or SIR have no relevance

5.2 Reliability of High and Low Metric Scores

The issue tackled in this section is to what extent a very low or high score according to the metric is reliable for detecting extreme cases (very good or very bad translations). In particular, note that detecting wrong translations is crucial in order to analyze the system drawbacks.

In order to define an accuracy measure for the reliability of very low/high metric scores, it is necessary to define quality thresholds for both the human assessments and metric scales. Defining thresholds for manual scores is immediate (e.g., lower than 4/10). However, each automatic evaluation metric has its own scale properties. In order to solve scaling problems we will focus on equivalent rank positions: we associate the ith translation according to the metric ranking with the quality value manually assigned to the ith translation in the manual ranking.

Being $Q_h(t)$ and $Q_m(t)$ the human and metric assessed quality for the translation t, and being $\text{rank}_h(t)$ and $\text{rank}_m(t)$ the rank of the translation t according to humans and the metric, the normalized metric assessed quality is:

$$Q_{N_m}(t) = Q_h(t')| \ (\text{rank}_h(t') = \text{rank}_m(t))$$

In order to analyze the reliability of metrics when identifying wrong or high quality translations, we look for contradictory results between the metric and the assessments. In other words, we look for metric errors in which the quality estimated by the metric is low ($Q_{N_m}(t) \leq 3$) but the quality assigned by assessors is high ($Q_h(t) \geq 5$) or viceversa ($Q_{N_m}(t) \geq 7$ and $Q_h(t) \leq 4$).

The vertical axis in Figure 4 represents the ratio of errors in the set of low scored translations according to a given metric. The horizontal axis represents the ratio of errors over the set of high scored translations. The first observation is that all metrics are less reliable when they assign low scores (which corresponds with the situation A described in Section 4.2). For instance, the best metric erroneously assigns a low score in more than 20% of the cases. In general, the linguistic metrics do not improve the ability to capture wrong translations (horizontal axis in the figure). However, again, the combining metric ULC achieves the same reliability as the best n-gram based metric.

In order to check the robustness of these results, we computed the correlation of individual metric failures between test beds, obtaining 0.67 Pearson for the lowest correlated test bed pair (AE_{2004} and CE_{2005}) and 0.88 for the highest correlated pair (AE_{2004} and CE_{2004}).

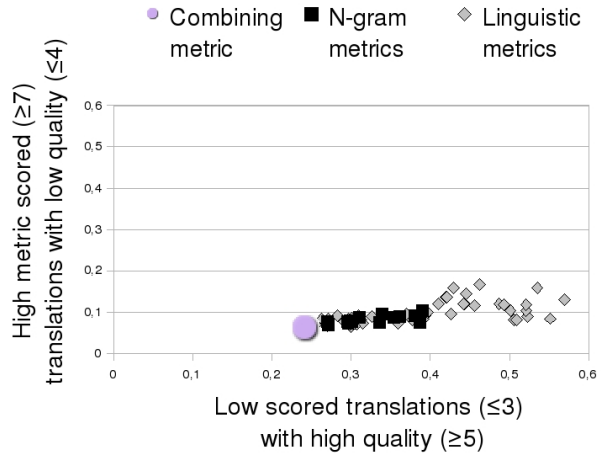


Figure 4: Counter sample ratio for high vs low metric scored translations

5.2.1 Analysis of Evaluation Samples

In order to shed some light on the reasons for the automatic evaluation failures when assigning low scores, we have manually analyzed cases in which a metric score is low but the quality according to humans is high ($Q_{N_m} \leq 3$ and $Q_h \geq 7$). We have studied 100 sentence evaluation cases from representatives of each metric family including: 1-PER, BLEU, DP-O_r-$\star$, GTM ($e = 2$), METEOR and ROUGE$_L$. The evaluation cases have been extracted from the four test beds. We have identified four main (non exclusive) failure causes:

Format issues, e.g. "*US* " vs "*United States*"). Elements such as abbreviations, acronyms or numbers which do not match the manual translation.

Pseudo-synonym terms, e.g. "*US* **Scheduled** *the Release*" vs. "*US* **set** *to Release*").) In most of these cases, synonymy can only be identified from the discourse context. Therefore, terminological resources (e.g., WordNet) are not enough to tackle this problem.

Non relevant information omissions, e.g. *"Thank you"* vs. *"Thank you very much"* or *"dollar"* vs. *"US dollar"*)). The translation system obviates some information which, in context, is not considered crucial by the human assessors. This effect is specially important in short sentences.

Incorrect structures that change the meaning while maintaining the same idea (e.g., "*Bush Praises NASA 's Mars Mission*" vs " *Bush praises nasa of Mars mission*").

Note that all of these kinds of failure - except formatting issues - require deep linguistic processing while n-gram overlap or even synonyms extracted from a standard ontology are not enough to deal with them. This conclusion motivates the incorporation of linguistic processing into automatic evaluation metrics.

5.3 Ability to Deal with Translations that are Dissimilar to References.

The results presented in Section 5.2 indicate that a high score in metrics tends to be highly related to truly good translations. This is due to the fact that a high word overlapping with human references is a reliable evidence of quality. However, in some cases the translations to be evaluated are not so similar to human references.

An example of this appears in the test bed NIST05AE which includes a human-aided system, LinearB (Callison-Burch, 2005). This system produces correct translations whose words do not necessarily overlap with references. On the other hand, a statistics based system tends to produce incorrect translations with a high level of lexical overlapping with the set of human references. This case was reported by Callison-Burch et al. (2006) and later studied by Giménez and Màrquez (2007). They found out that lexical metrics fail to produce reliable evaluation scores. They favor systems which share the expected reference sublanguage (e.g., statistical) and penalize those which do not (e.g., LinearB).

We can find in our test bed many instances in which the statistical systems obtain a metric score similar to the assisted system while achieving a lower mark according to human assessors. For instance, for the following translations, ROUGE$_L$ assigns a slightly higher score to the output of a statistical system which contains a lot of grammatical and syntactical failures.

Human assisted system: *The Chinese President made unprecedented criticism of the leaders of Hong Kong after political failings in the former British colony on Monday* . Human assessment=8.5.

Statistical system: *Chinese President Hu Jintao today unprecedented criticism to the leaders of Hong Kong wake political and financial failure in the former British colony.* Human assessment=3.

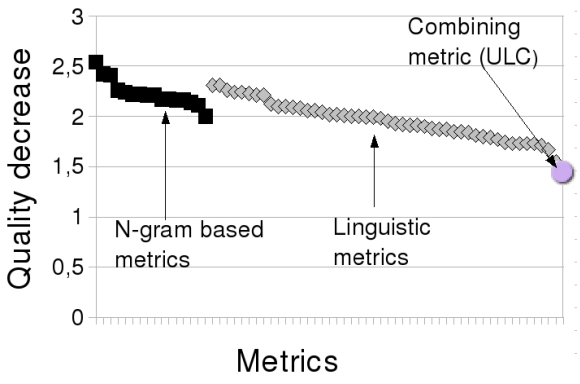


Figure 5: Maximum translation quality decreasing over similarly scored translation pairs.

In order to check the metric resistance to be cheated by translations with high lexical overlapping, we estimate the quality decrease that we could cause if we optimized the human-aided translations according to the automatic metric. For this, we consider in each translation case c, the worse automatic translation t that equals or improves the human-aided translation t_h according to the automatic metric m. Formally the averaged quality decrease is:

$$\text{Quality decrease}(m) = \text{Avg}_c(\max_t(Q_h(t_h) - Q_h(t) | Q_m(t_h) \leq Q_m(t)))$$

Figure 5 illustrates the results obtained. All metrics are suitable to be cheated, assigning similar or higher scores to worse translations. However, linguistic metrics are more resistant. In addition, the combined metric ULC obtains the best results, better than both linguistic and n-gram based metrics. Our conclusion is that including higher linguistic levels in metrics is relevant to prevent ungrammatical n-gram matching to achieve similar scores than grammatical constructions.

5.4 The Oracle System Test

In order to obtain additional evidence about the usefulness of combining evaluation metrics at different processing levels, let us consider the following situation: given a set of reference translations we want to train a combined system that takes the most appropriate translation approach for each text segment. We consider the set of translations system presented in each competition as the translation approaches pool. Then, the upper bound on the quality of the combined system is given by the predictive power of the employed automatic evaluation metric. This upper bound is obtained by selecting the highest scored translation t according to a specific metric m for each translation case c. The Oracle System Test (OST) consists of computing the averaged human assessed quality Q_h of the selected translations according to human assessors across all cases. Formally:

$$\text{OST}(m) = \text{Avg}_c(Q_h(\text{Argmax}_t(Q_m(t)) | t \in c))$$

We use the sum of adequacy and fluency, both in a 1-5 scale, as a global quality measure. Thus, OST scores are in a 2-10 range. In summary, the OST represents the best combined system that could be trained according to a specific automatic evaluation metric.

Metric	OST
maxOST	**6.72**
ULC	5.79
ROUGE$_W$	5.71
DP-O_r-$\star$	5.70
CP-O_c-$\star$	5.70
NIST	5.70
randOST	5.20
minOST	3.67

Table 4: Metrics ranked according to the Oracle System Test

Table 4 shows OST values obtained for the best metrics. In the table we have also included a random, a maximum (always pick the best translation according to humans) and a minimum (always pick the worse translation according to human) OST for all [4]. The most remarkable result in Table 4 is that metrics are closer to the random baseline than to the upperbound (maximum OST). This result confirms the idea that an improvement on metric reliability could contribute considerably to the systems optimization process. However, the key point is that the combined metric, ULC, improves all the others (5.79 vs. 5.71), indicating the importance of combining n-gram and linguistic features.

6 Conclusions

Our experiments show that, on one hand, traditional n-gram based metrics are more or equally

[4] In all our experiments, the meta-metric values are computed over each test bed independently before averaging in order to assign equal relevance to the four possible contexts (test beds)

reliable for estimating the translation quality at the segment level, for predicting significant improvement between systems and for detecting poor and excellent translations.

On the other hand, linguistically motivated metrics improve n-gram metrics in two ways: (i) they achieve higher correlation with human judgements at system level and (ii) they are more resistant to reward poor translations with high word overlapping with references.

The underlying phenomenon is that, rather than managing the linguistics variability, linguistic based metrics introduce additional restrictions for assigning high scores. This effect decreases the recall over significant system improvements achieved by n-gram based metrics and does not solve the problem of detecting wrong translations. Linguistic metrics, however, are more difficult to cheat.

In general, the greatest pitfall of metrics is the low reliability of low metric values. Our qualitative analysis of evaluated sentences has shown that deeper linguistic techniques are necessary to overcome the important surface differences between acceptable automatic translations and human references.

But our key finding is that combining both kinds of metrics gives top performance according to every meta-evaluation criteria. In addition, our Combined System Test shows that, when training a combined translation system, using metrics at several linguistic processing levels improves substantially the use of individual metrics.

In summary, our results motivate: (i) working on new linguistic metrics for overcoming the barrier of linguistic variability and (ii) performing new metric combining schemes based on linear regression over human judgements (Kulesza and Shieber, 2004), training models over human/machine discrimination (Albrecht and Hwa, 2007) or non parametric methods based on reference to reference distances (Amigó et al., 2005).

Acknowledgments

This work has been partially supported by the Spanish Government, project INES/Text-Mess. We are indebted to the three ACL anonymous reviewers which provided detailed suggestions to improve our work.

References

Joshua Albrecht and Rebecca Hwa. 2007. Regression for Sentence-Level MT Evaluation with Pseudo References. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 296–303.

Enrique Amigó, Julio Gonzalo, Anselmo Pe nas, and Felisa Verdejo. 2005. QARLA: a Framework for the Evaluation of Automatic Summarization. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 280–289.

Enrique Amigó, Jesús Giménez, Julio Gonzalo, and Lluís Màrquez. 2006. MT Evaluation: Human-Like vs. Human Acceptable. In *Proceedings of the Joint 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 17–24.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the Role of BLEU in Machine Translation Research. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Chris Callison-Burch. 2005. Linear B system description for the 2005 NIST MT evaluation exercise. In *Proceedings of the NIST 2005 Machine Translation Evaluation Workshop*.

Christopher Culy and Susanne Z. Riehemann. 2003. The Limits of N-gram Translation Evaluation Metrics. In *Proceedings of MT-SUMMIT IX*, pages 1–8.

George Doddington. 2002. Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics. In *Proceedings of the 2nd International Conference on Human Language Technology*, pages 138–145.

Jesús Giménez and Lluís Màrquez. 2007. Linguistic Features for Automatic Evaluation of Heterogeneous MT Systems. In *Proceedings of the ACL Workshop on Statistical Machine Translation*, pages 256–264.

Jesús Giménez and Lluís Màrquez. 2008a. Heterogeneous Automatic MT Evaluation Through Non-Parametric Metric Combinations. In *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP)*, pages 319–326.

Jesús Giménez and Lluís Màrquez. 2008b. On the Robustness of Linguistic Features for Automatic MT Evaluation. (Under submission).

Jesús Giménez and Lluís Màrquez. 2009. On the Robustness of Syntactic and Semantic Features for Automatic MT Evaluation. In *Proceedings of the 4th Workshop on Statistical Machine Translation (EACL 2009)*.

Jesús Giménez. 2007. IQMT v 2.0. Technical Manual (LSI-07-29-R). Technical report, TALP Research Center. LSI Department. `http://www.lsi.upc.edu/~nlp/IQMT/IQMT.v2.1.pdf`.

Alex Kulesza and Stuart M. Shieber. 2004. A learning approach to improving sentence-level MT evaluation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, pages 75–84.

Audrey Le and Mark Przybocki. 2005. NIST 2005 machine translation evaluation official results. In *Official release of automatic evaluation scores for all submissions, August*.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ding Liu and Daniel Gildea. 2005. Syntactic Features for Evaluation of Machine Translation. In *Proceedings of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*, pages 25–32.

Dennis Mehay and Chris Brew. 2007. BLEUATRE: Flattening Syntactic Dependencies for MT Evaluation. In *Proceedings of the 11th Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*.

I. Dan Melamed, Ryan Green, and Joseph P. Turian. 2003. Precision and Recall of Machine Translation. In *Proceedings of the Joint Conference on Human Language Technology and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*.

Sonja Nießen, Franz Josef Och, Gregor Leusch, and Hermann Ney. 2000. An Evaluation Tool for Machine Translation: Fast Evaluation for MT Research. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC)*.

Karolina Owczarzak, Declan Groves, Josef Van Genabith, and Andy Way. 2006. Contextual Bitext-Derived Paraphrases in Automatic MT Evaluation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 148–155.

Karolina Owczarzak, Josef van Genabith, and Andy Way. 2007. Labelled Dependencies in Machine Translation Evaluation. In *Proceedings of the ACL Workshop on Statistical Machine Translation*, pages 104–111.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation, RC22176. Technical report, IBM T.J. Watson Research Center.

Maja Popovic and Hermann Ney. 2007. Word Error Rates: Decomposition over POS classes and Applications for Error Analysis. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 48–55, Prague, Czech Republic, June. Association for Computational Linguistics.

Florence Reeder, Keith Miller, Jennifer Doyon, and John White. 2001. The Naming of Things and the Confusion of Tongues: an MT Metric. In *Proceedings of the Workshop on MT Evaluation "Who did what to whom?" at Machine Translation Summit VIII*, pages 55–59.

Christoph Tillmann, Stefan Vogel, Hermann Ney, A. Zubiaga, and H. Sawaf. 1997. Accelerated DP based Search for Statistical Translation. In *Proceedings of European Conference on Speech Communication and Technology*.

A Syntax-Driven Bracketing Model for Phrase-Based Translation

Deyi Xiong, Min Zhang, Aiti Aw and Haizhou Li
Human Language Technology
Institute for Infocomm Research
1 Fusionopolis Way, #21-01 South Connexis, Singapore 138632
{dyxiong, mzhang, aaiti, hli}@i2r.a-star.edu.sg

Abstract

Syntactic analysis influences the way in which the source sentence is translated. Previous efforts add syntactic constraints to phrase-based translation by directly rewarding/punishing a hypothesis whenever it matches/violates source-side constituents. We present a new model that automatically learns syntactic constraints, including but not limited to constituent matching/violation, from training corpus. The model brackets a source phrase as to whether it satisfies the learnt syntactic constraints. The bracketed phrases are then translated as a whole unit by the decoder. Experimental results and analysis show that the new model outperforms other previous methods and achieves a substantial improvement over the baseline which is not syntactically informed.

1 Introduction

The phrase-based approach is widely adopted in statistical machine translation (SMT). It segments a source sentence into a sequence of phrases, then translates and reorder these phrases in the target. In such a process, original phrase-based decoding (Koehn et al., 2003) does not take advantage of any linguistic analysis, which, however, is broadly used in rule-based approaches. Since it is not linguistically motivated, original phrase-based decoding might produce ungrammatical or even wrong translations. Consider the following Chinese fragment with its parse tree:

Src: [把 [[7月 11日]$_{\text{NP}}$ [设立 [为 [航海 节]$_{\text{NP}}$]$_{\text{PP}}$]$_{\text{VP}}$]$_{\text{IP}}$]$_{\text{VP}}$

Ref: established July 11 as Sailing Festival day

Output: [to/把 [⟨[set up/设立 [for/为 navigation/航海]] on July 11/7月11日⟩ knots/节]]

The output is generated from a phrase-based system which does not involve any syntactic analysis. Here we use "[]" (straight orientation) and "⟨⟩" (inverted orientation) to denote the common structure of the source fragment and its translation found by the decoder. We can observe that the decoder inadequately breaks up the second NP phrase and translates the two words "航海" and "节" separately. However, the parse tree of the source fragment constrains the phrase "航海 节" to be translated as a unit.

Without considering syntactic constraints from the parse tree, the decoder makes wrong decisions not only on phrase movement but also on the lexical selection for the multi-meaning word "节"[1]. To avert such errors, the decoder can fully respect linguistic structures by only allowing syntactic constituent translations and reorderings. This, unfortunately, significantly jeopardizes performance (Koehn et al., 2003; Xiong et al., 2008) because by integrating syntactic constraint into decoding as a hard constraint, it simply prohibits any other useful non-syntactic translations which violate constituent boundaries.

To better leverage syntactic constraint yet still allow non-syntactic translations, Chiang (2005) introduces a count for each hypothesis and accumulates it whenever the hypothesis exactly matches syntactic boundaries on the source side. On the contrary, Marton and Resnik (2008) and Cherry (2008) accumulate a count whenever hypotheses violate constituent boundaries. These constituent matching/violation counts are used as a feature in the decoder's log-linear model and their weights are tuned via minimal error rate training (MERT) (Och, 2003). In this way, syntactic constraint is integrated into decoding as a soft constraint to enable the decoder to reward hypotheses that respect syntactic analyses or to pe-

[1]This word can be translated into "section", "festival", and "knot" in different contexts.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 315–323,
Suntec, Singapore, 2-7 August 2009.

nalize hypotheses that violate syntactic structures.

Although experiments show that this constituent matching/violation counting feature achieves significant improvements on various language-pairs, one issue is that matching syntactic analysis can not always guarantee a good translation, and violating syntactic structure does not always induce a bad translation. Marton and Resnik (2008) find that some constituency types favor matching the source parse while others encourage violations. Therefore it is necessary to integrate more syntactic constraints into phrase translation, not just the constraint of constituent matching/violation.

The other issue is that during decoding we are more concerned with the question of phrase cohesion, i.e. whether the current phrase can be translated as a unit or not within particular syntactic contexts (Fox, 2002)[2], than that of constituent matching/violation. Phrase cohesion is one of the main reasons that we introduce syntactic constraints (Cherry, 2008). If a source phrase remains contiguous after translation, we refer this type of phrase **bracketable**, otherwise **unbracketable**. It is more desirable to translate a bracketable phrase than an unbracketable one.

In this paper, we propose a syntax-driven bracketing (SDB) model to predict whether a phrase (a sequence of contiguous words) is bracketable or not using rich syntactic constraints. We parse the source language sentences in the word-aligned training corpus. According to the word alignments, we define bracketable and unbracketable instances. For each of these instances, we automatically extract relevant syntactic features from the source parse tree as bracketing evidences. Then we tune the weights of these features using a maximum entropy (ME) trainer. In this way, we build two bracketing models: 1) a unary SDB model (UniSDB) which predicts whether an independent phrase is bracketable or not; and 2) a binary SDB model(BiSDB) which predicts whether two neighboring phrases are bracketable. Similar to previous methods, our SDB model is integrated into the decoder's log-linear model as a feature so that we can inherit the idea of soft constraints.

In contrast to the constituent matching/violation counting (CMVC) (Chiang, 2005; Marton and Resnik, 2008; Cherry, 2008), our SDB model has the following advantages

- The SDB model automatically learns syntactic constraints from training data while the CMVC uses manually defined syntactic constraints: constituency matching/violation. In our SDB model, each learned syntactic feature from bracketing instances can be considered as a syntactic constraint. Therefore we can use thousands of syntactic constraints to guide phrase translation.

- The SDB model maintains and protects the strength of the phrase-based approach in a better way than the CMVC does. It is able to reward non-syntactic translations by assigning an adequate probability to them if these translations are appropriate to particular syntactic contexts on the source side, rather than always punish them.

We test our SDB model against the baseline which doest not use any syntactic constraints on Chinese-to-English translation. To compare with the CMVC, we also conduct experiments using (Marton and Resnik, 2008)'s XP+. The XP+ accumulates a count for each hypothesis whenever it violates the boundaries of a constituent with a label from {NP, VP, CP, IP, PP, ADVP, QP, LCP, DNP}. The XP+ is the best feature among all features that Marton and Resnik use for Chinese-to-English translation. Our experimental results display that our SDB model achieves a substantial improvement over the baseline and significantly outperforms XP+ according to the BLEU metric (Papineni et al., 2002). In addition, our analysis shows further evidences of the performance gain from a different perspective than that of BLEU.

The paper proceeds as follows. In section 2 we describe how to learn bracketing instances from a training corpus. In section 3 we elaborate the syntax-driven bracketing model, including feature generation and the integration of the SDB model into phrase-based SMT. In section 4 and 5, we present our experiments and analysis. And we finally conclude in section 6.

2 The Acquisition of Bracketing Instances

In this section, we formally define the bracketing instance, comprising two types namely binary bracketing instance and unary bracketing instance.

[2]Here we expand the definition of phrase to include both syntactic and non-syntactic phrases.

We present an algorithm to automatically extract these bracketing instances from word-aligned bilingual corpus where the source language sentences are parsed.

Let c and e be the source sentence and the target sentence, W be the word alignment between them, T be the parse tree of c. We define a **binary bracketing instance** as a tuple $\langle b, \tau(c_{i..j}), \tau(c_{j+1..k}), \tau(c_{i..k})\rangle$ where $b \in \{bracketable, unbracketable\}$, $c_{i..j}$ and $c_{j+1..k}$ are two neighboring source phrases and $\tau(T, s)$ ($\tau(s)$ for short) is a subtree function which returns the minimal subtree covering the source sequence s from the source parse tree T. Note that $\tau(c_{i..k})$ includes both $\tau(c_{i..j})$ and $\tau(c_{j+1..k})$. For the two neighboring source phrases, the following conditions are satisfied:

$$\exists e_{u..v}, e_{p..q} \in e \;\; s.t.$$
$$\forall (m,n) \in W, i \le m \le j \leftrightarrow u \le n \le v \quad (1)$$
$$\forall (m,n) \in W, j+1 \le m \le k \leftrightarrow p \le n \le q \quad (2)$$

The above (1) means that there exists a target phrase $e_{u..v}$ aligned to $c_{i..j}$ and (2) denotes a target phrase $e_{p..q}$ aligned to $c_{j+1..k}$. If $e_{u..v}$ and $e_{p..q}$ are neighboring to each other or all words between the two phrases are aligned to null, we set $b = bracketable$, otherwise $b = unbracketable$. From a binary bracketing instance, we derive a **unary bracketing instance** $\langle b, \tau(c_{i..k})\rangle$, ignoring the subtrees $\tau(c_{i..j})$ and $\tau(c_{j+1..k})$.

Let n be the number of words of c. If we extract all potential bracketing instances, there will be $o(n^2)$ unary instances and $o(n^3)$ binary instances. To keep the number of bracketing instances tractable, we only record 4 representative bracketing instances for each index j: 1) the bracketable instance with the minimal $\tau(c_{i..k})$, 2) the bracketable instance with the maximal $\tau(c_{i..k})$, 3) the unbracketable instance with the minimal $\tau(c_{i..k})$, and 4) the unbracketable instance with the maximal $\tau(c_{i..k})$.

Figure 1 shows the algorithm to extract bracketing instances. Line 3-11 find all potential bracketing instances for each $(i, j, k) \in c$ but only keep 4 bracketing instances for each index j: two minimal and two maximal instances. This algorithm learns binary bracketing instances, from which we can derive unary bracketing instances.

1: **Input**: sentence pair (c, e), the parse tree T of c and the word alignment W between c and e
2: $\Re := \emptyset$
3: **for** each $(i, j, k) \in c$ **do**
4: **if** There exist a target phrase $e_{u..v}$ aligned to $c_{i..j}$ and $e_{p..q}$ aligned to $c_{j+1..k}$ **then**
5: Get $\tau(c_{i..j})$, $\tau(c_{j+1..k})$, and $\tau(c_{i..k})$
6: Determine b according to the relationship between $e_{u..v}$ and $e_{p..q}$
7: **if** $\tau(c_{i..k})$ is currently maximal or minimal **then**
8: Update bracketing instances for index j
9: **end if**
10: **end if**
11: **end for**
12: **for** each $j \in c$ **do**
13: $\Re := \Re \cup \{$bracketing instances from $j\}$
14: **end for**
15: **Output**: bracketing instances $\Re$

Figure 1: Bracketing Instances Extraction Algorithm.

3 The Syntax-Driven Bracketing Model

3.1 The Model

Our interest is to automatically detect phrase bracketing using rich contextual information. We consider this task as a binary-class classification problem: whether the current source phrase s is bracketable (b) within particular syntactic contexts ($\tau(s)$). If two neighboring sub-phrases s_1 and s_2 are given, we can use more inner syntactic contexts to complete this binary classification task.

We construct the syntax-driven bracketing model within the maximum entropy framework. A unary SDB model is defined as:

$$P_{UniSDB}(b|\tau(s), T) = \frac{exp(\sum_i \theta_i h_i(b, \tau(s), T)}{\sum_b exp(\sum_i \theta_i h_i(b, \tau(s), T)} \quad (3)$$

where $h_i \in \{0, 1\}$ is a binary feature function which we will describe in the next subsection, and θ_i is the weight of h_i. Similarly, a binary SDB model is defined as:

$$P_{BiSDB}(b|\tau(s_1), \tau(s_2), \tau(s), T) = \frac{exp(\sum_i \theta_i h_i(b, \tau(s_1), \tau(s_2), \tau(s), T)}{\sum_b exp(\sum_i \theta_i h_i(b, \tau(s_1), \tau(s_2), \tau(s), T)} \quad (4)$$

The most important advantage of ME-based SDB model is its capacity of incorporating more fine-grained contextual features besides the binary feature that detects constituent boundary violation or matching. By employing these features, we can investigate the value of various syntactic constraints in phrase translation.

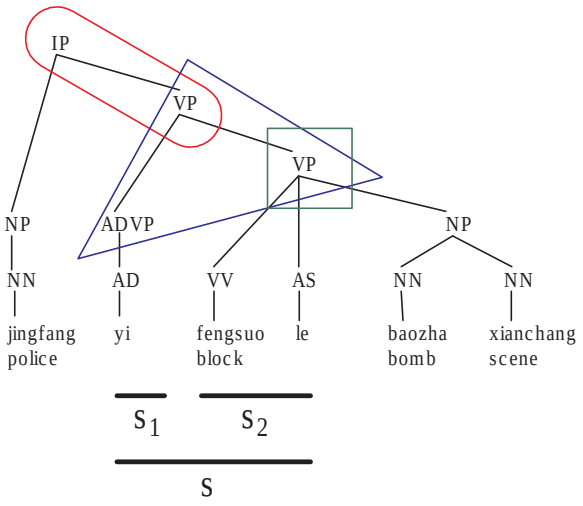


Figure 2: Illustration of syntax-driven features used in SDB. Here we only show the features for the source phrase s. The triangle, rounded rectangle and rectangle denote the rule feature, path feature and constituent boundary matching feature respectively.

3.2 Syntax-Driven Features

Let s be the source phrase in question, s_1 and s_2 be the two neighboring sub-phrases. $\sigma(.)$ is the root node of $\tau(.)$. The SDB model exploits various syntactic features as follows.

- Rule Features (RF)
  We use the CFG rules of $\sigma(s)$, $\sigma(s_1)$ and $\sigma(s_2)$ as features. These features capture syntactic "horizontal context" which demonstrates the expansion trend of the source phrase s, s_1 and s_2 on the parse tree.

  In figure 2, the CFG rule "ADVP→AD", "VP→VV AS NP", and "VP→ADVP VP" are used as features for s_1, s_2 and s respectively.

- Path Features (PF)
  The tree path $\sigma(s_1)..\sigma(s)$ connecting $\sigma(s_1)$ and $\sigma(s)$, $\sigma(s_2)..\sigma(s)$ connecting $\sigma(s_2)$ and $\sigma(s)$, and $\sigma(s)..\rho$ connecting $\sigma(s)$ and the root node ρ of the whole parse tree are used as features. These features provide syntactic "vertical context" which shows the generation history of the source phrases on the parse tree.

  In figure 2, the path features are "ADVP VP", "VP VP" and "VP IP" for s_1, s_2 and s respectively.

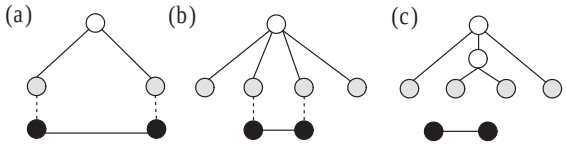


Figure 3: Three scenarios of the relationship between phrase boundaries and constituent boundaries. The gray circles are constituent boundaries while the black circles are phrase boundaries.

- Constituent Boundary Matching Features (CBMF)
  These features are to capture the relationship between a source phrase s and $\tau(s)$ or $\tau(s)$'s subtrees. There are three different scenarios[3]: 1) **exact match**, where s exactly matches the boundaries of $\tau(s)$ (figure 3(a)), 2) **inside match**, where s exactly spans a sequence of $\tau(s)$'s subtrees (figure 3(b)), and 3) **crossing**, where s crosses the boundaries of one or two subtrees of $\tau(s)$ (figure 3(c)). In the case of 1) or 2), we set the value of this feature to $\sigma(s)$-M or $\sigma(s)$-I respectively. When s crosses the boundaries of the sub-constituent ϵ_l on s's left, we set the value to $\sigma(\epsilon_l)$-LC; If s crosses the boundaries of the sub-constituent ϵ_r on s's right, we set the value to $\sigma(\epsilon_r)$-RC; If both, we set the value to $\sigma(\epsilon_l)$-LC-$\sigma(\epsilon_r)$-RC.

  Let's revisit the Figure 2. The source phrase s_1 exactly matches the constituent ADVP, therefore CBMF is "ADVP-M". The source phrase s_2 exactly spans two sub-trees VV and AS of VP, therefore CBMF is "VP-I". Finally, the source phrase s cross boundaries of the lower VP on the right, therefore CBMF is "VP-RC".

3.3 The Integration of the SDB Model into Phrase-Based SMT

We integrate the SDB model into phrase-based SMT to help decoder perform syntax-driven phrase translation. In particular, we add a

[3]The three scenarios that we define here are similar to those in (Lü et al., 2002).

new feature into the log-linear translation model: $P_{SDB}(b|T, \tau(.))$. This feature is computed by the SDB model described in equation (3) or equation (4), which estimates a probability that a source span is to be translated as a unit within particular syntactic contexts. If a source span can be translated as a unit, the feature will give a higher probability even though this span violates boundaries of a constituent. Otherwise, a lower probability is given. Through this additional feature, we want the decoder to prefer hypotheses that translate source spans which can be translated as a unit, and avoids translating those which are discontinuous after translation. The weight of this new feature is tuned via MERT, which measures the extent to which this feature should be trusted.

In this paper, we implement the SDB model in a state-of-the-art phrase-based system which adapts a binary bracketing transduction grammar (BTG) (Wu, 1997) to phrase translation and reordering, described in (Xiong et al., 2006). Whenever a BTG merging rule ($s \rightarrow [s_1\ s_2]$ or $s \rightarrow \langle s_1\ s_2 \rangle$) is used, the SDB model gives a probability to the span s covered by the rule, which estimates the extent to which the span is bracketable. For the unary SDB model, we only consider the features from $\tau(s)$. For the binary SDB model, we use all features from $\tau(s_1)$, $\tau(s_2)$ and $\tau(s)$ since the binary SDB model is naturally suitable to the binary BTG rules.

The SDB model, however, is not only limited to phrase-based SMT using BTG rules. Since it is applied on a source span each time, any other hierarchical phrase-based or syntax-based system that translates source spans recursively or linearly, can adopt the SDB model.

4 Experiments

We carried out the MT experiments on Chinese-to-English translation, using (Xiong et al., 2006)'s system as our baseline system. We modified the baseline decoder to incorporate our SDB models as descried in section 3.3. In order to compare with Marton and Resnik's approach, we also adapted the baseline decoder to their XP+ feature.

4.1 Experimental Setup

In order to obtain syntactic trees for SDB models and XP+, we parsed source sentences using a lexicalized PCFG parser (Xiong et al., 2005). The parser was trained on the Penn Chinese Treebank with an F1 score of 79.4%.

All translation models were trained on the FBIS corpus. We removed 15,250 sentences, for which the Chinese parser failed to produce syntactic parse trees. To obtain word-level alignments, we ran GIZA++ (Och and Ney, 2000) on the remaining corpus in both directions, and applied the "grow-diag-final" refinement rule (Koehn et al., 2005) to produce the final many-to-many word alignments. We built our four-gram language model using Xinhua section of the English Gigaword corpus (181.1M words) with the SRILM toolkit (Stolcke, 2002).

For the efficiency of MERT, we built our development set (580 sentences) using sentences not exceeding 50 characters from the NIST MT-02 set. We evaluated all models on the NIST MT-05 set using case-sensitive BLEU-4. Statistical significance in BLEU score differences was tested by paired bootstrap re-sampling (Koehn, 2004).

4.2 SDB Training

We extracted 6.55M bracketing instances from our training corpus using the algorithm shown in figure 1, which contains 4.67M bracketable instances and 1.89M unbracketable instances. From extracted bracketing instances we generated syntax-driven features, which include 73,480 rule features, 153,614 path features and 336 constituent boundary matching features. To tune weights of features, we ran the MaxEnt toolkit (Zhang, 2004) with iteration number being set to 100 and Gaussian prior to 1 to avoid overfitting.

4.3 Results

We ran the MERT module with our decoders to tune the feature weights. The values are shown in Table 1. The P_{SDB} receives the largest feature weight, 0.29 for UniSDB and 0.38 for BiSDB, indicating that the SDB models exert a nontrivial impact on decoder.

In Table 2, we present our results. Like (Marton and Resnik, 2008), we find that the XP+ feature obtains a significant improvement of 1.08 BLEU over the baseline. However, using all syntax-driven features described in section 3.2, our SDB models achieve larger improvements of up to 1.67 BLEU. The binary SDB (BiSDB) model statistically significantly outperforms Marton and Resnik's XP+ by an absolute improvement of 0.59 (relatively 2%). It is also marginally better than the unary SDB model.

System	Features									
	$P(c\|e)$	$P(e\|c)$	$P_w(c\|e)$	$P_w(e\|c)$	$P_{lm}(e)$	$P_r(e)$	Word	Phr.	XP+	P_{SDB}
Baseline	0.041	0.030	0.006	0.065	0.20	0.35	0.19	-0.12	—	—
XP+	0.002	0.049	0.046	0.044	0.17	0.29	0.16	0.12	-0.12	—
UniSDB	0.023	0.051	0.055	0.012	0.21	0.20	0.12	0.04	—	0.29
BiSDB	0.016	0.032	0.027	0.013	0.13	0.23	0.08	0.09	—	0.38

Table 1: Feature weights obtained by MERT on the development set. The first 4 features are the phrase translation probabilities in both directions and the lexical translation probabilities in both directions. P_{lm} = language model; P_r = MaxEnt-based reordering model; Word = word bonus; Phr = phrase bonus.

System	BLEU-n	n-gram Precision							
	4	1	2	3	4	5	6	7	8
Baseline	0.2612	0.71	0.36	0.18	0.10	0.054	0.030	0.016	0.009
XP+	0.2720**	0.72	0.37	0.19	0.11	0.060	0.035	0.021	0.012
UniSDB	**0.2762**+**	0.72	0.37	0.20	0.11	0.062	0.035	0.020	0.011
BiSDB	**0.2779**++**	0.72	0.37	0.20	0.11	0.065	0.038	0.022	0.014

Table 2: Results on the test set. **: significantly better than baseline ($p < 0.01$). + or ++: significantly better than Marton and Resnik's XP+ ($p < 0.05$ or $p < 0.01$, respectively).

5 Analysis

In this section, we present analysis to perceive the influence mechanism of the SDB model on phrase translation by studying the effects of syntax-driven features and differences of 1-best translation outputs.

5.1 Effects of Syntax-Driven Features

We conducted further experiments using individual syntax-driven features and their combinations. Table 3 shows the results, from which we have the following key observations.

- The constituent boundary matching feature (CBMF) is a very important feature, which by itself achieves significant improvement over the baseline (up to 1.13 BLEU). Both our CBMF and Marton and Resnik's XP+ feature focus on the relationship between a source phrase and a constituent. Their significant contribution to the improvement implies that this relationship is an important syntactic constraint for phrase translation.

- Adding more features, such as path feature and rule feature, achieves further improvements. This demonstrates the advantage of using more syntactic constraints in the SDB model, compared with Marton and Resnik's XP+.

Features	BLEU-4	
	UniSDB	BiSDB
PF + RF	0.2555	0.2644*@@
PF	0.2596	0.2671**@@
CBMF	0.2678**	0.2725**@
RF + CBMF	0.2737**	0.2780**++@@
PF + CBMF	0.2755**+	0.2782**++@$^-$
RF + PF + CBMF	0.2762**+	0.2779**++

Table 3: Results of different feature sets. * or **: significantly better than baseline ($p < 0.05$ or $p < 0.01$, respectively). + or ++: significantly better than XP+ ($p < 0.05$ or $p < 0.01$, respectively). @$^-$: almost significantly better than its UniSDB counterpart ($p < 0.075$). @ or @@: significantly better than its UniSDB counterpart ($p < 0.05$ or $p < 0.01$, respectively).

- In most cases, the binary SDB is constantly significantly better than the unary SDB, suggesting that inner contexts are useful in predicting phrase bracketing.

5.2 Beyond BLEU

We want to further study the happenings after we integrate the constraint feature (our SDB model and Marton and Resnik's XP+) into the log-linear translation model. In particular, we want to investigate: to what extent syntactic constraints change translation outputs? And in what direction the changes take place? Since BLEU is not sufficient

System	CCM Rate (%)
Baseline	43.5
XP+	74.5
BiSDB	72.4

Table 4: Consistent constituent matching rates reported on 1-best translation outputs.

	CCM Rates (%)				
System	<6	6-10	11-15	16-20	>20
XP+	75.2	70.9	71.0	76.2	82.2
BiSDB	69.3	74.7	74.2	80.0	85.6

Table 6: Consistent constituent matching rates for structures with different spans.

to provide such insights, we introduce a new statistical metric which measures the proportion of syntactic constituents [4] whose boundaries are consistently matched by decoder during translation. This proportion, which we call **consistent constituent matching (CCM) rate** , reflects the extent to which the translation output respects the source parse tree.

In order to calculate this rate, we output translation results as well as phrase alignments found by decoders. Then for each multi-branch constituent c_i^j spanning from i to j on the source side, we check the following conditions.

- If its boundaries i and j are aligned to phrase segmentation boundaries found by decoder.
- If all target phrases inside c_i^j's target span [5] are aligned to the source phrases within c_i^j and not to the phrases outside c_i^j.

If both conditions are satisfied, the constituent c_i^j is consistently matched by decoder.

Table 4 shows the consistent constituent matching rates. Without using any source-side syntactic information, the baseline obtains a low CCM rate of 43.53%, indicating that the baseline decoder violates the source parse tree more than it respects the source structure. The translation output described in section 1 is actually generated by the baseline decoder, where the second NP phrase boundaries are violated.

By integrating syntactic constraints into decoding, we can see that both Marton and Resnik's XP+ and our SDB model achieve a significantly higher constituent matching rate, suggesting that they are more likely to respect the source structure. The examples in Table 5 show that the decoder is able to generate better translations if it is faithful to the source parse tree by using syntactic constraints.

We further conducted a deep comparison of translation outputs of BiSDB vs. XP+ with regard to constituent matching and violation. We found two significant differences that may explain why our BiSDB outperforms XP+. First, although the overall CCM rate of XP+ is higher than that of BiSDB, BiSDB obtains higher CCM rates for long-span structures than XP+ does, which are shown in Table 6. Generally speaking, violations of long-span constituents have a more negative impact on performance than short-span violations if these violations are toxic. This explains why BiSDB achieves relatively higher precision improvements for higher n-grams over XP+, as shown in Table 3.

Second, compared with XP+ that only punishes constituent boundary violations, our SDB model is able to encourage violations if these violations are done on bracketable phrases. We observed in many cases that by violating constituent boundaries BiSDB produces better translations than XP+ does, which on the contrary matches these boundaries. Still consider the example shown in section 1. The following translations are found by XP+ and BiSDB respectively.

XP+: [to/把 ⟨[set up/设立 [for the/为 [navigation/航海 section/节]]] on July 11/7月11日⟩]

BiSDB: [to/把 ⟨[[set up/设立 a/为] [marine/航海 festival/节]] on July 11/7月11日⟩]

XP+ here matches all constituent boundaries while BiSDB violates the PP constituent to translate the non-syntactic phrase "设立 为". Table 7 shows more examples. From these examples, we clearly see that appropriate violations are helpful and even necessary for generating better translations. By allowing appropriate violations to translate non-syntactic phrases according to particular syntactic contexts, our SDB model better inherits the strength of phrase-based approach than XP+.

[4] We only consider multi-branch constituents.

[5] Given a phrase alignment $P = \{c_f^g \leftrightarrow e_p^q\}$, if the segmentation within c_i^j defined by P is $c_i^j = c_{i_1}^{j_1}...c_{i_k}^{j_k}$, and $c_{i_r}^{j_r} \leftrightarrow e_{u_r}^{v_r} \in P, 1 \leq r \leq k$, we define the **target span** of c_i^j as a pair where the first element is $min(e_{u_1}...e_{u_k})$ and the second element is $max(e_{v_1}...e_{v_k})$, similar to (Fox, 2002).

Src:	[[为 [印度 洋 灾区 民众]$_{NP}$]$_{PP}$ [奉献 [自己]$_{NP}$ [一 份 爱心]$_{NP}$]$_{VP}$]$_{VP}$
Ref:	show their loving hearts to people in the Indian Ocean disaster areas
Baseline:	⟨love/爱心 [for the/为 ⟨[people/民众 [to/奉献 [own/自己 a report/一份]]]⟩ ⟨in/灾区 the Indian Ocean/印度洋⟩]⟩
XP+:	⟨[contribute/奉献 [its/自己 [part/一份 love/爱心]]] [for/为 ⟨the people/民众 ⟨in/灾区 the Indian Ocean/印度洋⟩⟩]⟩
BiSDB:	⟨[[[contribute/奉献 its/自己] part/一份] love/爱心] [for/为 ⟨the people/民众 ⟨in/灾区 the Indian Ocean印度洋⟩⟩]⟩
Src:	[五角大厦 [已]$_{ADVP}$ [派遣 [[二十 架]$_{QP}$ 飞机]$_{NP}$ [至 南亚]$_{PP}$]$_{VP}$]$_{IP}$ [，]$_{PU}$ [其中 包括...]$_{IP}$
Ref:	The Pentagon has dispatched 20 airplanes to South Asia, including...
Baseline:	[[The Pentagon/五角大厦 has sent/已派遣] [⟨[to/至 [[South Asia/南亚 ,/，] including/其中包括]] [20/二十 plane/架飞机]⟩]]
XP+:	[The Pentagon/五角大厦 [has/已 [sent/派遣 [[20/二十 planes/架飞机] [to/至 South Asia/南亚]]]]] [,/，[including/其中包括...]]
BiSDB:	[The Pentagon/五角大厦 [has sent/已派遣 [[20/二十 planes/架飞机] [to/至 South Asia/南亚]]] [,/， [including/其中包括...]]

Table 5: Translation examples showing that both XP+ and BiSDB produce better translations than the baseline, which inappropriately violates constituent boundaries (within underlined phrases).

Src:	[[在 [[[美国国务院 与 鲍尔]$_{NP}$ [短暂]$_{ADJP}$ [会谈]$_{NP}$]$_{NP}$ 后]$_{LCP}$]$_{PP}$ 表示]$_{VP}$
Ref:	said after a brief discussion with Powell at the US State Department
XP+:	[⟨after/后 ⟨⟨[a brief/短暂 meeting/会谈] [with/与 Powell/鲍尔]⟩ [in/在 the US State Department/美国国务院]⟩ said/表示]
BiSDB:	⟨said after/后 表示 ⟨[a brief/短暂 meeting/会谈] ⟨ with Powell/与 鲍尔 [at/在 the State Department of the United States/美国国务院]⟩⟩⟩
Src:	[向 [[建立 [未来 民主 政治]$_{NP}$]$_{VP}$]$_{IP}$]$_{PP}$ [迈出 了 [关键性 的 一 步]$_{NP}$]$_{VP}$
Ref:	took a key step towards building future democratic politics
XP+:	⟨[a/了 [key/关键性 step/的一步]] ⟨forward/迈出 [to/向 [a/建立 [future/未来 political democracy/民主政治]]]⟩⟩
BiSDB:	⟨[made a/迈出了 [key/关键性 step/的一步]] [towards establishing a/向 建立 ⟨democratic politics/民主政治 in the future/未来⟩]⟩

Table 7: Translation examples showing that BiSDB produces better translations than XP+ via appropriate violations of constituent boundaries (within double-underlined phrases).

6 Conclusion

In this paper, we presented a syntax-driven bracketing model that automatically learns bracketing knowledge from training corpus. With this knowledge, the model is able to predict whether source phrases can be translated together, regardless of matching or crossing syntactic constituents. We integrate this model into phrase-based SMT to increase its capacity of linguistically motivated translation without undermining its strengths. Experiments show that our model achieves substantial improvements over baseline and significantly outperforms (Marton and Resnik, 2008)'s XP+.

Compared with previous constituency feature, our SDB model is capable of incorporating more syntactic constraints, and rewarding necessary violations of the source parse tree. Marton and Resnik (2008) find that their constituent constraints are sensitive to language pairs. In the future work, we will use other language pairs to test our models so that we could know whether our method is language-independent.

References

Colin Cherry. 2008. Cohesive Phrase-based Decoding for Statistical Machine Translation. In *Proceedings of ACL*.

David Chiang. 2005. A Hierarchical Phrase-based Model for Statistical Machine Translation. In *Proceedings of ACL*, pages 263–270.

David Chiang, Yuval Marton and Philip Resnik. 2008. Online Large-Margin Training of Syntactic and Structural Translation Features. In *Proceedings of EMNLP*.

Heidi J. Fox 2002. Phrasal Cohesion and Statistical Machine Translation. In *Proceedings of EMNLP*, pages 304–311.

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical Phrase-based Translation. In *Proceedings of HLT-NAACL*.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of EMNLP*.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne and David Talbot. 2005. Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In *International Workshop on Spoken Language Translation*.

Yajuan Lü, Sheng Li, Tiezhun Zhao and Muyun Yang. 2002. Learning Chinese Bracketing Knowledge Based on a Bilingual Language Model. In *Proceedings of COLING*.

Yuval Marton and Philip Resnik. 2008. Soft Syntactic Constraints for Hierarchical Phrase-Based Translation. In *Proceedings of ACL*.

Franz Josef Och and Hermann Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of ACL 2000*.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of ACL 2003*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatically Evaluation of Machine Translation. In *Proceedings of ACL*.

Andreas Stolcke. 2002. SRILM - an Extensible Language Modeling Toolkit. In *Proceedings of International Conference on Spoken Language Processing*, volume 2, pages 901-904.

Dekai Wu. 1997. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3):377-403.

Deyi Xiong, Shuanglong Li, Qun Liu, Shouxun Lin, Yueliang Qian. 2005. Parsing the Penn Chinese Treebank with Semantic Knowledge. In *Proceedings of IJCNLP*, Jeju Island, Korea.

Deyi Xiong, Qun Liu and Shouxun Lin. 2006. Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. In *Proceedings of ACL-COLING 2006*.

Deyi Xiong, Min Zhang, Aiti Aw, and Haizhou Li. 2008. Linguistically Annotated BTG for Statistical Machine Translation. In *Proceedings of COLING 2008*.

Le Zhang. 2004. Maximum Entropy Modeling Tooklkit for Python and C++. Available at http://homepages.inf.ed.ac.uk/s0450736 /maxent_toolkit.html.

Topological Ordering of Function Words in Hierarchical Phrase-based Translation

Hendra Setiawan[1] and **Min-Yen Kan**[2] and **Haizhou Li**[3] and **Philip Resnik**[1]
[1]University of Maryland Institute for Advanced Computer Studies
[2]School of Computing, National University of Singapore
[3]Human Language Technology, Institute for Infocomm Research, Singapore
{hendra,resnik}@umiacs.umd.edu,
kanmy@comp.nus.edu.sg, hli@i2r.a-star.edu.sg

Abstract

Hierarchical phrase-based models are attractive because they provide a consistent framework within which to characterize both local and long-distance reorderings, but they also make it difficult to distinguish many implausible reorderings from those that are linguistically plausible. Rather than appealing to annotation-driven syntactic modeling, we address this problem by observing the influential role of function words in determining syntactic structure, and introducing soft constraints on function word relationships as part of a standard log-linear hierarchical phrase-based model. Experimentation on Chinese-English and Arabic-English translation demonstrates that the approach yields significant gains in performance.

1 Introduction

Hierarchical phrase-based models (Chiang, 2005; Chiang, 2007) offer a number of attractive benefits in statistical machine translation (SMT), while maintaining the strengths of phrase-based systems (Koehn et al., 2003). The most important of these is the ability to model long-distance reordering efficiently. To model such a reordering, a hierarchical phrase-based system demands no additional parameters, since long and short distance reorderings are modeled identically using synchronous context free grammar (SCFG) rules. The same rule, depending on its topological ordering – i.e. its position in the hierarchical structure – can affect both short and long spans of text. Interestingly, hierarchical phrase-based models provide this benefit without making any linguistic commitments beyond the structure of the model.

However, the system's lack of linguistic commitment is also responsible for one of its greatest drawbacks. In the absence of linguistic knowledge, the system models linguistic structure using an SCFG that contains only one type of nonterminal symbol[1]. As a result, the system is susceptible to the *overgeneration* problem: the grammar may suggest more reordering choices than appropriate, and many of those choices lead to ungrammatical translations.

Chiang (2005) hypothesized that incorrect reordering choices would often correspond to hierarchical phrases that violate syntactic boundaries in the source language, and he explored the use of a "constituent feature" intended to reward the application of hierarchical phrases which respect source language syntactic categories. Although this did not yield significant improvements, Marton and Resnik (2008) and Chiang et al. (2008) extended this approach by introducing soft syntactic constraints similar to the constituent feature, but more fine-grained and sensitive to distinctions among syntactic categories; these led to substantial improvements in performance. Zollman et al. (2006) took a complementary approach, constraining the application of hierarchical rules to respect syntactic boundaries in the target language syntax. Whether the focus is on constraints from the source language or the target language, the main ingredient in both previous approaches is the idea of constraining the spans of hierarchical phrases to respect syntactic boundaries.

In this paper, we pursue a different approach to improving reordering choices in a hierarchical phrase-based model. Instead of biasing the model toward hierarchical phrases whose spans respect syntactic boundaries, we focus on the topological ordering of phrases in the hierarchical structure. We conjecture that since incorrect reordering choices correspond to incorrect topological orderings, boosting the probability of correct topo-

[1]In practice, one additional nonterminal symbol is used in "glue rules". This is not relevant in the present discussion.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 324–332,
Suntec, Singapore, 2-7 August 2009.

logical ordering choices should improve the system. Although related to previous proposals (correct topological orderings lead to correct spans and vice versa), our proposal incorporates broader context and is structurally more aware, since we look at the topological ordering of a phrase relative to other phrases, rather than modeling additional properties of a phrase in isolation. In addition, our proposal requires no monolingual parsing or linguistically informed syntactic modeling for either the source or target language.

The key to our approach is the observation that we can approximate the topological ordering of hierarchical phrases via the topological ordering of function words. We introduce a statistical reordering model that we call the *pairwise dominance model*, which characterizes reorderings of phrases around a pair of function words. In modeling function words, our model can be viewed as a successor to the function words-centric reordering model (Setiawan et al., 2007), expanding on the previous approach by modeling pairs of function words rather than individual function words in isolation.

The rest of the paper is organized as follows. In Section 2, we briefly review hierarchical phrase-based models. In Section 3, we first describe the overgeneration problem in more detail with a concrete example, and then motivate our idea of using the topological ordering of function words to address the problem. In Section 4, we develop our idea by introducing the pairwise dominance model, expressing function word relationships in terms of what we call the the *dominance* predicate. In Section 5, we describe an algorithm to estimate the parameters of the dominance predicate from parallel text. In Sections 6 and 7, we describe our experiments, and in Section 8, we analyze the output of our system and lay out a possible future direction. Section 9 discusses the relation of our approach to prior work and Section 10 wraps up with our conclusions.

2 Hierarchical Phrase-based System

Formally, a hierarchical phrase-based SMT system is based on a weighted synchronous context free grammar (SCFG) with one type of nonterminal symbol. Synchronous rules in hierarchical phrase-based models take the following form:

$$X \rightarrow \langle \gamma, \alpha, \sim \rangle \quad (1)$$

where X is the nonterminal symbol and γ and α are strings that contain the combination of lexical items and nonterminals in the source and target languages, respectively. The $\sim$ symbol indicates that nonterminals in γ and α are synchronized through co-indexation; i.e., nonterminals with the same index are aligned. Nonterminal correspondences are strictly one-to-one, and in practice the number of nonterminals on the right hand side is constrained to at most two, which must be separated by lexical items.

Each rule is associated with a score that is computed via the following log linear formula:

$$w(X \rightarrow \langle \gamma, \alpha, \sim \rangle) = \prod_i f_i^{\lambda_i} \quad (2)$$

where f_i is a feature describing one particular aspect of the rule and λ_i is the corresponding weight of that feature. Given $\tilde{e}$ and $\tilde{f}$ as the source and target phrases associated with the rule, typical features used are rule's translation probability $P_{trans}(\tilde{f}|\tilde{e})$ and its inverse $P_{trans}(\tilde{e}|\tilde{f})$, the lexical probability $P_{lex}(\tilde{f}|\tilde{e})$ and its inverse $P_{lex}(\tilde{e}|\tilde{f})$. Systems generally also employ a word penalty, a phrase penalty, and target language model feature. (See (Chiang, 2005) for more detailed discussion.) Our pairwise dominance model will be expressed as an additional rule-level feature in the model.

Translation of a source sentence e using hierarchical phrase-based models is formulated as a search for the most probable derivation D^* whose source side is equal to e:

$$D^* = \text{argmax } P(D), \text{where source}(D)\text{=}e.$$

$D = X^i, i \in 1...|D|$ is a set of rules following a certain topological ordering, indicated here by the use of the superscript.

3 Overgeneration and Topological Ordering of Function Words

The use of only one type of nonterminal allows a flexible permutation of the topological ordering of the same set of rules, resulting in a huge number of possible derivations from a given source sentence. In that respect, the overgeneration problem is not new to SMT: Bracketing Transduction Grammar (BTG) (Wu, 1997) uses a single type of nonterminal and is subject to overgeneration problems, as well.[2]

[2]Note, however, that overgeneration in BTG can be viewed as a feature, not a bug, since the formalism was origi-

The problem may be less severe in hierarchical phrase-based MT than in BTG, since lexical items on the rules' right hand sides often limit the span of nonterminals. Nonetheless overgeneration of reorderings is still problematic, as we illustrate using the hypothetical Chinese-to-English example in Fig. 1.

Suppose we want to translate the Chinese sentence in Fig. 1 into English using the following set of rules:

1. $X_a \rightarrow \langle$ 电脑 和 X_1, computers and $X_1 \rangle$
2. $X_b \rightarrow \langle X_1$ 是 X_2, X_1 are $X_2 \rangle$
3. $X_c \rightarrow \langle$ 手机 , cell phones $\rangle$
4. $X_d \rightarrow \langle X_1$ 的 发明 , inventions of $X_1 \rangle$
5. $X_e \rightarrow \langle$ 上个世纪 , the last century $\rangle$

Co-indexation of nonterminals on the right hand side is indicated by subscripts, and for our examples the label of the nonterminal on the left hand side is used as the rule's unique identifier. To correctly translate the sentence, a hierarchical phrase-based system needs to model the subject noun phrase, object noun phrase and copula constructions; these are captured by rules X_a, X_d and X_b respectively, so this set of rules represents a hierarchical phrase-based system that can be used to correctly translate the Chinese sentence. Note that the Chinese word order is correctly preserved in the subject (X_a) as well as copula constructions (X_b), and correctly inverted in the object construction (X_d).

However, although it can generate the correct translation in Fig. 2, the grammar has no mechanism to prevent the generation of an incorrect translation like the one illustrated in Fig. 3. If we contrast the topological ordering of the rules in Fig. 2 and Fig. 3, we observe that the difference is small but quite significant. Using precede symbol ($\prec$) to indicate the first operand immediately dominates the second operand in the hierarchical structure, the topological orderings in Fig. 2 and Fig. 3 are $X_a \prec X_b \prec X_c \prec X_d \prec X_e$ and $X_d \prec X_a \prec X_b \prec X_c \prec X_e$, respectively. The only difference is the topological ordering of X_d: in Fig. 2, it appears below most of the other hierarchical phrases, while in Fig. 3, it appears above all the other hierarchical phrases.

nally introduced for bilingual analysis rather than generation of translations.

Modeling the topological ordering of hierarchical phrases is computationally prohibitive, since there are literally millions of hierarchical rules in the system's automatically-learned grammar and millions of possible ways to order their application. To avoid this computational problem and still model the topological ordering, we propose to use the topological ordering of function words as a practical approximation. This is motivated by the fact that function words tend to carry crucial syntactic information in sentences, serving as the "glue" for content-bearing phrases. Moreover, the positional relationships between function words and content phrases tends to be fixed (e.g., in English, prepositions invariably precede their object noun phrase), at least for the languages we have worked with thus far.

In the Chinese sentence above, there are three function words involved: the conjunction 和 (and), the copula 是 (are), and the noun phrase marker 的 (of).[3] Using the function words as approximate representations of the rules in which they appear, the topological ordering of hierarchical phrases in Fig. 2 is 和(and) $\prec$ 是(are) $\prec$ 的(of), while that in Fig. 3 is 的(of) $\prec$ 和(and) $\prec$ 是(are).[4] We can distinguish the correct and incorrect reordering choices by looking at this simple information. In the correct reordering choice, 的(of) appears at the lower level of the hierarchy while in the incorrect one, 的(of) appears at the highest level of the hierarchy.

4 Pairwise Dominance Model

Our example suggests that we may be able to improve the translation model's sensitivity to correct versus incorrect reordering choices by modeling the topological ordering of function words. We do so by introducing a predicate capturing the *dominance* relationship in a derivation between pairs of neighboring function words.[5]

Let us define a predicate $d(Y', Y'')$ that takes two function words as input and outputs one of

[3] We use the term "noun phrase marker" here in a general sense, meaning that in this example it helps tell us that the phrase is part of an NP, not as a technical linguistic term. It serves in other grammatical roles, as well. Disambiguating the syntactic roles of function words might be a particularly useful thing to do in the model we are proposing; this is a question for future research.

[4] Note that for expository purposes, we designed our simple grammar to ensure that these function words appear in separate rules.

[5] Two function words are considered neighbors iff no other function word appears between them in the source sentence.

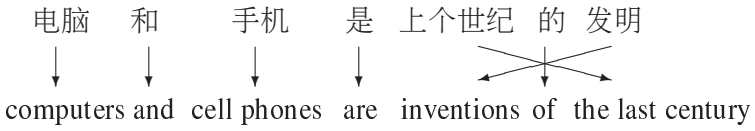


Figure 1: A running example of Chinese-to-English translation.

$X_a \Rightarrow \langle$ 电脑 和 X_b, computers and $X_b \rangle$
$\Rightarrow \langle$ 电脑 和 X_c 是 X_d, computers and X_c are $X_d \rangle$
$\Rightarrow \langle$ 电脑 和 手机 是 X_d, computers and cell phones are $X_d \rangle$
$\Rightarrow \langle$ 电脑 和 手机 是 X_e 的 发明 , computers and cell phones are inventions of $X_e \rangle$
$\Rightarrow \langle$ 电脑 和 手机 是 上个世纪 的 发明 , computers and cell phones are inventions of the last century$\rangle$

Figure 2: The derivation that leads to the correct translation

$X_d \Rightarrow \langle X_a$ 的 发明 , inventions of $X_a \rangle$
$\Rightarrow \langle$电脑 和 X_b 的 发明 , inventions of computers and $X_b \rangle$
$\Rightarrow \langle$电脑 和 X_c 是 X_e 的 发明 , inventions of computers and X_c are $X_e \rangle$
$\Rightarrow \langle$电脑 和 手机 是 X_e 的 发明 , inventions of computers and cell phones are $X_e \rangle$
$\Rightarrow \langle$电脑 和 手机 是 上个世纪 的 发明 , inventions of computers and cell phones are the last century$\rangle$

Figure 3: The derivation that leads to the incorrect translation

four values: {leftFirst, rightFirst, dontCare, neither}, where Y' appears to the left of Y'' in the source sentence. The value leftFirst indicates that in the derivation's topological ordering, Y' precedes Y'' (i.e. Y' dominates Y'' in the hierarchical structure), while rightFirst indicates that Y'' dominates Y'. In Fig. 2, $d(Y', Y'') =$ leftFirst for $Y' =$ the copula 是 (are) and $Y'' =$ the noun phrase marker 的 (of).

The dontCare and neither values capture two additional relationships: dontCare indicates that the topological ordering of the function words is flexible, and neither indicates that the topological ordering of the function words is disjoint. The former is useful in cases where the hierarchical phrases suggest the same kind of reordering, and therefore restricting their topological ordering is not necessary. This is illustrated in Fig. 2 by the pair 和(and) and the copula 是(are), where putting either one above the other does not change the final word order. The latter is useful in cases where the two function words do not share a same parent.

Formally, this model requires several changes in the design of the hierarchical phrase-based system.

1. To facilitate topological ordering of function words, the hierarchical phrases must be subcategorized with function words. Taking X_b in Fig. 2 as a case in point, subcategorization using function words would yield:[6]

$$X_b(\text{是} \prec \text{的}) \rightarrow X_c \text{ 是 } X_d(\text{的}) \quad (3)$$

The subcategorization (indicated by the information in parentheses following the nonterminal) propagates the function word 是(are) of X_b to the higher level structure together with the function word 的(of) of X_d. This propagation process generalizes to other rules by maintaining the ordering of the function words according to their appearance in the source sentence. Note that the subcategorized nonterminals often resemble genuine syntactic categories, for instance X(的) can frequently be interpreted as a noun phrase.

2. To facilitate the computation of the dominance relationship, the coindexing in synchronized rules (indicated by the $\sim$ symbol in Eq. 1) must be expanded to include information not only about the nonterminal correspondences but also about the alignment of the lexical items. For example, adding lexical alignment information to rule X_d would yield:

$$X_d \rightarrow \langle X_1\text{的}_2\text{发明}_3,\ \text{inventions}_3\ \text{of}_2\ X_1 \rangle \quad (4)$$

[6]The target language side is concealed for clarity.

The computation of the dominance relationship using this alignment information will be discussed in detail in the next section.

Again taking X_b in Fig. 2 as a case in point, the dominance feature takes the following form:

$$f_{dom}(X_b) \approx dom(d(是, 的)|是, 的)) \quad (5)$$

$$dom(d(Y_L, Y_R)|Y_L, Y_R)) \quad (6)$$

where the probability of 是 $\prec$ 的 is estimated according to the probability of d(是, 的).

In practice, both 是(are) and 的(of) may appear together in one same rule. In such a case, a dominance score is not calculated since the topological ordering of the two function words is unambiguous. Hence, in our implementation, a dominance score is only calculated at the points where the topological ordering of the hierarchical phrases needs to be resolved, i.e. the two function words always come from two different hierarchical phrases.

5 Parameter Estimation

Learning the dominance model involves extracting d values for every pair of neighboring function words in the training bitext. Such statistics are not directly observable in parallel corpora, so estimation is needed. Our estimation method is based on two facts: (1) the topological ordering of hierarchical phrases is tightly coupled with the span of the hierarchical phrases, and (2) the span of a hierarchical phrase at a higher level is always a superset of the span of all other hierarchical phrases at the lower level of its substructure. Thus, to establish soft estimates of dominance counts, we utilize alignment information available in the rule together with the consistent alignment heuristic (Och and Ney, 2004) traditionally used to guess phrase alignments.

Specifically, we define the span of a function word as a maximal, consistent alignment in the source language that either starts from or ends with the function word. (Requiring that spans be maximal ensures their uniqueness.) We will refer to such spans as Maximal Consistent Alignments (MCA). Note that each function word has two such Maximal Consistent Alignments: one that ends with the function word (MCA_R)and another that starts from the function word (MCA_L).

Y' Y''	left-First	right-First	dont-Care	neither
和 (and) 是 (are)	0.11	0.16	**0.68**	0.05
是 (are) 的 (of)	**0.57**	0.15	0.06	0.22

Table 1: The distribution of the dominance values of the function words involved in Fig. 1. The value with the highest probability is in **bold**.

Given two function words Y' and Y'', with Y' preceding Y'', we define the value of d by examining the MCAs of the two function words.

$$d(Y', Y'') = \begin{cases} \textsf{leftFirst}, & Y' \notin \text{MCA}_R(Y'') \wedge Y'' \in \text{MCA}_L(Y') \\ \textsf{rightFirst}, & Y' \in \text{MCA}_R(Y'') \wedge Y'' \notin \text{MCA}_L(Y') \\ \textsf{dontCare}, & Y' \in \text{MCA}_R(Y'') \wedge Y'' \in \text{MCA}_L(Y') \\ \textsf{neither}, & Y' \notin \text{MCA}_R(Y'') \wedge Y'' \notin \text{MCA}_L(Y') \end{cases} \quad (6)$$

Fig. 4a illustrates the leftFirst dominance value where the intersection of the MCAs contains only the second function word (的(of)). Fig. 4b illustrates the dontCare value, where the intersection contains both function words. Similarly, rightFirst and neither are represented by an intersection that contains only Y', or by an empty intersection, respectively. Once all the d values are counted, the pairwise dominance model of neighboring function words can be estimated simply from counts using maximum likelihood. Table 1 illustrates estimated dominance values that correctly resolve the topological ordering for our running example.

6 Experimental Setup

We tested the effect of introducing the pairwise dominance model into hierarchical phrase-based translation on Chinese-to-English and Arabic-to-English translation tasks, thus studying its effect in two languages where the use of function words differs significantly. Following Setiawan et al. (2007), we identify function words as the N most frequent words in the corpus, rather than identifying them according to linguistic criteria; this approximation removes the need for any additional language-specific resources. We report results for $N = 32, 64, 128, 256, 512, 1024, 2048$.[7] For

[7] We observe that even $N = 2048$ represents less than 1.5% and 0.8% of the words in the Chinese and Arabic vocabularies, respectively. The validity of the frequency-based strategy, relative to linguistically-defined function words, is discussed in Section 8

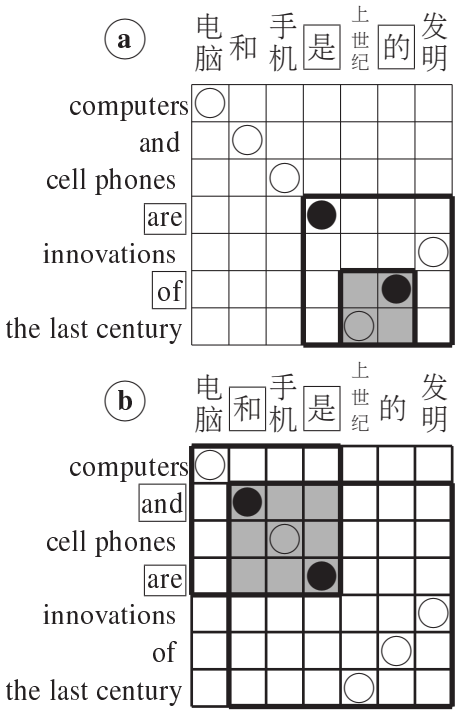


Figure 4: Illustrations for: a) the leftFirst value, and b) the dontCare value. Thickly bordered boxes are MCAs of the function words while solid circles are the alignment points of the function words. The gray boxes are the intersections of the two MCAs.

all experiments, we report performance using the BLEU score (Papineni et al., 2002), and we assess statistical significance using the standard bootstrapping approach introduced by (Koehn, 2004).

Chinese-to-English experiments. We trained the system on the NIST MT06 Eval corpus excluding the UN data (approximately 900K sentence pairs). For the language model, we used a 5-gram model with modified Kneser-Ney smoothing (Kneser and Ney, 1995) trained on the English side of our training data as well as portions of the Gigaword v2 English corpus. We used the NIST MT03 test set as the development set for optimizing interpolation weights using minimum error rate training (MERT; (Och and Ney, 2002)). We carried out evaluation of the systems on the NIST 2006 evaluation test (MT06) and the NIST 2008 evaluation test (MT08). We segmented Chinese as a preprocessing step using the Harbin segmenter (Zhao et al., 2001).

Arabic-to-English experiments. We trained the system on a subset of 950K sentence pairs from the NIST MT08 training data, selected by "subsampling" from the full training data using a method proposed by Kishore Papineni (personal communication). The subsampling algorithm selects sentence pairs from the training data in a way that seeks reasonable representation for all n-grams appearing in the test set. For the language model, we used a 5-gram model trained on the English portion of the whole training data plus portions of the Gigaword v2 corpus. We used the NIST MT03 test set as the development set for optimizing the interpolation weights using MERT. We carried out the evaluation of the systems on the NIST 2006 evaluation set (MT06) and the NIST 2008 evaluation set (MT08). Arabic source text was preprocessed by separating clitics, the definiteness marker, and the future tense marker from their stems.

7 Experimental Results

Chinese-to-English experiments. Table 2 summarizes the results of our Chinese-to-English experiments. These results confirm that the pairwise dominance model can significantly increase performance as measured by the BLEU score, with a consistent pattern of results across the MT06 and MT08 test sets. Modeling $N = 32$ drops the performance marginally below baseline, suggesting that perhaps there are not enough words for the pairwise dominance model to work with. Doubling the number of words ($N = 64$) produces a small gain, and defining the pairwise dominance model using $N = 128$ most frequent words produces a statistically significant 1-point gain over the baseline ($p < 0.01$). Larger values of N yield statistically significant performance above the baseline, but without further improvements over $N = 128$.

Arabic-to-English experiments. Table 3 summarizes the results of our Arabic-to-English experiments. This set of experiments shows a pattern consistent with what we observed in Chinese-to-English translation, again generally consistent across MT06 and MT08 test sets although modeling a small number of lexical items ($N = 32$) brings a marginal improvement over the baseline. In addition, we again find that the pairwise dominance model with $N = 128$ produces the most significant gain over the baseline in the MT06, although, interestingly, modeling a much larger number of lexical items ($N = 2048$) yields the strongest improvement for the MT08 test set.

	MT06	MT08
baseline	30.58	24.08
+$dom(N = 32)$	30.43	23.91
+$dom(N = 64)$	30.96	24.45
+$dom(N = 128)$	**31.59**	**24.91**
+$dom(N = 256)$	**31.24**	24.26
+$dom(N = 512)$	**31.33**	24.39
+$dom(N = 1024)$	**31.22**	**24.79**
+$dom(N = 2048)$	30.75	23.92

Table 2: Experimental results on Chinese-to-English translation with the pairwise dominance model (dom) of different N. The baseline (the first line) is the original hierarchical phrase-based system. Statistically significant results ($p < 0.01$) over the baseline are in **bold**.

	MT06	MT08
baseline	41.56	40.06
+$dom(N = 32)$	41.66	40.26
+$dom(N = 64)$	**42.03**	**40.73**
+$dom(N = 128)$	**42.66**	**41.08**
+$dom(N = 256)$	**42.28**	40.69
+$dom(N = 512)$	41.97	**40.95**
+$dom(N = 1024)$	42.05	40.55
+$dom(N = 2048)$	**42.48**	**41.47**

Table 3: Experimental results on Arabic-to-English translation with the pairwise dominance model (dom) of different N. The baseline (the first line) is the original hierarchical phrase-based system. Statistically significant results over the baseline ($p < 0.01$) are in **bold**.

8 Discussion and Future Work

The results in both sets of experiments show consistently that we have achieved a significant gains by modeling the topological ordering of function words. When we visually inspect and compare the outputs of our system with those of the baseline, we observe that improved BLEU score often corresponds to visible improvements in the subjective translation quality. For example, the translations for the Chinese sentence "军情$_1$ 观察$_2$:$_3$ 伊朗$_4$ 在$_5$ 美军$_6$ 空袭$_7$ 下$_8$ 能$_9$ 撑$_{10}$ 多$_{11}$ 久$_{12}$?$_{13}$", taken from Chinese MT06 test set, are as follows (co-indexing subscripts represent reconstructed word alignments):

- baseline: "military$_1$ intelligence$_2$ under_observation$_8$ in$_5$ u.s.$_6$ air_raids$_7$:$_3$ iran$_4$ to$_9$ how$_{11}$ long$_{12}$?$_{13}$ "
- *+dom(N=128)*: " military$_1$ survey$_2$:$_3$ how$_{11}$ long$_{12}$ iran$_4$ under$_8$ air_strikes$_7$ of_the_u.s$_6$ can$_9$ hold_out$_{10}$?$_{13}$ "

In addition to some lexical translation errors (e.g. 美军$_6$ should be translated to U.S. Army), the baseline system also makes mistakes in reordering. The most obvious, perhaps, is its failure to capture the wh-movement involving the interrogative word 多$_{11}$ (how); this should move to the beginning of the translated clause, consistent with English wh-fronting as opposed to Chinese *wh in situ*. The pairwise dominance model helps, since the dominance value between the interrogative word and its previous function word, the modal verb 能$_9$(can) in the baseline system's output, is neither, rather than rightFirst as in the better translation.

The fact that performance tends to be best using a frequency threshold of $N = 128$ strikes us as intuitively sensible, given what we know about word frequency rankings.[8] In English, for example, the most frequent 128 words include virtually all common conjunctions, determiners, prepositions, auxiliaries, and complementizers – the crucial elements of "syntactic glue" that characterize the types of linguistic phrases and the ordering relationships between them – and a very small proportion of content words. Using Adam Kilgarriff's lemmatized frequency list from the British National Corpus, http://www.kilgarriff.co.uk/bnc-readme.html, the most frequent 128 words in English are heavily dominated by determiners, "functional" adverbs like *not* and *when*, "particle" adverbs like *up*, prepositions, pronouns, and conjunctions, with some arguably "functional" auxiliary and light verbs like *be*, *have*, *do*, *give*, *make*, *take*. Content words are generally limited to a small number of frequent verbs like *think* and *want* and a very small handful of frequent nouns. In contrast, ranks 129-256 are heavily dominated by the traditional content-word categories, i.e. nouns, verbs, adjectives and adverbs, with a small number of left-over function words such as less frequent conjunctions *while*, *when*, and *although*.

Consistent with these observations for English, the empirical results for Chinese suggest that our

[8] In fact, we initially simply chose $N = 128$ for our experimentation, and then did runs with alternative N to confirm our intuitions.

approximation of function words using word frequency is reasonable. Using a list of approximately 900 linguistically identified function words in Chinese extracted from (Howard, 2002), we observe that that the performance drops when increasing N above 128 corresponds to a large increase in the number of non-function words used in the model. For example, with $N = 2048$, the proportion of non-function words is 88%, compared to 60% when $N = 128$.[9]

One natural extension of this work, therefore, would be to tighten up our characterization of function words, whether statistically, distributionally, or simply using manually created resources that exist for many languages. As a first step, we did a version of the Chinese-English experiment using the list of approximately 900 genuine function words, testing on the Chinese MT06 set. Perhaps surprisingly, translation performance, 30.90 BLEU, was around the level we obtained when using frequency to approximate function words at $N = 64$. However, we observe that many of the words in the linguistically motivated function word list are quite infrequent; this suggests that data sparseness may be an additional factor worth investigating.

Finally, although we believe there are strong motivations for focusing on the role of function words in reordering, there may well be value in extending the dominance model to include content categories. Verbs and many nouns have subcategorization properties that may influence phrase ordering, for example, and this may turn out to explain the increase in Arabic-English performance for $N = 2048$ using the MT08 test set. More generally, the approach we are taking can be viewed as a way of selectively lexicalizing the automatically extracted grammar, and there is a large range of potentially interesting choices in how such lexicalization could be done.

9 Related Work

In the introduction, we discussed Chiang's (2005) constituency feature, related ideas explored by Marton and Resnik (2008) and Chiang et al. (2008), and the target-side variation investigated by Zollman et al. (2006). These methods differ from each other mainly in terms of the specific linguistic knowledge being used and on which side the constraints are applied.

Shen et al. (2008) proposed to use linguistic knowledge expressed in terms of a dependency grammar, instead of a syntactic constituency grammar. Villar et al. (2008) attempted to use syntactic constituency on both the source and target languages in the same spirit as the constituency feature, along with some simple pattern-based heuristics – an approach also investigated by Iglesias et al. (2009). Aiming at improving the selection of derivations, Zhou et al. (2008) proposed prior derivation models utilizing syntactic annotation of the source language, which can be seen as smoothing the probabilities of hierarchical phrase features.

A key point is that the model we have introduced in this paper does not require the linguistic supervision needed in most of this prior work. We estimate the parameters of our model from parallel text without any linguistic annotation. That said, we would emphasize that our approach is, in fact, motivated in linguistic terms by the role of function words in natural language syntax.

10 Conclusion

We have presented a pairwise dominance model to address reordering issues that are not handled particularly well by standard hierarchical phrase-based modeling. In particular, the minimal linguistic commitment in hierarchical phrase-based models renders them susceptible to overgeneration of reordering choices. Our proposal handles the overgeneration problem by identifying hierarchical phrases with function words and by using function word relationships to incorporate soft constraints on topological orderings. Our experimental results demonstrate that introducing the pairwise dominance model into hierarchical phrase-based modeling improves performance significantly in large-scale Chinese-to-English and Arabic-to-English translation tasks.

Acknowledgments

This research was supported in part by the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-2-001. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the view of the sponsors.

[9] We plan to do corresponding experimentation and analysis for Arabic once we identify a suitable list of manually identified function words.

References

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 224–233, Honolulu, Hawaii, October.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan, June. Association for Computational Linguistics.

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

Jiaying Howard. 2002. *A Student Handbook for Chinese Function Words*. The Chinese University Press.

Gonzalo Iglesias, Adria de Gispert, Eduardo R. Banga, and William Byrne. 2009. Rule filtering by pattern for efficient hierarchical translation. In *Proceedings of the 12th Conference of the European Chapter of the Association of Computational Linguistics (to appear)*.

R. Kneser and H. Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing95*, pages 181–184, Detroit, MI, May.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133, Edmonton, Alberta, Canada, May. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of The 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1003–1011, Columbus, Ohio, June.

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

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

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

Hendra Setiawan, Min-Yen Kan, and Haizhou Li. 2007. Ordering phrases with function words. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 712–719, Prague, Czech Republic, June.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of The 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 577–585, Columbus, Ohio, June.

David Vilar, Daniel Stein, and Hermann Ney. 2008. Analysing soft syntax features and heuristics for hierarchical phrase based machine translation. *International Workshop on Spoken Language Translation 2008*, pages 190–197, October.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404, Sep.

Tiejun Zhao, Yajuan Lv, Jianmin Yao, Hao Yu, Muyun Yang, and Fang Liu. 2001. Increasing accuracy of chinese segmentation with strategy of multi-step processing. *Journal of Chinese Information Processing (Chinese Version)*, 1:13–18.

Bowen Zhou, Bing Xiang, Xiaodan Zhu, and Yuqing Gao. 2008. Prior derivation models for formally syntax-based translation using linguistically syntactic parsing and tree kernels. In *Proceedings of the ACL-08: HLT Second Workshop on Syntax and Structure in Statistical Translation (SSST-2)*, pages 19–27, Columbus, Ohio, June.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 138–141, New York City, June.

Phrase-Based Statistical Machine Translation as a Traveling Salesman Problem

Mikhail Zaslavskiy* Mines ParisTech, Institut Curie, 77305 Fontainebleau, France, mikhail.zaslavskiy@ensmp.fr

Marc Dymetman Xerox Research Centre Europe, 38240 Meylan, France, {marc.dymetman,nicola.cancedda}@xrce.xerox.com

Nicola Cancedda Xerox Research Centre Europe, 38240 Meylan, France, {marc.dymetman,nicola.cancedda}@xrce.xerox.com

Abstract

An efficient decoding algorithm is a crucial element of any statistical machine translation system. Some researchers have noted certain similarities between SMT decoding and the famous Traveling Salesman Problem; in particular (Knight, 1999) has shown that any TSP instance can be mapped to a sub-case of a word-based SMT model, demonstrating NP-hardness of the decoding task. In this paper, we focus on the reverse mapping, showing that any phrase-based SMT decoding problem can be directly reformulated as a TSP. The transformation is very natural, deepens our understanding of the decoding problem, and allows direct use of any of the powerful existing TSP solvers for SMT decoding. We test our approach on three datasets, and compare a TSP-based decoder to the popular beam-search algorithm. In all cases, our method provides competitive or better performance.

1 Introduction

Phrase-based systems (Koehn et al., 2003) are probably the most widespread class of Statistical Machine Translation systems, and arguably one of the most successful. They use aligned sequences of words, called biphrases, as building blocks for translations, and score alternative candidate translations for the same source sentence based on a log-linear model of the conditional probability of target sentences given the source sentence:

$$p(T, a|S) = \frac{1}{Z_S} \exp \sum_k \lambda_k h_k(S, a, T) \qquad (1)$$

where the h_k are features, that is, functions of the source string S, of the target string T, and of the alignment a, where the alignment is a representation of the sequence of biphrases that where used in order to build T from S; The λ_k's are weights and Z_S is a normalization factor that guarantees that p is a proper conditional probability distribution over the pairs (T, A). Some features are *local*, i.e. decompose over biphrases and can be precomputed and stored in advance. These typically include forward and reverse phrase conditional probability features $\log p(\tilde{t}|\tilde{s})$ as well as $\log p(\tilde{s}|\tilde{t})$, where $\tilde{s}$ is the source side of the biphrase and $\tilde{t}$ the target side, and the so-called "phrase penalty" and "word penalty" features, which count the number of phrases and words in the alignment. Other features are *non-local*, i.e. depend on the order in which biphrases appear in the alignment. Typical non-local features include one or more n-gram language models as well as a distortion feature, measuring by how much the order of biphrases in the candidate translation deviates from their order in the source sentence.

Given such a model, where the λ_i's have been tuned on a development set in order to minimize some error rate (see e.g. (Lopez, 2008)), together with a library of biphrases extracted from some large training corpus, a *decoder* implements the actual search among alternative translations:

$$(a^*, T^*) = \arg\max_{(a,T)} P(T, a|S). \qquad (2)$$

The decoding problem (2) is a discrete optimization problem. Usually, it is very hard to find the exact optimum and, therefore, an approximate solution is used. Currently, most decoders are based on some variant of a heuristic left-to-right search, that is, they attempt to build a candidate translation (a, T) incrementally, from left to right, extending the current partial translation at each step with a new biphrase, and computing a score composed of two contributions: one for the known elements of the partial translation so far, and one a heuristic

* This work was conducted during an internship at XRCE.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 333–341, Suntec, Singapore, 2-7 August 2009.

estimate of the remaining cost for completing the translation. The variant which is mostly used is a form of *beam-search*, where several partial candidates are maintained in parallel, and candidates for which the current score is too low are pruned in favor of candidates that are more promising.

We will see in the next section that some characteristics of beam-search make it a suboptimal choice for phrase-based decoding, and we will propose an alternative. This alternative is based on the observation that phrase-based decoding can be very naturally cast as a Traveling Salesman Problem (TSP), one of the best studied problems in combinatorial optimization. We will show that this formulation is not only a powerful conceptual device for reasoning on decoding, but is also practically convenient: in the same amount of time, off-the-shelf TSP solvers can find higher scoring solutions than the state-of-the art beam-search decoder implemented in *Moses* (Hoang and Koehn, 2008).

2 Related work

Beam-search decoding

In beam-search decoding, candidate translation prefixes are iteratively extended with new phrases. In its most widespread variant, *stack decoding*, prefixes obtained by consuming the same number of source words, no matter which, are grouped together in the same *stack*[1] and compete against one another. *Threshold* and *histogram* pruning are applied: the former consists in dropping all prefixes having a score lesser than the best score by more than some fixed amount (a parameter of the algorithm), the latter consists in dropping all prefixes below a certain rank.

While quite successful in practice, stack decoding presents some shortcomings. A first one is that prefixes obtained by translating different subsets of source words compete against one another. In one early formulation of stack decoding for SMT (Germann et al., 2001), the authors indeed proposed to lazily create one stack for each subset of source words, but acknowledged issues with the potential combinatorial explosion in the number of stacks. This problem is reduced by the use of heuristics for estimating the cost of translating the remaining part of the source sentence. However, this solution is only partially satisfactory. On the one hand, heuristics should be computationally light, much lighter than computing the actual best score itself, while, on the other hand, the heuristics should be tight, as otherwise pruning errors will ensue. There is no clear criterion to guide in this trade-off. Even when good heuristics are available, the decoder will show a bias towards putting at the beginning the translation of a certain portion of the source, either because this portion is less ambiguous (i.e. its translation has larger conditional probability) or because the associated heuristics is less tight, hence more optimistic. Finally, since the translation is built left-to-right the decoder cannot optimize the search by taking advantage of highly unambiguous and informative portions that should be best translated far from the beginning. All these reasons motivate considering alternative decoding strategies.

Word-based SMT and the TSP

As already mentioned, the similarity between SMT decoding and TSP was recognized in (Knight, 1999), who focussed on showing that any TSP can be reformulated as a sub-class of the SMT decoding problem, proving that SMT decoding is NP-hard. Following this work, the existence of many efficient TSP algorithms then inspired certain adaptations of the underlying techniques to SMT decoding for word-based models. Thus, (Germann et al., 2001) adapt a TSP subtour elimination strategy to an IBM-4 model, using generic Integer Programming techniques. The paper comes close to a TSP formulation of decoding with IBM-4 models, but does not pursue this route to the end, stating that *"It is difficult to convert decoding into straight TSP, but a wide range of combinatorial optimization problems (including TSP) can be expressed in the more general framework of linear integer programming"*. By employing generic IP techniques, it is however impossible to rely on the variety of more efficient both exact and approximate approaches which have been designed specifically for the TSP. In (Tillmann and Ney, 2003) and (Tillmann, 2006), the authors modify a certain Dynamic Programming technique used for TSP for use with an IBM-4 word-based model and a phrase-based model respectively. However, to our knowledge, none of these works has proposed a direct reformulation of these SMT models as TSP instances. We believe we are the first to do so, working in our case

[1] While commonly adopted in the speech and SMT communities, this is a bit of a misnomer, since the used data structures are priority queues, not stacks.

with the mainstream phrase-based SMT models, and therefore making it possible to directly apply existing TSP solvers to SMT.

3 The Traveling Salesman Problem and its variants

In this paper the Traveling Salesman Problem appears in four variants:

STSP. The most standard, and most studied, variant is the *Symmetric TSP*: we are given a non-directed graph G on N nodes, where the edges carry real-valued costs. The STSP problem consists in finding a tour of minimal total cost, where a tour (also called Hamiltonian Circuit) is a "circular" sequence of nodes visiting each node of the graph exactly once;

ATSP. The *Asymmetric TSP*, or ATSP, is a variant where the underlying graph G is directed and where, for i and j two nodes of the graph, the edges (i,j) and (j,i) may carry different costs.

SGTSP. The *Symmetric Generalized TSP*, or SGTSP: given a non-oriented graph G of $|G|$ nodes with edges carrying real-valued costs, given a partition of these $|G|$ nodes into m non-empty, disjoint, subsets (called clusters), find a circular sequence of m nodes of minimal total cost, where each cluster is visited exactly once.

AGTSP. The *Asymmetric Generalized TSP*, or AGTSP: similar to the SGTSP, but G is now a directed graph.

The STSP is often simply denoted TSP in the literature, and is known to be NP-hard (Applegate et al., 2007); however there has been enormous interest in developing efficient solvers for it, both exact and approximate.

Most of existing algorithms are designed for *STSP*, but *ATSP*, *SGTSP* and *AGTSP* may be reduced to *STSP*, and therefore solved by *STSP* algorithms.

3.1 Reductions AGTSP→ATSP→STSP

The transformation of the AGTSP into the ATSP, introduced by (Noon and Bean, 1993)), is illustrated in Figure (1). In this diagram, we assume that $Y_1, \ldots, Y_K$ are the nodes of a given cluster, while X and Z are arbitrary nodes belonging to other clusters. In the transformed graph, we introduce edges between the Y_i's in order to form a cycle as shown in the figure, where each edge has a large negative cost $-K$. We leave alone the incoming edge to Y_i from X, but the outgoing edge

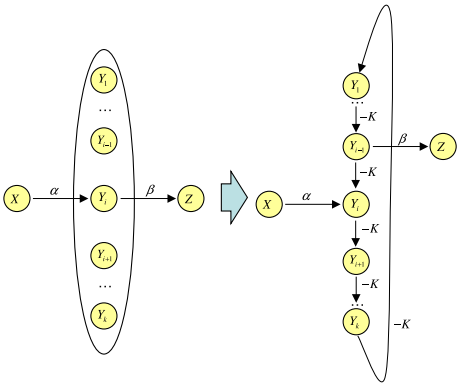


Figure 1: AGTSP→ATSP.

from Y_i to X has its origin changed to Y_{i-1}. A feasible tour in the original AGTSP problem passing through X, Y_i, Z will then be "encoded" as a tour of the transformed graph that first traverses X , then traverses $Y_i, \ldots, Y_K, \ldots, Y_{i-1}$, then traverses Z (this encoding will have the same cost as the original cost, minus $(k-1)K$). Crucially, if K is large enough, then the solver for the transformed ATSP graph will tend to traverse as many K edges as possible, meaning that it will traverse exactly $k-1$ such edges in the cluster, that is, it will produce an encoding of some feasible tour of the AGTSP problem.

As for the transformation ATSP→STSP, several variants are described in the literature, e.g. (Applegate et al., 2007, p. 126); the one we use is from (Wikipedia, 2009) (not illustrated here for lack of space).

3.2 TSP algorithms

TSP is one of the most studied problems in combinatorial optimization, and even a brief review of existing approaches would take too much place. Interested readers may consult (Applegate et al., 2007; Gutin, 2003) for good introductions.

One of the best existing TSP solvers is implemented in the open source *Concorde* package (Applegate et al., 2005). *Concorde* includes the fastest exact algorithm and one of the most efficient implementations of the Lin-Kernighan (LK) heuristic for finding an approximate solution. LK works by generating an initial random feasible solution for the TSP problem, and then repeatedly identifying an ordered subset of k edges in the current tour and an ordered subset of k edges not included in the tour such that when they are swapped the objective function is improved. This is somewhat

reminiscent of the *Greedy decoding* of (Germann et al., 2001), but in LK several transformations can be applied simultaneously, so that the risk of being stuck in a local optimum is reduced (Applegate et al., 2007, chapter 15).

As will be shown in the next section, phrase-based SMT decoding can be directly reformulated as an AGTSP. Here we use *Concorde* through first transforming AGTSP into STSP, but it might also be interesting in the future to use algorithms specifically designed for AGTSP, which could improve efficiency further (see Conclusion).

4 Phrase-based Decoding as TSP

In this section we reformulate the SMT decoding problem as an *AGTSP*. We will illustrate the approach through a simple example: translating the French sentence *"cette traduction automatique est curieuse"* into English. We assume that the relevant biphrases for translating the sentence are as follows:

ID	source	target
h	*cette*	*this*
t	*traduction*	*translation*
ht	*cette traduction*	*this translation*
mt	*traduction automatique*	*machine translation*
a	*automatique*	*automatic*
m	*automatique*	*machine*
i	*est*	*is*
s	*curieuse*	*strange*
c	*curieuse*	*curious*

Under this model, we can produce, among others, the following translations:

$h \cdot mt \cdot i \cdot s$	*this machine translation is strange*
$h \cdot c \cdot t \cdot i \cdot a$	*this curious translation is automatic*
$ht \cdot s \cdot i \cdot a$	*this translation strange is automatic*

where we have indicated on the left the ordered sequence of biphrases that leads to each translation.

We now formulate decoding as an AGTSP, in the following way. The graph nodes are all the possible pairs (w, b), where w is a source word in the source sentence s and b is a biphrase containing this source word. The graph clusters are the subsets of the graph nodes that share a common source word w.

The costs of a transition between nodes M and N of the graph are defined as follows:
(a) If M is of the form (w, b) and N of the form (w', b), in which b is a single biphrase, and w and w' are consecutive words in b, then the transition cost is 0: once we commit to using the first word of b, there is no additional cost for traversing the other source words covered by b.
(b) If $M = (w, b)$, where w is the *rightmost source word* in the biphrase b, and $N = (w', b')$, where $w' \neq w$ is the *leftmost source word* in b', then the transition cost corresponds to the cost of selecting b' just after b; this will correspond to "consuming" the source side of b' after having consumed the source side of b (whatever their relative positions in the source sentence), and to producing the target side of b' directly after the target side of b; the transition cost is then the addition of several contributions (weighted by their respective λ (not shown), as in equation 1):

- The cost associated with the features local to b in the biphrase library;
- The "distortion" cost of consuming the source word w' just after the source word w: $|\text{pos}(w') - \text{pos}(w) - 1|$, where $\text{pos}(w)$ and $\text{pos}(w')$ are the positions of w and w' in the source sentence.
- The language model cost of producing the target words of b' right after the target words of b; with a bigram language model, this cost can be precomputed directly from b and b'. This restriction to bigram models will be removed in Section 4.1.

(c) In all other cases, the transition cost is infinite, or, in other words, there is no edge in the graph between M and N.

A special cluster containing a single node (denoted by $-$$ in the figures), and corresponding to special *beginning-of-sentence* symbols must also be included: the corresponding edges and weights can be worked out easily. Figures 2 and 3 give some illustrations of what we have just described.

4.1 From Bigram to N-gram LM

Successful phrase-based systems typically employ language models of order higher than two. However, our models so far have the following important "Markovian" property: the cost of a path is additive relative to the costs of transitions. For example, in the example of Figure 3, the cost of *this · machine translation · is · strange*, can only take into account the conditional probability of the word *strange* relative to the word *is*, but not relative to the words *translation* and *is*. If we want to extend the power of the model to general n-gram language models, and in particular to the 3-gram

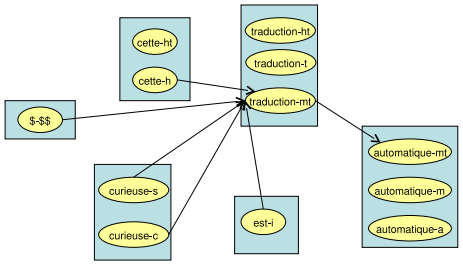


Figure 2: Transition graph for the source sentence *cette traduction automatique est curieuse*. Only edges entering or exiting the node traduction – mt are shown. The only successor to [traduction – mt] is [automatique – mt], and [cette – ht] is not a predecessor of [traduction – mt].

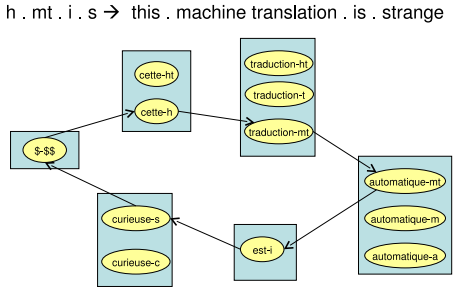


Figure 3: A GTSP tours is illustrated, corresponding to the displayed output.

case (on which we concentrate here, but the techniques can be easily extended to the general case), the following approach can be applied.

Compiling Out for Trigram models

This approach consists in "compiling out" all biphrases with a target side of only one word. We replace each biphrase b with single-word target side by "extended" biphrases $b_1, \ldots, b_r$, which are "concatenations" of b and some other biphrase b' in the library.[2] To give an example, consider that we: (1) remove from the biphrase library the biphrase i, which has a single word target, and (2) add to the library the extended biphrases mti, ti, $si, \ldots$, that is, all the extended biphrases consisting of the concatenation of a biphrase in the library with i, then it is clear that these extended biphrases will provide enough context to compute a trigram probability for the target word produced immediately next (in the examples, for the words *strange*, *automatic* and *automatic* respectively). If we do that exhaustively for all biphrases (relevant for the source sentence at hand) that, like i, have a single-word target, we will obtain a representation that allows a trigram language model to be computed at each point.

[2] In the figures, such "concatenations" are denoted by $[b' \cdot b]$; they are interpreted as encapsulations of first consuming the source side of b', *whether or not this source side precedes the source side of* b *in the source sentence*, producing the target side of b', consuming the source side of b, and producing the target side of b immediately after that of b'.

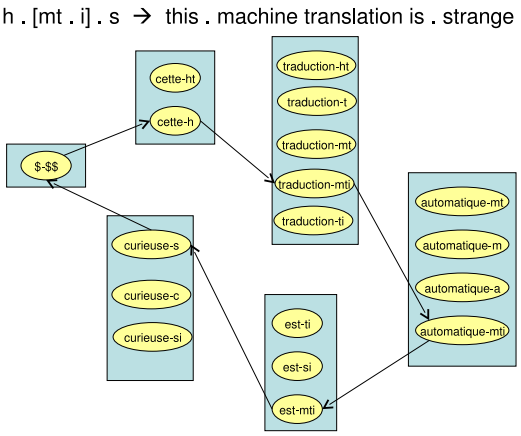


Figure 4: Compiling-out of biphrase i: (est,is).

The situation becomes clearer by looking at Figure 4, where we have only eliminated the biphrase i, and only shown some of the extended biphrases that now encapsulate i, and where we show one valid circuit. Note that we are now able to associate with the edge connecting the two nodes (est, mti) and $(\text{curieuse}, s)$ a trigram cost because mti provides a large enough target context.

While this exhaustive "compiling out" method works in principle, it has a serious defect: if for the sentence to be translated, there are m relevant biphrases, among which k have single-word targets, then we will create on the order of km extended biphrases, which may represent a significant overhead for the TSP solver, as soon as k is large relative to m, which is typically the case. The problem becomes even worse if we extend the compiling-out method to n-gram language models with $n > 3$. In the Future Work section below, we describe a powerful approach for circumventing this problem, but with which we have not experimented yet.

5 Experiments

5.1 Monolingual word re-ordering

In the first series of experiments we consider the artificial task of reconstructing the original word order of a given English sentence. First, we randomly permute words in the sentence, and then we try to reconstruct the original order by max-

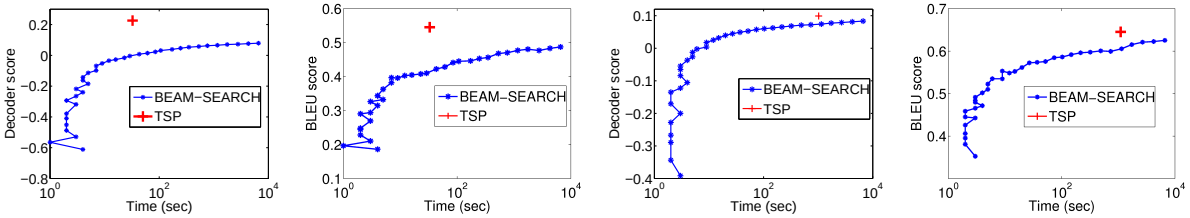


Figure 5: (a), (b): LM and BLEU scores as functions of time for a bigram LM; (c), (d): the same for a trigram LM. The x axis corresponds to the cumulative time for processing the test set; for (a) and (c), the y axis corresponds to the mean difference (over all sentences) between the lm score of the output and the lm score of the reference normalized by the sentence length N: (LM(ref)-LM(true))/N. The solid line with star marks corresponds to using beam-search with different pruning thresholds, which result in different processing times and performances. The cross corresponds to using the exact-TSP decoder (in this case the time to the optimal solution is not under the user's control).

imizing the LM score over all possible permutations. The reconstruction procedure may be seen as a translation problem from "Bad English" to "Good English". Usually the LM score is used as one component of a more complex decoder score which also includes biphrase and distortion scores. But in this particular "translation task" from bad to good English, we consider that all "biphrases" are of the form $e - e$, where e is an English word, and we do not take into account any distortion: we only consider the quality of the permutation as it is measured by the LM component. Since for each "source word" e, there is exactly one possible "biphrase" $e - e$ each cluster of the Generalized TSP representation of the decoding problem contains exactly one node; in other terms, the Generalized TSP in this situation is simply a standard TSP. Since the decoding phase is then equivalent to a word reordering, the LM score may be used to compare the performance of different decoding algorithms. Here, we compare three different algorithms: classical beam-search (*Moses*); a decoder based on an exact TSP solver (*Concorde*); a decoder based on an approximate TSP solver (Lin-Kernighan as implemented in the Concorde solver) [3]. In the Beam-search and the LK-based TSP solver we can control the trade-off between approximation quality and running time. To measure re-ordering quality, we use two scores. The first one is just the "internal" LM score; since all three algorithms attempt to maximize this score, a natural evaluation procedure is to plot its value versus the elapsed time. The second score is BLEU (Papineni et al., 2001), computed between the reconstructed and the original sentences, which allows us to check how well the quality of reconstruction correlates with the internal score. The training dataset for learning the LM consists of 50000 sentences from NewsCommentary corpus (Callison-Burch et al., 2008), the test dataset for word reordering consists of 170 sentences, the average length of test sentences is equal to 17 words.

Bigram based reordering. First we consider a bigram Language Model and the algorithms try to find the re-ordering that maximizes the LM score. The TSP solver used here is exact, that is, it actually finds the optimal tour. Figures 5(a,b) present the performance of the TSP and Beam-search based methods.

Trigram based reordering. Then we consider a trigram based Language Model and the algorithms again try to maximize the LM score. The trigram model used is a variant of the exhaustive compiling-out procedure described in Section 4.1. Again, we use an exact TSP solver.

Looking at Figure 5a, we see a somewhat surprising fact: the cross and some star points have positive y coordinates! This means that, when using a bigram language model, it is often possible to reorder the words of a randomly permuted reference sentence in such a way that the LM score of the reordered sentence is larger than the LM of the reference. A second notable point is that the increase in the LM-score of the beam-search with time is steady but very slow, and never reaches the level of performance obtained with the exact-TSP procedure, even when increasing the time by sev-

[3] Both TSP decoders may be used with/or without a *distortion limit*; in our experiments we do not use this parameter.

eral orders of magnitude. Also to be noted is that the solution obtained by the exact-TSP is provably the optimum, which is almost never the case of the beam-search procedure. In Figure 5b, we report the BLEU score of the reordered sentences in the test set relative to the original reference sentences. Here we see that the exact-TSP outputs are closer to the references in terms of BLEU than the beam-search solutions. Although the TSP output does not recover the reference sentences (it produces sentences with a slightly higher LM score than the references), it does reconstruct the references better than the beam-search. The experiments with trigram language models (Figures 5(c,d)) show similar trends to those with bigrams.

5.2 Translation experiments with a bigram language model

In this section we consider two real translation tasks, namely, translation from English to French, trained on Europarl (Koehn et al., 2003) and translation from German to Spanish training on the NewsCommentary corpus. For Europarl, the training set includes 2.81 million sentences, and the test set 500. For NewsCommentary the training set is smaller: around 63k sentences, with a test set of 500 sentences. Figure 6 presents Decoder and Bleu scores as functions of time for the two corpuses.

Since in the real translation task, the size of the TSP graph is much larger than in the artificial reordering task (in our experiments the median size of the TSP graph was around 400 nodes, sometimes growing up to 2000 nodes), directly applying the exact TSP solver would take too long; instead we use the approximate LK algorithm and compare it to Beam-Search. The efficiency of the LK algorithm can be significantly increased by using a good initialization. To compare the quality of the LK and Beam-Search methods we take a rough initial solution produced by the Beam-Search algorithm using a small value for the stack size and then use it as initial point, both for the LK algorithm and for further Beam-Search optimization (where as before we vary the Beam-Search thresholds in order to trade quality for time).

In the case of the Europarl corpus, we observe that LK outperforms Beam-Search in terms of the Decoder score as well as in terms of the BLEU score. Note that the difference between the two algorithms increases steeply at the beginning, which means that we can significantly increase the quality of the Beam-Search solution by using the LK algorithm at a very small price. In addition, it is important to note that the BLEU scores obtained in these experiments correspond to feature weights, in the log-linear model (1), that have been optimized for the Moses decoder, but not for the TSP decoder: optimizing these parameters relatively to the TSP decoder could improve its BLEU scores still further.

On the News corpus, again, LK outperforms Beam-Search in terms of the Decoder score. The situation with the BLEU score is more confuse. Both algorithms do not show any clear score improvement with increasing running time which suggests that the decoder's objective function is not very well correlated with the BLEU score on this corpus.

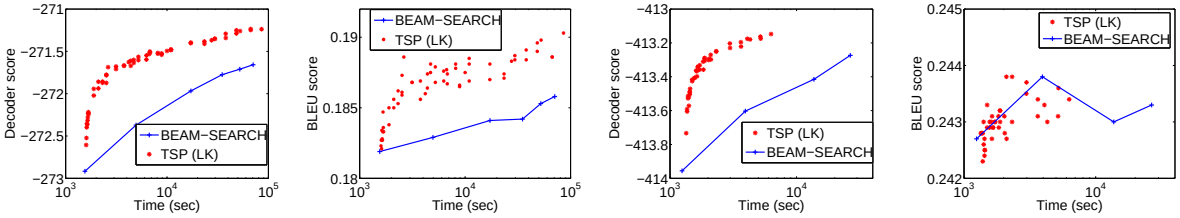
6 Future Work

In section 4.1, we described a general "compiling out" method for extending our TSP representation to handling trigram and N-gram language models, but we noted that the method may lead to combinatorial explosion of the TSP graph. While this problem was manageable for the artificial monolingual word re-ordering (which had only one possible translation for each source word), it becomes unwieldy for the real translation experiments, which is why in this paper we only considered bigram LMs for these experiments. However, we know how to handle this problem in principle, and we now describe a method that we plan to experiment with in the future.

To avoid the large number of artificial biphrases as in 4.1, we perform an *adaptive selection*. Let us suppose that (w, b) is a SMT decoding graph node, where b is a biphrase containing only one word on the target side. On the first step, when we evaluate the traveling cost from (w, b) to (w', b'), we take the language model component equal to

$$\min_{b'' \neq b', b} -\log p(b'.v | b.e, b''.e),$$

where $b'.v$ represents the first word of the b' target side, $b.e$ is the only word of the b target side, and $b''.e$ is the last word of the b'' target size. This procedure underestimates the total cost of tour passing through biphrases that have a single-word target. Therefore if the optimal tour passes only through biphrases with more than one



(a) (b) (c) (d)

Figure 6: (a), (b): Europarl corpus, translation from English to French; (c),(d): NewsCommentary corpus, translation from German to Spanish. Average value of the decoder and the BLEU scores (over 500 test sentences) as a function of time. The trade-off quality/time in the case of LK is controlled by the number of iterations, and each point corresponds to a particular number of iterations, in our experiments LK was run with a number of iterations varying between 2k and 170k. The same trade-off in the case of Beam-Search is controlled by varying the beam thresholds.

word on their target side, then we are sure that this tour is also optimal in terms of the tri-gram language model. Otherwise, if the optimal tour passes through (w, b), where b is a biphrase having a single-word target, we add only the extended biphrases related to b as we described in section 4.1, and then we recompute the optimal tour. Iterating this procedure provably converges to an optimal solution.

This powerful method, which was proposed in (Kam and Kopec, 1996; Popat et al., 2001) in the context of a finite-state model (but not of TSP), can be easily extended to N-gram situations, and typically converges in a small number of iterations.

7 Conclusion

The main contribution of this paper has been to propose a transformation for an arbitrary phrase-based SMT decoding instance into a TSP instance. While certain similarities of SMT decoding and TSP were already pointed out in (Knight, 1999), where it was shown that any Traveling Salesman Problem may be reformulated as an instance of a (simplistic) SMT decoding task, and while certain techniques used for TSP were then adapted to word-based SMT decoding (Germann et al., 2001; Tillmann and Ney, 2003; Tillmann, 2006), we are not aware of any previous work that shows that SMT decoding can be directly reformulated as a TSP. Beside the general interest of this transformation for understanding decoding, it also opens the door to direct application of the variety of existing TSP algorithms to SMT. Our experiments on synthetic and real data show that fast TSP algorithms can handle selection and reordering in SMT comparably or better than the state-of-the-art beam-search strategy, converging on solutions with higher objective function in a shorter time.

The proposed method proceeds by first constructing an AGTSP instance from the decoding problem, and then converting this instance first into ATSP and finally into STSP. At this point, a direct application of the well known STSP solver *Concorde* (with Lin-Kernighan heuristic) already gives good results. We believe however that there might exist even more efficient alternatives. Instead of converting the AGTSP instance into a STSP instance, it might prove better to use directly algorithms expressly designed for ATSP or AGTSP. For instance, some of the algorithms tested in the context of the *DIMACS* implementation challenge for ATSP (Johnson et al., 2002) might well prove superior. There is also active research around AGTSP algorithms. Recently new effective methods based on a "memetic" strategy (Buriol et al., 2004; Gutin et al., 2008) have been put forward. These methods combined with our proposed formulation provide ready-to-use SMT decoders, which it will be interesting to compare.

Acknowledgments

Thanks to Vassilina Nikoulina for her advice about running Moses on the test datasets.

References

David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. 2005. Concorde tsp solver. `http://www.tsp.gatech.edu/concorde.html`.

David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. 2007. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, January.

Luciana Buriol, Paulo M. França, and Pablo Moscato. 2004. A new memetic algorithm for the asymmetric traveling salesman problem. *Journal of Heuristics*, 10(5):483–506.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Josh Schroeder, and Cameron Shaw Fordyce, editors. 2008. *Proceedings of the Third Workshop on SMT*. ACL, Columbus, Ohio, June.

Ulrich Germann, Michael Jahr, Kevin Knight, and Daniel Marcu. 2001. Fast decoding and optimal decoding for machine translation. In *In Proceedings of ACL 39*, pages 228–235.

Gregory Gutin, Daniel Karapetyan, and Krasnogor Natalio. 2008. Memetic algorithm for the generalized asymmetric traveling salesman problem. In *NICSO 2007*, pages 199–210. Springer Berlin.

G. Gutin. 2003. Travelling salesman and related problems. In *Handbook of Graph Theory*.

Hieu Hoang and Philipp Koehn. 2008. Design of the Moses decoder for statistical machine translation. In *ACL 2008 Software workshop*, pages 58–65, Columbus, Ohio, June. ACL.

D.S. Johnson, G. Gutin, L.A. McGeoch, A. Yeo, W. Zhang, and A. Zverovich. 2002. Experimental analysis of heuristics for the atsp. In *The Travelling Salesman Problem and Its Variations*, pages 445–487.

Anthony C. Kam and Gary E. Kopec. 1996. Document image decoding by heuristic search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:945–950.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25:607–615.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL 2003*, pages 48–54, Morristown, NJ, USA. Association for Computational Linguistics.

Adam Lopez. 2008. Statistical machine translation. *ACM Comput. Surv.*, 40(3):1–49.

C. Noon and J.C. Bean. 1993. An efficient transformation of the generalized traveling salesman problem. *INFOR*, pages 39–44.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei J. Zhu. 2001. BLEU: a Method for Automatic Evaluation of Machine Translation. *IBM Research Report*, RC22176.

Kris Popat, Daniel H. Greene, Justin K. Romberg, and Dan S. Bloomberg. 2001. Adding linguistic constraints to document image decoding: Comparing the iterated complete path and stack algorithms.

Christoph Tillmann and Hermann Ney. 2003. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Comput. Linguist.*, 29(1):97–133.

Christoph Tillmann. 2006. Efficient Dynamic Programming Search Algorithms For Phrase-Based SMT. In *Workshop On Computationally Hard Problems And Joint Inference In Speech And Language Processing*.

Wikipedia. 2009. Travelling Salesman Problem — Wikipedia, The Free Encyclopedia. [Online; accessed 5-May-2009].

Concise Integer Linear Programming Formulations for Dependency Parsing

André F. T. Martins[*†] **Noah A. Smith**[*] **Eric P. Xing**[*]
[*]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[†]Instituto de Telecomunicações, Instituto Superior Técnico, Lisboa, Portugal
{afm,nasmith,epxing}@cs.cmu.edu

Abstract

We formulate the problem of non-projective dependency parsing as a polynomial-sized integer linear program. Our formulation is able to handle non-local output features in an efficient manner; not only is it compatible with prior knowledge encoded as hard constraints, it can also learn soft constraints from data. In particular, our model is able to learn correlations among neighboring arcs (siblings and grandparents), word valency, and tendencies toward nearly-projective parses. The model parameters are learned in a max-margin framework by employing a linear programming relaxation. We evaluate the performance of our parser on data in several natural languages, achieving improvements over existing state-of-the-art methods.

1 Introduction

Much attention has recently been devoted to integer linear programming (ILP) formulations of NLP problems, with interesting results in applications like semantic role labeling (Roth and Yih, 2005; Punyakanok et al., 2004), dependency parsing (Riedel and Clarke, 2006), word alignment for machine translation (Lacoste-Julien et al., 2006), summarization (Clarke and Lapata, 2008), and coreference resolution (Denis and Baldridge, 2007), among others. In general, the rationale for the development of ILP formulations is to incorporate non-local features or global constraints, which are often difficult to handle with traditional algorithms. ILP formulations focus more on the modeling of problems, rather than algorithm design. While solving an ILP is NP-hard in general, fast solvers are available today that make it a practical solution for many NLP problems.

This paper presents new, concise ILP formulations for projective and non-projective dependency parsing. We believe that our formulations can pave the way for efficient exploitation of global features and constraints in parsing applications, leading to more powerful models. Riedel and Clarke (2006) cast dependency parsing as an ILP, but *efficient* formulations remain an open problem. Our formulations offer the following comparative advantages:

- The numbers of variables and constraints are polynomial in the sentence length, as opposed to requiring exponentially many constraints, eliminating the need for incremental procedures like the cutting-plane algorithm;
- LP relaxations permit fast online discriminative training of the constrained model;
- Soft constraints may be automatically learned from data. In particular, our formulations handle higher-order arc interactions (like siblings and grandparents), model word valency, and can learn to favor nearly-projective parses.

We evaluate the performance of the new parsers on standard parsing tasks in seven languages. The techniques that we present are also compatible with scenarios where expert knowledge is available, for example in the form of hard or soft first-order logic constraints (Richardson and Domingos, 2006; Chang et al., 2008).

2 Dependency Parsing

2.1 Preliminaries

A dependency tree is a lightweight syntactic representation that attempts to capture functional relationships between words. Lately, this formalism has been used as an alternative to phrase-based parsing for a variety of tasks, ranging from machine translation (Ding and Palmer, 2005) to relation extraction (Culotta and Sorensen, 2004) and question answering (Wang et al., 2007).

Let us first describe formally the set of legal dependency parse trees. Consider a sentence $x =$

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 342–350,
Suntec, Singapore, 2-7 August 2009.

$\langle w_0, \ldots, w_n \rangle$, where w_i denotes the word at the i-th position, and $w_0 = \$$ is a wall symbol. We form the (complete[1]) directed graph $D = \langle V, A \rangle$, with vertices in $V = \{0, \ldots, n\}$ (the i-th vertex corresponding to the i-th word) and arcs in $A = V^2$. Using terminology from graph theory, we say that $B \subseteq A$ is an r-**arborescence**[2] of the directed graph D if $\langle V, B \rangle$ is a (directed) tree rooted at r. We define the set of legal dependency parse trees of x (denoted $\mathcal{Y}(x)$) as the set of 0-arborescences of D, i.e., we admit each arborescence as a potential dependency tree.

Let $y \in \mathcal{Y}(x)$ be a legal dependency tree for x; if the arc $a = \langle i, j \rangle \in y$, we refer to i as the parent of j (denoted $i = \pi(j)$) and j as a child of i. We also say that a is **projective** (in the sense of Kahane et al., 1998) if any vertex k in the span of a is reachable from i (in other words, if for any k satisfying $\min(i, j) < k < \max(i, j)$, there is a directed path in y from i to k). A dependency tree is called projective if it only contains projective arcs. Fig. 1 illustrates this concept.[3]

The formulation to be introduced in §3 makes use of the notion of the *incidence vector* associated with a dependency tree $y \in \mathcal{Y}(x)$. This is the binary vector $\mathbf{z} \triangleq \langle z_a \rangle_{a \in A}$ with each component defined as $z_a = \mathbb{I}(a \in y)$ (here, $\mathbb{I}(.)$ denotes the indicator function). Considering simultaneously all incidence vectors of legal dependency trees and taking the convex hull, we obtain a polyhedron that we call the **arborescence polytope**, denoted by $\mathcal{Z}(x)$. Each vertex of $\mathcal{Z}(x)$ can be identified with a dependency tree in $\mathcal{Y}(x)$. The Minkowski-Weyl theorem (Rockafellar, 1970) ensures that $\mathcal{Z}(x)$ has a representation of the form $\mathcal{Z}(x) = \{\mathbf{z} \in \mathbb{R}^{|A|} \mid \mathbf{A}\mathbf{z} \leq \mathbf{b}\}$, for some p-by-$|A|$ matrix $\mathbf{A}$ and some vector $\mathbf{b}$ in $\mathbb{R}^p$. However, it is not easy to obtain a *compact* representation (where p grows polynomially with the number of words n). In §3, we will provide a compact representation of an outer polytope $\bar{\mathcal{Z}}(x) \supseteq \mathcal{Z}(x)$ whose *integer vertices* correspond to dependency trees. Hence, the problem of finding the dependency tree that maximizes some linear function of the incidence vectors can be cast as an ILP. A similar idea was applied to word alignment by Lacoste-Julien et al. (2006), where permutations (rather than arborescences) were the combinatorial structure being requiring representation.

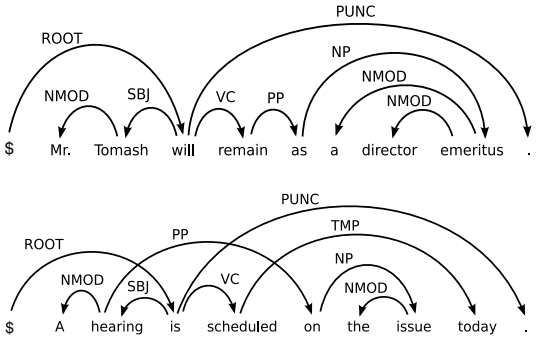


Figure 1: A projective dependency parse (top), and a non-projective dependency parse (bottom) for two English sentences; examples from McDonald and Satta (2007).

Letting $\mathcal{X}$ denote the set of possible sentences, define $\mathcal{Y} \triangleq \bigcup_{x \in \mathcal{X}} \mathcal{Y}(x)$. Given a labeled dataset $\mathcal{L} \triangleq \langle \langle x_1, y_1 \rangle, \ldots, \langle x_m, y_m \rangle \rangle \in (\mathcal{X} \times \mathcal{Y})^m$, we aim to learn a parser, i.e., a function $h : \mathcal{X} \to \mathcal{Y}$ that given $x \in \mathcal{X}$ outputs a legal dependency parse $y \in \mathcal{Y}(x)$. The fact that there are exponentially many candidates in $\mathcal{Y}(x)$ makes dependency parsing a structured classification problem.

2.2 Arc Factorization and Locality

There has been much recent work on dependency parsing using graph-based, transition-based, and hybrid methods; see Nivre and McDonald (2008) for an overview. Typical graph-based methods consider linear classifiers of the form

$$h_{\mathbf{w}}(x) = \text{argmax}_{y \in \mathcal{Y}} \, \mathbf{w}^\top \mathbf{f}(x, y), \qquad (1)$$

where $\mathbf{f}(x, y)$ is a vector of features and $\mathbf{w}$ is the corresponding weight vector. One wants $h_{\mathbf{w}}$ to have small expected loss; the typical loss function is the Hamming loss, $\ell(y'; y) \triangleq |\{\langle i, j \rangle \in y' : \langle i, j \rangle \notin y\}|$. Tractability is usually ensured by strong factorization assumptions, like the one underlying the **arc-factored model** (Eisner, 1996; McDonald et al., 2005), which forbids any feature that depends on two or more arcs. This induces a decomposition of the feature vector $\mathbf{f}(x, y)$ as:

$$\mathbf{f}(x, y) = \textstyle\sum_{a \in y} \mathbf{f}_a(x). \qquad (2)$$

Under this decomposition, each arc receives a score; parsing amounts to choosing the configuration that maximizes the overall score, which, as

[1] The general case where $A \subseteq V^2$ is also of interest; it arises whenever a constraint or a lexicon forbids some arcs from appearing in dependency tree. It may also arise as a consequence of a first-stage pruning step where some candidate arcs are eliminated; this will be further discussed in §4.

[2] Or "directed spanning tree with designated root r."

[3] In this paper, we consider *unlabeled* dependency parsing, where only the backbone structure (i.e., the arcs without the labels depicted in Fig. 1) is to be predicted.

shown by McDonald et al. (2005), is an instance of the **maximal arborescence problem**. Combinatorial algorithms (Chu and Liu, 1965; Edmonds, 1967) can solve this problem in cubic time.[4] If the dependency parse trees are restricted to be projective, cubic-time algorithms are available via dynamic programming (Eisner, 1996). While in the projective case, the arc-factored assumption can be weakened in certain ways while maintaining polynomial parser runtime (Eisner and Satta, 1999), the same does not happen in the nonprojective case, where finding the highest-scoring tree becomes NP-hard (McDonald and Satta, 2007). Approximate algorithms have been employed to handle models that are not arc-factored (although features are still fairly local): McDonald and Pereira (2006) adopted an approximation based on $O(n^3)$ projective parsing followed by a hill-climbing algorithm to rearrange arcs, and Smith and Eisner (2008) proposed an algorithm based on loopy belief propagation.

3 Dependency Parsing as an ILP

Our approach will build a graph-based parser without the drawback of a restriction to local features. By formulating inference as an ILP, nonlocal features can be easily accommodated in our model; furthermore, by using a relaxation technique we can still make learning tractable. The impact of LP-relaxed inference in the learning problem was studied elsewhere (Martins et al., 2009).

A **linear program** (LP) is an optimization problem of the form

$$\begin{array}{ll} \min_{\mathbf{x}\in\mathbb{R}^d} & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} & \mathbf{A}\mathbf{x} \leq \mathbf{b}. \end{array} \quad (3)$$

If the problem is feasible, the optimum is attained at a vertex of the polyhedron that defines the constraint space. If we add the constraint $\mathbf{x} \in \mathbb{Z}^d$, then the above is called an **integer linear program** (ILP). For some special parameter settings—e.g., when $\mathbf{b}$ is an integer vector and $\mathbf{A}$ is totally unimodular[5]—all vertices of the constraining polyhedron are integer points; in these cases, the integer constraint may be suppressed and (3) is guaranteed to have integer solutions (Schrijver, 2003). Of course, this need not happen: solving a general ILP is an NP-complete problem. Despite this fact, fast solvers are available today that make this a practical solution for many problems. Their performance depends on the dimensions and degree of sparsity of the constraint matrix $\mathbf{A}$.

Riedel and Clarke (2006) proposed an ILP formulation for dependency parsing which refines the arc-factored model by imposing linguistically motivated "hard" constraints that forbid some arc configurations. Their formulation includes an exponential number of constraints—one for each possible cycle. Since it is intractable to throw in all constraints at once, they propose a cutting-plane algorithm, where the cycle constraints are only invoked when violated by the current solution. The resulting algorithm is still slow, and an arc-factored model is used as a surrogate during training (i.e., the hard constraints are only used at test time), which implies a discrepancy between the model that is optimized and the one that is actually going to be used.

Here, we propose ILP formulations that eliminate the need for cycle constraints; in fact, they require only a *polynomial* number of constraints. Not only does our model allow expert knowledge to be injected in the form of constraints, it is also capable of *learning* soft versions of those constraints from data; indeed, it can handle features that are not arc-factored (correlating, for example, siblings and grandparents, modeling valency, or preferring nearly projective parses). While, as pointed out by McDonald and Satta (2007), the inclusion of these features makes inference NP-hard, by *relaxing* the integer constraints we obtain approximate algorithms that are very efficient and competitive with state-of-the-art methods. In this paper, we focus on unlabeled dependency parsing, for clarity of exposition. If it is extended to labeled parsing (a straightforward extension), our formulation fully subsumes that of Riedel and Clarke (2006), since it allows using the same hard constraints and features while keeping the ILP polynomial in size.

3.1 The Arborescence Polytope

We start by describing our *constraint space*. Our formulations rely on a concise polyhedral representation of the set of candidate dependency parse trees, as sketched in §2.1. This will be accomplished by drawing an analogy with a network flow problem.

Let $D = \langle V, A \rangle$ be the complete directed graph

[4] There is also a quadratic algorithm due to Tarjan (1977).

[5] A matrix is called totally unimodular if the determinants of each square submatrix belong to $\{0, 1, -1\}$.

associated with a sentence $x \in \mathcal{X}$, as stated in §2. A subgraph $y = \langle V, B \rangle$ is a legal dependency tree (i.e., $y \in \mathcal{Y}(x)$) if and only if the following conditions are met:

1. Each vertex in $V \setminus \{0\}$ must have exactly one incoming arc in B,
2. 0 has no incoming arcs in B,
3. B does not contain cycles.

For each vertex $v \in V$, let $\delta^-(v) \triangleq \{\langle i, j \rangle \in A \mid j = v\}$ denote its set of incoming arcs, and $\delta^+(v) \triangleq \{\langle i, j \rangle \in A \mid i = v\}$ denote its set of outgoing arcs. The two first conditions can be easily expressed by linear constraints on the incidence vector $\mathbf{z}$:

$$\sum_{a \in \delta^-(j)} z_a = 1, \quad j \in V \setminus \{0\} \tag{4}$$

$$\sum_{a \in \delta^-(0)} z_a = 0 \tag{5}$$

Condition 3 is somewhat harder to express. Rather than adding exponentially many constraints, one for each potential cycle (like Riedel and Clarke, 2006), we equivalently replace condition 3 by

3′. B is connected.

Note that conditions 1-2-3 are equivalent to 1-2-3′, in the sense that both define the same set $\mathcal{Y}(x)$. However, as we will see, the latter set of conditions is more convenient. Connectedness of graphs can be imposed via flow constraints (by requiring that, for any $v \in V \setminus \{0\}$, there is a directed path in B connecting 0 to v). We adapt the **single commodity flow** formulation for the (undirected) minimum spanning tree problem, due to Magnanti and Wolsey (1994), that requires $O(n^2)$ variables and constraints. Under this model, the root node must send one unit of flow to every other node. By making use of extra variables, $\boldsymbol{\phi} \triangleq \langle \phi_a \rangle_{a \in A}$, to denote the flow of commodities through each arc, we are led to the following constraints in addition to Eqs. 4–5 (we denote $\mathbb{U} \triangleq [0, 1]$, and $\mathbb{B} \triangleq \{0, 1\} = \mathbb{U} \cap \mathbb{Z}$):

- Root sends flow n:

$$\sum_{a \in \delta^+(0)} \phi_a = n \tag{6}$$

- Each node consumes one unit of flow:

$$\sum_{a \in \delta^-(j)} \phi_a - \sum_{a \in \delta^+(j)} \phi_a = 1, \quad j \in V \setminus \{0\} \tag{7}$$

- Flow is zero on disabled arcs:

$$\phi_a \leq n z_a, \quad a \in A \tag{8}$$

- Each arc indicator lies in the unit interval:

$$z_a \in \mathbb{U}, \quad a \in A. \tag{9}$$

These constraints project an outer bound of the arborescence polytope, i.e.,

$$\begin{aligned} \bar{\mathcal{Z}}(x) &\triangleq \{\mathbf{z} \in \mathbb{R}^{|A|} \mid (\mathbf{z}, \boldsymbol{\phi}) \text{ satisfy (4–9)}\} \\ &\supseteq \mathcal{Z}(x). \end{aligned} \tag{10}$$

Furthermore, the integer points of $\bar{\mathcal{Z}}(x)$ are precisely the incidence vectors of dependency trees in $\mathcal{Y}(x)$; these are obtained by replacing Eq. 9 by

$$z_a \in \mathbb{B}, \quad a \in A. \tag{11}$$

3.2 Arc-Factored Model

Given our polyhedral representation of (an outer bound of) the arborescence polytope, we can now formulate dependency parsing with an arc-factored model as an ILP. By storing the arc-local feature vectors into the columns of a matrix $\mathbf{F}(x) \triangleq [\mathbf{f}_a(x)]_{a \in A}$, and defining the score vector $\mathbf{s} \triangleq \mathbf{F}(x)^\top \mathbf{w}$ (each entry is an arc score) the inference problem can be written as

$$\begin{aligned} \max_{y \in \mathcal{Y}(x)} \mathbf{w}^\top \mathbf{f}(x, y) &= \max_{\mathbf{z} \in \mathcal{Z}(x)} \quad \mathbf{w}^\top \mathbf{F}(x) \mathbf{z} \\ &= \max_{\mathbf{z}, \boldsymbol{\phi}} \quad \mathbf{s}^\top \mathbf{z} \\ &\quad \text{s.t.} \quad \mathbf{A} \begin{bmatrix} \mathbf{z} \\ \boldsymbol{\phi} \end{bmatrix} \leq \mathbf{b} \\ &\quad \mathbf{z} \in \mathbb{B} \end{aligned} \tag{12}$$

where $\mathbf{A}$ is a sparse constraint matrix (with $O(|A|)$ non-zero elements), and $\mathbf{b}$ is the constraint vector; $\mathbf{A}$ and $\mathbf{b}$ encode the constraints (4–9). This is an ILP with $O(|A|)$ variables and constraints (hence, quadratic in n); if we drop the integer constraint the problem becomes the LP relaxation. As is, this formulation is no more attractive than solving the problem with the existing combinatorial algorithms discussed in §2.2; however, we can now start adding non-local features to build a more powerful model.

3.3 Sibling and Grandparent Features

To cope with higher-order features of the form $\mathbf{f}_{a_1, \ldots, a_K}(x)$ (i.e., features whose values depend on the simultaneous inclusion of arcs $a_1, \ldots, a_K$ on

a candidate dependency tree), we employ a linearization trick (Boros and Hammer, 2002), defining extra variables $z_{a_1 \dots a_K} \triangleq z_{a_1} \wedge \dots \wedge z_{a_K}$. This logical relation can be expressed by the following $O(K)$ agreement constraints:[6]

$$\begin{aligned} z_{a_1 \dots a_K} &\leq z_{a_i}, \quad i = 1, \dots, K \\ z_{a_1 \dots a_K} &\geq \textstyle\sum_{i=1}^{K} z_{a_i} - K + 1. \end{aligned} \tag{13}$$

As shown by McDonald and Pereira (2006) and Carreras (2007), the inclusion of features that correlate sibling and grandparent arcs may be highly beneficial, even if doing so requires resorting to approximate algorithms.[7] Define $\mathcal{R}^{\text{sibl}} \triangleq \{\langle i, j, k\rangle \,|\, \langle i, j\rangle \in A, \langle i, k\rangle \in A\}$ and $\mathcal{R}^{\text{grand}} \triangleq \{\langle i, j, k\rangle \,|\, \langle i, j\rangle \in A, \langle j, k\rangle \in A\}$. To include such features in our formulation, we need to add extra variables $\mathbf{z}^{\text{sibl}} \triangleq \langle z_r \rangle_{r \in \mathcal{R}^{\text{sibl}}}$ and $\mathbf{z}^{\text{grand}} \triangleq \langle z_r \rangle_{r \in \mathcal{R}^{\text{grand}}}$ that indicate the presence of sibling and grandparent arcs. Observe that these indicator variables are *conjunctions* of arc indicator variables, i.e., $z^{\text{sibl}}_{ijk} = z_{ij} \wedge z_{ik}$ and $z^{\text{grand}}_{ijk} = z_{ij} \wedge z_{jk}$. Hence, these features can be handled in our formulation by adding the following $O(|A| \cdot |V|)$ variables and constraints:

$$z^{\text{sibl}}_{ijk} \leq z_{ij}, \quad z^{\text{sibl}}_{ijk} \leq z_{ik}, \quad z^{\text{sibl}}_{ijk} \geq z_{ij} + z_{ik} - 1 \tag{14}$$

for all triples $\langle i, j, k\rangle \in \mathcal{R}^{\text{sibl}}$, and

$$z^{\text{grand}}_{ijk} \leq z_{ij}, \quad z^{\text{grand}}_{ijk} \leq z_{jk}, \quad z^{\text{grand}}_{ijk} \geq z_{ij} + z_{jk} - 1 \tag{15}$$

for all triples $\langle i, j, k\rangle \in \mathcal{R}^{\text{grand}}$. Let $\mathcal{R} \triangleq A \cup \mathcal{R}^{\text{sibl}} \cup \mathcal{R}^{\text{grand}}$; by redefining $\mathbf{z} \triangleq \langle z_r \rangle_{r \in \mathcal{R}}$ and $\mathbf{F}(x) \triangleq [\mathbf{f}_r(x)]_{r \in \mathcal{R}}$, we may express our inference problem as in Eq. 12, with $O(|A| \cdot |V|)$ variables and constraints.

Notice that the strategy just described to handle sibling features is not fully compatible with the features proposed by Eisner (1996) for projective parsing, as the latter correlate only *consecutive* siblings and are also able to place special features on the *first* child of a given word. The ability to handle such "ordered" features is intimately associated with Eisner's dynamic programming parsing algorithm and with the Markovian assumptions made explicitly by his generative model. We next show how similar features can be incorporated in our model by adding "dynamic" constraints to our ILP. Define:

$$z^{\text{next sibl}}_{ijk} \triangleq \begin{cases} 1 & \text{if } \langle i, j\rangle \text{ and } \langle i, k\rangle \text{ are} \\ & \text{consecutive siblings,} \\ 0 & \text{otherwise,} \end{cases}$$

$$z^{\text{first child}}_{ij} \triangleq \begin{cases} 1 & \text{if } j \text{ is the first child of } i, \\ 0 & \text{otherwise.} \end{cases}$$

Suppose (without loss of generality) that $i < j < k \leq n$. We could naively compose the constraints (14) with additional linear constraints that encode the logical relation

$$z^{\text{next sibl}}_{ijk} = z^{\text{sibl}}_{ijk} \wedge \textstyle\bigwedge_{j<l<k} \neg z_{il},$$

but this would yield a constraint matrix with $O(n^4)$ non-zero elements. Instead, we define auxiliary variables β_{jk} and γ_{ij}:

$$\begin{aligned} \beta_{jk} &= \begin{cases} 1, & \text{if } \exists l \text{ s.t. } \pi(l) = \pi(j) < j < l < k \\ 0, & \text{otherwise,} \end{cases} \\ \gamma_{ij} &= \begin{cases} 1, & \text{if } \exists k \text{ s.t. } i < k < j \text{ and } \langle i, k\rangle \in y \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{16}$$

Then, we have that $z^{\text{next sibl}}_{ijk} = z^{\text{sibl}}_{ijk} \wedge (\neg \beta_{jk})$ and $z^{\text{first child}}_{ij} = z_{ij} \wedge (\neg \gamma_{ij})$, which can be encoded via

$$\begin{array}{llll} z^{\text{next sibl}}_{ijk} \leq z^{\text{sibl}}_{ijk} & & z^{\text{first child}}_{ij} \leq z_{ij} & \\ z^{\text{next sibl}}_{ijk} \leq 1 - \beta_{jk} & & z^{\text{first child}}_{ij} \leq 1 - \gamma_{ij} & \\ z^{\text{next sibl}}_{ijk} \geq z^{\text{sibl}}_{ijk} - \beta_{jk} & & z^{\text{first child}}_{ij} \geq z_{ij} - \gamma_{ij} & \end{array}$$

The following "dynamic" constraints encode the logical relations for the auxiliary variables (16):

$$\begin{array}{llll} \beta_{j(j+1)} = 0 & & \gamma_{i(i+1)} = 0 & \\ \beta_{j(k+1)} \geq \beta_{jk} & & \gamma_{i(j+1)} \geq \gamma_{ij} & \\ \beta_{j(k+1)} \geq \displaystyle\sum_{i<j} z^{\text{sibl}}_{ijk} & & \gamma_{i(j+1)} \geq z_{ij} & \\ \beta_{j(k+1)} \leq \beta_{jk} + \displaystyle\sum_{i<j} z^{\text{sibl}}_{ijk} & & \gamma_{i(j+1)} \leq \gamma_{ij} + z_{ij} & \end{array}$$

Auxiliary variables and constraints are defined analogously for the case $n \geq i > j > k$. This results in a sparser constraint matrix, with only $O(n^3)$ non-zero elements.

3.4 Valency Features

A crucial fact about dependency grammars is that words have preferences about the number and arrangement of arguments and modifiers they accept. Therefore, it is desirable to include features

[6] Actually, any logical condition can be encoded with linear constraints involving binary variables; see e.g. Clarke and Lapata (2008) for an overview.

[7] By *sibling features* we mean features that depend on pairs of sibling arcs (i.e., of the form $\langle i, j\rangle$ and $\langle i, k\rangle$); by *grandparent features* we mean features that depend on pairs of grandparent arcs (of the form $\langle i, j\rangle$ and $\langle j, k\rangle$).

that indicate, for a candidate arborescence, how many outgoing arcs depart from each vertex; denote these quantities by $v_i \triangleq \sum_{a \in \delta^+(i)} z_a$, for each $i \in V$. We call v_i the **valency** of the ith vertex. We add valency indicators $z^{\text{val}}_{ik} \triangleq \mathbb{I}(v_i = k)$ for $i \in V$ and $k = 0, \ldots, n-1$. This way, we are able to penalize candidate dependency trees that assign unusual valencies to some of their vertices, by specifying a individual cost for each possible value of valency. The following $O(|V|^2)$ constraints encode the agreement between valency indicators and the other variables:

$$\begin{aligned} \textstyle\sum_{k=0}^{n-1} k z^{\text{val}}_{ik} &= \textstyle\sum_{a \in \delta^+(i)} z_a, \quad i \in V \qquad (17) \\ \textstyle\sum_{k=0}^{n-1} z^{\text{val}}_{ik} &= 1, \quad i \in V \\ z^{\text{val}}_{ik} &\geq 0, \quad i \in V, k \in \{0, \ldots, n-1\} \end{aligned}$$

3.5 Projectivity Features

For most languages, dependency parse trees tend to be nearly projective (cf. Buchholz and Marsi, 2006). We wish to make our model capable of learning to prefer "nearly" projective parses whenever that behavior is observed in the data.

The **multicommodity directed flow** model of Magnanti and Wolsey (1994) is a refinement of the model described in §3.1 which offers a compact and elegant way to indicate nonprojective arcs, requiring $O(n^3)$ variables and constraints. In this model, every node $k \neq 0$ defines a commodity: one unit of commodity k originates at the root node and must be delivered to node k; the variable ϕ^k_{ij} denotes the flow of commodity k in arc $\langle i, j \rangle$. We first replace (4–9) by (18–22):

- The root sends one unit of commodity to each node:

$$\sum_{a \in \delta^-(0)} \phi^k_a - \sum_{a \in \delta^+(0)} \phi^k_a = -1, \quad k \in V \setminus \{0\} \quad (18)$$

- Any node consumes its own commodity and no other:

$$\sum_{a \in \delta^-(j)} \phi^k_a - \sum_{a \in \delta^+(j)} \phi^k_a = \delta^k_j, \quad j, k \in V \setminus \{0\} \quad (19)$$

  where $\delta^k_j \triangleq \mathbb{I}(j = k)$ is the Kronecker delta.

- Disabled arcs do not carry any flow:

$$\phi^k_a \leq z_a, \quad a \in A, k \in V \qquad (20)$$

- There are exactly n enabled arcs:

$$\textstyle\sum_{a \in A} z_a = n \qquad (21)$$

- All variables lie in the unit interval:

$$z_a \in \mathbb{U}, \quad \phi^k_a \in \mathbb{U}, \quad a \in A, k \in V \qquad (22)$$

We next define auxiliary variables ψ_{jk} that indicate if there is a path from j to k. Since each vertex except the root has only one incoming arc, the following linear equalities are enough to describe these new variables:

$$\begin{aligned} \psi_{jk} &= \textstyle\sum_{a \in \delta^-(j)} \phi^k_a, \quad j, k \in V \setminus \{0\} \\ \psi_{0k} &= 1, \quad k \in V \setminus \{0\}. \qquad (23) \end{aligned}$$

Now, define indicators $\mathbf{z}^{\text{np}} \triangleq \langle z^{\text{np}}_a \rangle_{a \in A}$, where

$$z^{\text{np}}_a \triangleq \mathbb{I}(a \in y \text{ and } a \text{ is nonprojective}).$$

From the definition of projective arcs in §2.1, we have that $z^{\text{np}}_a = 1$ if and only if the arc is active ($z_a = 1$) and there is some vertex k in the span of $a = \langle i, j \rangle$ such that $\psi_{ik} = 0$. We are led to the following $O(|A| \cdot |V|)$ constraints for $\langle i, j \rangle \in A$:

$$\begin{aligned} z^{\text{np}}_{ij} &\leq z_{ij} \\ z^{\text{np}}_{ij} &\geq z_{ij} - \psi_{ik}, \quad \min(i,j) \leq k \leq \max(i,j) \\ z^{\text{np}}_{ij} &\leq -\textstyle\sum_{k=\min(i,j)+1}^{\max(i,j)-1} \psi_{ik} + |j - i| - 1 \end{aligned}$$

There are other ways to introduce nonprojectivity indicators and alternative definitions of "nonprojective arc." For example, by using dynamic constraints of the same kind as those in §3.3, we can indicate arcs that "cross" other arcs with $O(n^3)$ variables and constraints, and a cubic number of non-zero elements in the constraint matrix (omitted for space).

3.6 Projective Parsing

It would be straightforward to adapt the constraints in §3.5 to allow only projective parse trees: simply force $z^{\text{np}}_a = 0$ for any $a \in A$. But there are more efficient ways of accomplish this. While it is difficult to impose projectivity constraints *or* cycle constraints individually, there is a simpler way of imposing *both*. Consider 3 (or 3′) from §3.1.

Proposition 1 *Replace condition 3 (or 3′) with*

3″. If $\langle i, j \rangle \in B$, then, for any $k = 1, \ldots, n$ such that $k \neq j$, the parent of k must satisfy (defining $i' \triangleq \min(i, j)$ and $j' \triangleq \max(i, j)$):

$$\begin{cases} i' \leq \pi(k) \leq j', & \text{if } i' < k < j', \\ \pi(k) < i' \vee \pi(k) > j', & \text{if } k < i' \text{ or } k > j' \\ & \text{or } k = i. \end{cases}$$

*Then, $\mathcal{Y}(x)$ will be redefined as the set of **projective** dependency parse trees.*

We omit the proof for space. Conditions 1, 2, and $3''$ can be encoded with $O(n^2)$ constraints.

4 Experiments

We report experiments on seven languages, six (Danish, Dutch, Portuguese, Slovene, Swedish and Turkish) from the CoNLL-X shared task (Buchholz and Marsi, 2006), and one (English) from the CoNLL-2008 shared task (Surdeanu et al., 2008).[8] All experiments are evaluated using the *unlabeled attachment score* (UAS), using the default settings.[9] We used the same arc-factored features as McDonald et al. (2005) (included in the MSTParser toolkit[10]); for the higher-order models described in §3.3–3.5, we employed simple higher order features that look at the word, part-of-speech tag, and (if available) morphological information of the words being correlated through the indicator variables. For scalability (and noting that some of the models require $O(|V| \cdot |A|)$ constraints and variables, which, when $A = V^2$, grows cubically with the number of words), we first prune the base graph by running a simple algorithm that ranks the k-best candidate parents for each word in the sentence (we set $k = 10$); this reduces the number of candidate arcs to $|A| = kn$.[11] This strategy is similar to the one employed by Carreras et al. (2008) to prune the search space of the actual parser. The ranker is a local model trained using a max-margin criterion; it is arc-factored and not subject to *any* structural constraints, so it is very fast.

The actual parser was trained via the online structured passive-aggressive algorithm of Crammer et al. (2006); it differs from the 1-best MIRA algorithm of McDonald et al. (2005) by solving a sequence of *loss-augmented* inference problems.[12] The number of iterations was set to 10.

The results are summarized in Table 1; for the sake of comparison, we reproduced three strong baselines, all of them state-of-the-art parsers based on non-arc-factored models: the second order model of McDonald and Pereira (2006), the hybrid model of Nivre and McDonald (2008), which combines a (labeled) transition-based and a graph-based parser, and a refinement of the latter, due to Martins et al. (2008), which attempts to approximate non-local features.[13] We did not reproduce the model of Riedel and Clarke (2006) since the latter is tailored for *labeled* dependency parsing; however, experiments reported in that paper for Dutch (and extended to other languages in the CoNLL-X task) suggest that their model performs worse than our three baselines.

By looking at the middle four columns, we can see that adding non-arc-factored features makes the models more accurate, for all languages. With the exception of Portuguese, the best results are achieved with the full set of features. We can also observe that, for some languages, the valency features do not seem to help. Merely modeling the *number* of dependents of a word may not be as valuable as knowing what *kinds* of dependents they are (for example, distinguishing among arguments and adjuncts).

Comparing with the baselines, we observe that our full model outperforms that of McDonald and Pereira (2006), and is in line with the most accurate dependency parsers (Nivre and McDonald, 2008; Martins et al., 2008), obtained by combining transition-based and graph-based parsers.[14] Notice that our model, compared with these hybrid parsers, has the advantage of not requiring an ensemble configuration (eliminating, for example, the need to tune two parsers). Unlike the ensembles, it directly handles non-local output features by optimizing a single global objective. Perhaps more importantly, it makes it possible to exploit expert knowledge through the form of *hard global constraints*. Although not pursued here, the same kind of constraints employed by Riedel and Clarke (2006) can straightforwardly fit into our model, after extending it to perform labeled dependency parsing. We believe that a careful design of fea-

[8] We used the provided train/test splits except for English, for which we tested on the development partition. For training, sentences longer than 80 words were discarded. For testing, all sentences were kept (the longest one has length 118).

[9] `http://nextens.uvt.nl/~conll/software.html`

[10] `http://sourceforge.net/projects/mstparser`

[11] Note that, unlike reranking approaches, there are still exponentially many candidate parse trees after pruning. The oracle constrained to pick parents from these lists achieves $> 98\%$ in every case.

[12] The loss-augmented inference problem can also be expressed as an LP for Hamming loss functions that factor over arcs; we refer to Martins et al. (2009) for further details.

[13] Unlike our model, the hybrid models used here as baselines make use of the dependency labels at training time; indeed, the transition-based parser is trained to predict a *labeled* dependency parse tree, and the graph-based parser use these predicted labels as input features. Our model ignores this information at training time; therefore, this comparison is slightly unfair to us.

[14] See also Zhang and Clark (2008) for a different approach that combines transition-based and graph-based methods.

	[MP06]	[NM08]	[MDSX08]	ARC-FACTORED	+SIBL/GRANDP.	+VALENCY	+PROJ. (FULL)	FULL, RELAXED
DANISH	90.60	91.30	**91.54**	89.80	91.06	90.98	91.18	91.04 (-0.14)
DUTCH	84.11	84.19	84.79	83.55	84.65	84.93	**85.57**	85.41 (-0.16)
PORTUGUESE	91.40	91.81	**92.11**	90.66	**92.11**	92.01	91.42	91.44 (+0.02)
SLOVENE	83.67	85.09	85.13	83.93	85.13	85.45	**85.61**	85.41 (-0.20)
SWEDISH	89.05	90.54	90.50	89.09	90.50	90.34	**90.60**	90.52 (-0.08)
TURKISH	75.30	75.68	**76.36**	75.16	76.20	76.08	76.34	76.32 (-0.02)
ENGLISH	90.85	–	–	90.15	91.13	91.12	**91.16**	91.14 (-0.02)

Table 1: Results for nonprojective dependency parsing (unlabeled attachment scores). The three baselines are the second order model of McDonald and Pereira (2006) and the hybrid models of Nivre and McDonald (2008) and Martins et al. (2008). The four middle columns show the performance of our model using exact (ILP) inference at test time, for increasing sets of features (see §3.2–§3.5). The rightmost column shows the results obtained with the full set of features using relaxed LP inference followed by projection onto the feasible set. Differences are with respect to exact inference for the same set of features. Bold indicates the best result for a language. As for overall performance, both the exact and relaxed full model outperform the arc-factored model and the second order model of McDonald and Pereira (2006) with statistical significance ($p < 0.01$) according to Dan Bikel's randomized method (`http://www.cis.upenn.edu/~dbikel/software.html`).

tures and constraints can lead to further improvements on accuracy.

We now turn to a different issue: scalability. In previous work (Martins et al., 2009), we showed that training the model via LP-relaxed inference (as we do here) makes it learn to avoid fractional solutions; as a consequence, ILP solvers will converge faster to the optimum (on average). Yet, it is known from worst case complexity theory that solving a general ILP is NP-hard; hence, these solvers may not scale well with the sentence length. Merely considering the LP-relaxed version of the problem at test time is unsatisfactory, as it may lead to a fractional solution (i.e., a solution whose components indexed by arcs, $\tilde{\mathbf{z}} = \langle z_a \rangle_{a \in A}$, are not all integer), which does not correspond to a valid dependency tree. We propose the following approximate algorithm to obtain an actual parse: first, solve the LP relaxation (which can be done in polynomial time with interior-point methods); then, if the solution is fractional, project it onto the feasible set $\mathcal{Y}(x)$. Fortunately, the Euclidean projection can be computed in a straightforward way by finding a maximal arborescence in the directed graph whose weights are defined by $\tilde{\mathbf{z}}$ (we omit the proof for space); as we saw in §2.2, the Chu-Liu-Edmonds algorithm can do this in polynomial time. The overall parsing runtime becomes polynomial with respect to the length of the sentence.

The last column of Table 1 compares the accuracy of this approximate method with the exact one. We observe that there is not a substantial drop in accuracy; on the other hand, we observed a considerable speed-up with respect to exact inference, particularly for long sentences. The average runtime (across all languages) is 0.632 seconds per sentence, which is in line with existing higher-order parsers and is much faster than the runtimes reported by Riedel and Clarke (2006).

5 Conclusions

We presented new dependency parsers based on concise ILP formulations. We have shown how non-local output features can be incorporated, while keeping only a polynomial number of constraints. These features can act as soft constraints whose penalty values are automatically learned from data; in addition, our model is also compatible with expert knowledge in the form of hard constraints. Learning through a max-margin framework is made effective by the means of a LP-relaxation. Experimental results on seven languages show that our rich-featured parsers outperform arc-factored and approximate higher-order parsers, and are in line with stacked parsers, having with respect to the latter the advantage of not requiring an ensemble configuration.

Acknowledgments

The authors thank the reviewers for their comments. Martins was supported by a grant from FCT/ICTI through the CMU-Portugal Program, and also by Priberam Informática. Smith was supported by NSF IIS-0836431 and an IBM Faculty Award. Xing was supported by NSF DBI-0546594, DBI-0640543, IIS-0713379, and an Alfred Sloan Foundation Fellowship in Computer Science.

References

E. Boros and P.L. Hammer. 2002. Pseudo-Boolean optimization. *Discrete Applied Mathematics*, 123(1–3):155–225.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

X. Carreras, M. Collins, and T. Koo. 2008. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proc. of CoNLL*.

X. Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proc. of CoNLL*.

M. Chang, L. Ratinov, and D. Roth. 2008. Constraints as prior knowledge. In *ICML Workshop on Prior Knowledge for Text and Language Processing*.

Y. J. Chu and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

J. Clarke and M. Lapata. 2008. Global inference for sentence compression an integer linear programming approach. *JAIR*, 31:399–429.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online passive-aggressive algorithms. *JMLR*, 7:551–585.

A. Culotta and J. Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proc. of ACL*.

P. Denis and J. Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *Proc. of HLT-NAACL*.

Y. Ding and M. Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammar. In *Proc. of ACL*.

J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proc. of ACL*.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING*.

S. Kahane, A. Nasr, and O. Rambow. 1998. Pseudo-projectivity: a polynomially parsable non-projective dependency grammar. In *Proc. of COLING-ACL*.

S. Lacoste-Julien, B. Taskar, D. Klein, and M. I. Jordan. 2006. Word alignment via quadratic assignment. In *Proc. of HLT-NAACL*.

T. L. Magnanti and L. A. Wolsey. 1994. Optimal Trees. Technical Report 290-94, Massachusetts Institute of Technology, Operations Research Center.

A. F. T. Martins, D. Das, N. A. Smith, and E. P. Xing. 2008. Stacking dependency parsers. In *Proc. of EMNLP*.

A. F. T. Martins, N. A. Smith, and E. P. Xing. 2009. Polyhedral outer approximations with application to natural language parsing. In *Proc. of ICML*.

R. T. McDonald and F. C. N. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL*.

R. McDonald and G. Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proc. of IWPT*.

R. T. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*.

J. Nivre and R. McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proc. of ACL-HLT*.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proc. of COLING*.

M. Richardson and P. Domingos. 2006. Markov logic networks. *Machine Learning*, 62(1):107–136.

S. Riedel and J. Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proc. of EMNLP*.

R. T. Rockafellar. 1970. *Convex Analysis*. Princeton University Press.

D. Roth and W. T. Yih. 2005. Integer linear programming inference for conditional random fields. In *ICML*.

A. Schrijver. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer.

D. A. Smith and J. Eisner. 2008. Dependency parsing by belief propagation. In *Proc. of EMNLP*.

M. Surdeanu, R. Johansson, A. Meyers, L. Màrquez, and J. Nivre. 2008. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. *Proc. of CoNLL*.

R. E. Tarjan. 1977. Finding optimum branchings. *Networks*, 7(1):25–36.

M. Wang, N. A. Smith, and T. Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proceedings of EMNLP-CoNLL*.

Y. Zhang and S. Clark. 2008. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proc. of EMNLP*.

Non-Projective Dependency Parsing in Expected Linear Time

Joakim Nivre
Uppsala University, Department of Linguistics and Philology, SE-75126 Uppsala
Växjö University, School of Mathematics and Systems Engineering, SE-35195 Växjö
E-mail: joakim.nivre@lingfil.uu.se

Abstract

We present a novel transition system for dependency parsing, which constructs arcs only between adjacent words but can parse arbitrary non-projective trees by swapping the order of words in the input. Adding the swapping operation changes the time complexity for deterministic parsing from linear to quadratic in the worst case, but empirical estimates based on treebank data show that the expected running time is in fact linear for the range of data attested in the corpora. Evaluation on data from five languages shows state-of-the-art accuracy, with especially good results for the labeled exact match score.

1 Introduction

Syntactic parsing using dependency structures has become a standard technique in natural language processing with many different parsing models, in particular data-driven models that can be trained on syntactically annotated corpora (Yamada and Matsumoto, 2003; Nivre et al., 2004; McDonald et al., 2005a; Attardi, 2006; Titov and Henderson, 2007). A hallmark of many of these models is that they can be implemented very efficiently. Thus, transition-based parsers normally run in linear or quadratic time, using greedy deterministic search or fixed-width beam search (Nivre et al., 2004; Attardi, 2006; Johansson and Nugues, 2007; Titov and Henderson, 2007), and graph-based models support exact inference in at most cubic time, which is efficient enough to make global discriminative training practically feasible (McDonald et al., 2005a; McDonald et al., 2005b).

However, one problem that still has not found a satisfactory solution in data-driven dependency parsing is the treatment of discontinuous syntactic constructions, usually modeled by non-projective dependency trees, as illustrated in Figure 1. In a projective dependency tree, the yield of every subtree is a contiguous substring of the sentence. This is not the case for the tree in Figure 1, where the subtrees rooted at node 2 (*hearing*) and node 4 (*scheduled*) both have discontinuous yields.

Allowing non-projective trees generally makes parsing *computationally* harder. Exact inference for parsing models that allow non-projective trees is NP hard, except under very restricted independence assumptions (Neuhaus and Bröker, 1997; McDonald and Pereira, 2006; McDonald and Satta, 2007). There is recent work on algorithms that can cope with important subsets of all non-projective trees in polynomial time (Kuhlmann and Satta, 2009; Gómez-Rodríguez et al., 2009), but the time complexity is at best $O(n^6)$, which can be problematic in practical applications. Even the best algorithms for deterministic parsing run in quadratic time, rather than linear (Nivre, 2008a), unless restricted to a subset of non-projective structures as in Attardi (2006) and Nivre (2007).

But allowing non-projective dependency trees also makes parsing *empirically* harder, because it requires that we model relations between non-adjacent structures over potentially unbounded distances, which often has a negative impact on parsing accuracy. On the other hand, it is hardly possible to ignore non-projective structures completely, given that 25% or more of the sentences in some languages cannot be given a linguistically adequate analysis without invoking non-projective structures (Nivre, 2006; Kuhlmann and Nivre, 2006; Havelka, 2007).

Current approaches to data-driven dependency parsing typically use one of two strategies to deal with non-projective trees (unless they ignore them completely). Either they employ a non-standard parsing algorithm that can combine non-adjacent substructures (McDonald et al., 2005b; Attardi, 2006; Nivre, 2007), or they try to recover non-

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 351–359,
Suntec, Singapore, 2-7 August 2009.

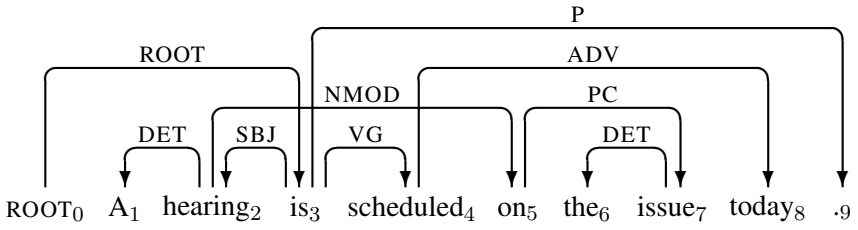


Figure 1: Dependency tree for an English sentence (non-projective).

projective dependencies by post-processing the output of a strictly projective parser (Nivre and Nilsson, 2005; Hall and Novák, 2005; McDonald and Pereira, 2006). In this paper, we will adopt a different strategy, suggested in recent work by Nivre (2008b) and Titov et al. (2009), and propose an algorithm that only combines adjacent substructures but derives non-projective trees by reordering the input words.

The rest of the paper is structured as follows. In Section 2, we define the formal representations needed and introduce the framework of transition-based dependency parsing. In Section 3, we first define a minimal transition system and explain how it can be used to perform projective dependency parsing in linear time; we then extend the system with a single transition for swapping the order of words in the input and demonstrate that the extended system can be used to parse unrestricted dependency trees with a time complexity that is quadratic in the worst case but still linear in the best case. In Section 4, we present experiments indicating that the expected running time of the new system on naturally occurring data is in fact linear and that the system achieves state-of-the-art parsing accuracy. We discuss related work in Section 5 and conclude in Section 6.

2 Background Notions

2.1 Dependency Graphs and Trees

Given a set L of dependency labels, a *dependency graph* for a sentence $x = w_1, \ldots, w_n$ is a directed graph $G = (V_x, A)$, where

1. $V_x = \{0, 1, \ldots, n\}$ is a set of nodes,
2. $A \subseteq V_x \times L \times V_x$ is a set of labeled arcs.

The set V_x of *nodes* is the set of positive integers up to and including n, each corresponding to the linear position of a word in the sentence, plus an extra artificial root node 0. The set A of *arcs* is a set of triples (i, l, j), where i and j are nodes and l is a label. For a dependency graph $G = (V_x, A)$ to be well-formed, we in addition require that it is a *tree* rooted at the node 0, as illustrated in Figure 1.

2.2 Transition Systems

Following Nivre (2008a), we define a *transition system* for dependency parsing as a quadruple $S = (C, T, c_s, C_t)$, where

1. C is a set of *configurations*,
2. T is a set of *transitions*, each of which is a (partial) function $t : C \rightarrow C$,
3. c_s is an *initialization function*, mapping a sentence $x = w_1, \ldots, w_n$ to a configuration $c \in C$,
4. $C_t \subseteq C$ is a set of *terminal configurations*.

In this paper, we take the set C of configurations to be the set of all triples $c = (\Sigma, B, A)$ such that Σ and B are disjoint sublists of the nodes V_x of some sentence x, and A is a set of dependency arcs over V_x (and some label set L); we take the initial configuration for a sentence $x = w_1, \ldots, w_n$ to be $c_s(x) = ([0], [1, \ldots, n], \{\,\})$; and we take the set C_t of terminal configurations to be the set of all configurations of the form $c = ([0], [\,], A)$ (for any arc set A). The set T of transitions will be discussed in detail in Sections 3.1–3.2.

We will refer to the list Σ as the *stack* and the list B as the *buffer*, and we will use the variables σ and β for arbitrary sublists of Σ and B, respectively. For reasons of perspicuity, we will write Σ with its head (top) to the right and B with its head to the left. Thus, $c = ([\sigma|i], [j|\beta], A)$ is a configuration with the node i on top of the stack Σ and the node j as the first node in the buffer B.

Given a transition system $S = (C, T, c_s, C_t)$, a *transition sequence* for a sentence x is a sequence $C_{0,m} = (c_0, c_1, \ldots, c_m)$ of configurations, such that

1. $c_0 = c_s(x)$,
2. $c_m \in C_t$,
3. for every i $(1 \leq i \leq m)$, $c_i = t(c_{i-1})$ for some $t \in T$.

Transition		Condition
LEFT-ARC$_l$	$([\sigma\|i,j],B,A) \Rightarrow ([\sigma\|j],B,A\cup\{(j,l,i)\})$	$i \neq 0$
RIGHT-ARC$_l$	$([\sigma\|i,j],B,A) \Rightarrow ([\sigma\|i],B,A\cup\{(i,l,j)\})$	
SHIFT	$(\sigma,[i\|\beta],A) \Rightarrow ([\sigma\|i],\beta,A)$	
SWAP	$([\sigma\|i,j],\beta,A) \Rightarrow ([\sigma\|j],[i\|\beta],A)$	$0 < i < j$

Figure 2: Transitions for dependency parsing; $T_p = \{\text{LEFT-ARC}_l, \text{RIGHT-ARC}_l, \text{SHIFT}\}$; $T_u = T_p \cup \{\text{SWAP}\}$.

The *parse* assigned to S by $C_{0,m}$ is the dependency graph $G_{c_m} = (V_x, A_{c_m})$, where A_{c_m} is the set of arcs in c_m.

A transition system S is *sound* for a class $\mathbb{G}$ of dependency graphs iff, for every sentence x and transition sequence $C_{0,m}$ for x in S, $G_{c_m} \in \mathbb{G}$. S is *complete* for $\mathbb{G}$ iff, for every sentence x and dependency graph G for x in $\mathbb{G}$, there is a transition sequence $C_{0,m}$ for x in S such that $G_{c_m} = G$.

2.3 Deterministic Transition-Based Parsing

An *oracle* for a transition system S is a function $o : C \rightarrow T$. Ideally, o should always return the optimal transition t for a given configuration c, but all we require formally is that it respects the preconditions of transitions in T. That is, if $o(c) = t$ then t is permissible in c. Given an oracle o, deterministic transition-based parsing can be achieved by the following simple algorithm:

PARSE(o, x)
1 $c \leftarrow c_s(x)$
2 **while** $c \notin C_t$
3 **do** $t \leftarrow o(c)$; $c \leftarrow t(c)$
4 **return** G_c

Starting in the initial configuration $c_s(x)$, the parser repeatedly calls the oracle function o for the current configuration c and updates c according to the oracle transition t. The iteration stops when a terminal configuration is reached. It is easy to see that, provided that there is at least one transition sequence in S for every sentence, the parser constructs exactly one transition sequence $C_{0,m}$ for a sentence x and returns the parse defined by the terminal configuration c_m, i.e., $G_{c_m} = (V_x, A_{c_m})$. Assuming that the calls $o(c)$ and $t(c)$ can both be performed in constant time, the worst-case time complexity of a deterministic parser based on a transition system S is given by an upper bound on the length of transition sequences in S.

When building practical parsing systems, the oracle can be approximated by a classifier trained on treebank data, a technique that has been used successfully in a number of systems (Yamada and Matsumoto, 2003; Nivre et al., 2004; Attardi, 2006). This is also the approach we will take in the experimental evaluation in Section 4.

3 Transitions for Dependency Parsing

Having defined the set of configurations, including initial and terminal configurations, we will now focus on the transition set T required for dependency parsing. The total set of transitions that will be considered is given in Figure 2, but we will start in Section 3.1 with the subset T_p (p for projective) consisting of the first three. In Section 3.2, we will add the fourth transition (SWAP) to get the full transition set T_u (u for unrestricted).

3.1 Projective Dependency Parsing

The minimal transition set T_p for projective dependency parsing contains three transitions:

1. LEFT-ARC$_l$ updates a configuration with i, j on top of the stack by adding (j, l, i) to A and replacing i, j on the stack by j alone. It is permissible as long as i is distinct from 0.
2. RIGHT-ARC$_l$ updates a configuration with i, j on top of the stack by adding (i, l, j) to A and replacing i, j on the stack by i alone.
3. SHIFT updates a configuration with i as the first node of the buffer by removing i from the buffer and pushing it onto the stack.

The system $S_p = (C, T_p, c_s, C_t)$ is sound and complete for the set of projective dependency trees (over some label set L) and has been used, in slightly different variants, by a number of transition-based dependency parsers (Yamada and Matsumoto, 2003; Nivre, 2004; Attardi, 2006;

Transition	Stack (Σ)	Buffer (B)	Added Arc
	$[\text{ROOT}_0]$	$[\text{A}_1, \ldots, .9]$	
SHIFT	$[\text{ROOT}_0, \text{A}_1]$	$[\text{hearing}_2, \ldots, .9]$	
SHIFT	$[\text{ROOT}_0, \text{A}_1, \text{hearing}_2]$	$[\text{is}_3, \ldots, .9]$	
LA_{DET}	$[\text{ROOT}_0, \text{hearing}_2]$	$[\text{is}_3, \ldots, .9]$	$(2, \text{DET}, 1)$
SHIFT	$[\text{ROOT}_0, \text{hearing}_2, \text{is}_3]$	$[\text{scheduled}_4, \ldots, .9]$	
SHIFT	$[\text{ROOT}_0, \ldots, \text{is}_3, \text{scheduled}_4]$	$[\text{on}_5, \ldots, .9]$	
SHIFT	$[\text{ROOT}_0, \ldots, \text{scheduled}_4, \text{on}_5]$	$[\text{the}_6, \ldots, .9]$	
SWAP	$[\text{ROOT}_0, \ldots, \text{is}_3, \text{on}_5]$	$[\text{scheduled}_4, \ldots, .9]$	
SWAP	$[\text{ROOT}_0, \text{hearing}_2, \text{on}_5]$	$[\text{is}_3, \ldots, .9]$	
SHIFT	$[\text{ROOT}_0, \ldots, \text{on}_5, \text{is}_3]$	$[\text{scheduled}_4, \ldots, .9]$	
SHIFT	$[\text{ROOT}_0, \ldots, \text{is}_3, \text{scheduled}_4]$	$[\text{the}_6, \ldots, .9]$	
SHIFT	$[\text{ROOT}_0, \ldots, \text{scheduled}_4, \text{the}_6]$	$[\text{issue}_7, \ldots, .9]$	
SWAP	$[\text{ROOT}_0, \ldots, \text{is}_3, \text{the}_6]$	$[\text{scheduled}_4, \ldots, .9]$	
SWAP	$[\text{ROOT}_0, \ldots, \text{on}_5, \text{the}_6]$	$[\text{is}_3, \ldots, .9]$	
SHIFT	$[\text{ROOT}_0, \ldots, \text{the}_6, \text{is}_3]$	$[\text{scheduled}_4, \ldots, .9]$	
SHIFT	$[\text{ROOT}_0, \ldots, \text{is}_3, \text{scheduled}_4]$	$[\text{issue}_7, \ldots, .9]$	
SHIFT	$[\text{ROOT}_0, \ldots, \text{scheduled}_4, \text{issue}_7]$	$[\text{today}_8, .9]$	
SWAP	$[\text{ROOT}_0, \ldots, \text{is}_3, \text{issue}_7]$	$[\text{scheduled}_4, \ldots, .9]$	
SWAP	$[\text{ROOT}_0, \ldots, \text{the}_6, \text{issue}_7]$	$[\text{is}_3, \ldots, .9]$	
LA_{DET}	$[\text{ROOT}_0, \ldots, \text{on}_5, \text{issue}_7]$	$[\text{is}_3, \ldots, .9]$	$(7, \text{DET}, 6)$
RA_{PC}	$[\text{ROOT}_0, \text{hearing}_2, \text{on}_5]$	$[\text{is}_3, \ldots, .9]$	$(5, \text{PC}, 7)$
RA_{NMOD}	$[\text{ROOT}_0, \text{hearing}_2]$	$[\text{is}_3, \ldots, .9]$	$(2, \text{NMOD}, 5)$
SHIFT	$[\text{ROOT}_0, \ldots, \text{hearing}_2, \text{is}_3]$	$[\text{scheduled}_4, \ldots, .9]$	
LA_{SBJ}	$[\text{ROOT}_0, \text{is}_3]$	$[\text{scheduled}_4, \ldots, .9]$	$(3, \text{SBJ}, 2)$
SHIFT	$[\text{ROOT}_0, \text{is}_3, \text{scheduled}_4]$	$[\text{today}_8, .9]$	
SHIFT	$[\text{ROOT}_0, \ldots, \text{scheduled}_4, \text{today}_8]$	$[.9]$	
RA_{ADV}	$[\text{ROOT}_0, \text{is}_3, \text{scheduled}_4]$	$[.9]$	$(4, \text{ADV}, 8)$
RA_{VG}	$[\text{ROOT}_0, \text{is}_3]$	$[.9]$	$(3, \text{VG}, 4)$
SHIFT	$[\text{ROOT}_0, \text{is}_3, .9]$	$[\,]$	
RA_{P}	$[\text{ROOT}_0, \text{is}_3]$	$[\,]$	$(3, \text{P}, 9)$
RA_{ROOT}	$[\text{ROOT}_0]$	$[\,]$	$(0, \text{ROOT}, 3)$

Figure 3: Transition sequence for parsing the sentence in Figure 1 (LA = LEFT-ARC, RA = REFT-ARC).

Nivre, 2008a). For proofs of soundness and completeness, see Nivre (2008a).

As noted in section 2, the worst-case time complexity of a deterministic transition-based parser is given by an upper bound on the length of transition sequences. In S_p, the number of transitions for a sentence $x = w_1, \ldots, w_n$ is always exactly $2n$, since a terminal configuration can only be reached after n SHIFT transitions (moving nodes $1, \ldots, n$ from B to Σ) and n applications of LEFT-ARC$_l$ or RIGHT-ARC$_l$ (removing the same nodes from Σ). Hence, the complexity of deterministic parsing is $O(n)$ in the worst case (as well as in the best case).

3.2 Unrestricted Dependency Parsing

We now consider what happens when we add the fourth transition from Figure 2 to get the extended transition set T_u. The SWAP transition updates a configuration with stack $[\sigma|i, j]$ by moving the node i back to the buffer. This has the effect that the order of the nodes i and j in the appended list $\Sigma + B$ is reversed compared to the original word order in the sentence. It is important to note that SWAP is only permissible when the two nodes on top of the stack are in the original word order, which prevents the same two nodes from being swapped more than once, and when the leftmost node i is distinct from the root node 0. Note also that SWAP moves the node i back to the buffer, so that LEFT-ARC$_l$, RIGHT-ARC$_l$ or SWAP can subsequently apply with the node j on top of the stack.

The fact that we can swap the order of nodes, implicitly representing subtrees, means that we can construct non-projective trees by applying

$$o(c) = \begin{cases} \text{LEFT-ARC}_l & \text{if } c = ([\sigma|i,j], B, A_c),\ (j,l,i) \in A \text{ and } A^i \subseteq A_c \\ \text{RIGHT-ARC}_l & \text{if } c = ([\sigma|i,j], B, A_c),\ (i,l,j) \in A \text{ and } A^j \subseteq A_c \\ \text{SWAP} & \text{if } c = ([\sigma|i,j], B, A_c) \text{ and } j <_G i \\ \text{SHIFT} & \text{otherwise} \end{cases}$$

Figure 4: Oracle function for $S_u = (C, T_u, c_s, C_t)$ with target tree $G = (V_x, A)$. We use the notation A^i to denote the subset of A that only contains the outgoing arcs of the node i.

LEFT-ARC$_l$ or RIGHT-ARC$_l$ to subtrees whose yields are not adjacent according to the original word order. This is illustrated in Figure 3, which shows the transition sequence needed to parse the example in Figure 1. For readability, we represent both the stack Σ and the buffer B as lists of tokens, indexed by position, rather than abstract nodes. The last column records the arc that is added to the arc set A in a given transition (if any).

Given the simplicity of the extension, it is rather remarkable that the system $S_u = (C, T_u, c_s, C_t)$ is sound and complete for the set of all dependency trees (over some label set L), including all non-projective trees. The soundness part is trivial, since any terminating transition sequence will have to move all the nodes $1, \ldots, n$ from B to Σ (using SHIFT) and then remove them from Σ (using LEFT-ARC$_l$ or RIGHT-ARC$_l$), which will produce a tree with root 0.

For completeness, we note first that projectivity is not a property of a dependency tree in itself, but of the tree in combination with a word order, and that a tree can always be made projective by reordering the nodes. For instance, let x be a sentence with dependency tree $G = (V_x, A)$, and let $<_G$ be the total order on V_x defined by an inorder traversal of G that respects the local ordering of a node and its children given by the original word order. Regardless of whether G is projective with respect to x, it must by necessity be projective with respect to $<_G$. We call $<_G$ the *projective order* corresponding to x and G and use it as our canonical way of finding a node order that makes the tree projective. By way of illustration, the projective order for the sentence and tree in Figure 1 is: $\text{A}_1 <_G \text{hearing}_2 <_G \text{on}_5 <_G \text{the}_6 <_G \text{issue}_7 <_G \text{is}_3 <_G \text{scheduled}_4 <_G \text{today}_8 <_G .9$.

If the words of a sentence x with dependency tree G are already in projective order, this means that G is projective with respect to x and that we can parse the sentence using only transitions in T_p, because nodes can be pushed onto the stack in projective order using only the SHIFT transition. If the words are *not* in projective order, we can use a combination of SHIFT and SWAP transitions to ensure that nodes are still pushed onto the stack in projective order. More precisely, if the next node in the projective order is the kth node in the buffer, we perform k SHIFT transitions, to get this node onto the stack, followed by $k-1$ SWAP transitions, to move the preceding $k-1$ nodes back to the buffer.[1] In this way, the parser can effectively sort the input nodes into projective order on the stack, repeatedly extracting the minimal element of $<_G$ from the buffer, and build a tree that is projective with respect to the sorted order. Since any input can be sorted using SHIFT and SWAP, and any projective tree can be built using SHIFT, LEFT-ARC$_l$ and RIGHT-ARC$_l$, the system S_u is complete for the set of all dependency trees.

In Figure 4, we define an oracle function o for the system S_u, which implements this "sort and parse" strategy and predicts the optimal transition t out of the current configuration c, given the target dependency tree $G = (V_x, A)$ and the projective order $<_G$. The oracle predicts LEFT-ARC$_l$ or RIGHT-ARC$_l$ if the two top nodes on the stack should be connected by an arc and if the dependent node of this arc is already connected to all its dependents; it predicts SWAP if the two top nodes are not in projective order; and it predicts SHIFT otherwise. This is the oracle that has been used to generate training data for classifiers in the experimental evaluation in Section 4.

Let us now consider the time complexity of the extended system $S_u = (C, T_u, c_s, C_t)$ and let us begin by observing that $2n$ is still a *lower* bound on the number of transitions required to reach a terminal configuration. A sequence of $2n$ transi-

[1] This can be seen in Figure 3, where transitions 4–8, 9–13, and 14–18 are the transitions needed to make sure that *on*$_5$, *the*$_6$ and *issue*$_7$ are processed on the stack before *is*$_3$ and *scheduled*$_4$.

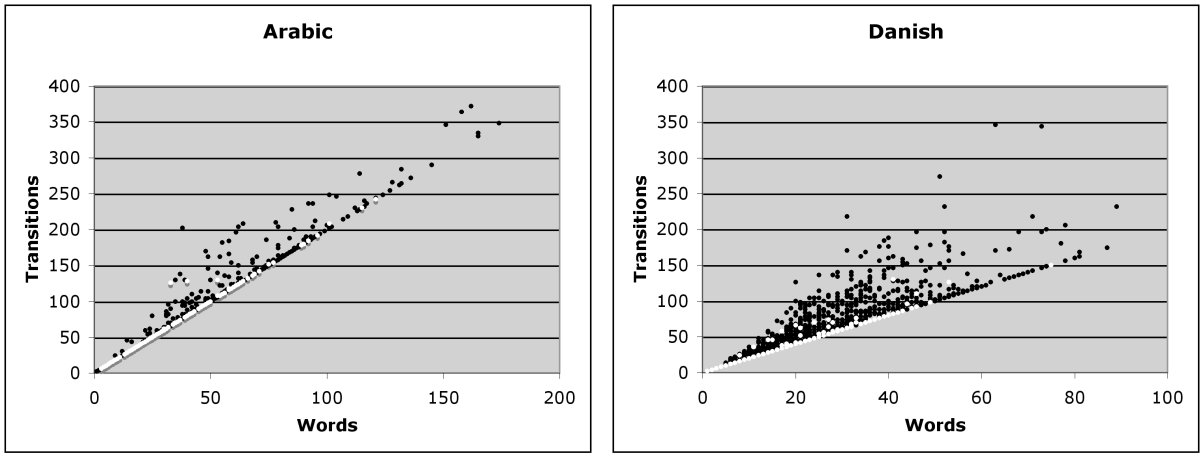


Figure 5: Abstract running time during training (black) and parsing (white) for Arabic (1460/146 sentences) and Danish (5190/322 sentences).

tions occurs when no SWAP transitions are performed, in which case the behavior of the system is identical to the simpler system S_p. This is important, because it means that the best-case complexity of the deterministic parser is still $O(n)$ and that the we can expect to observe the best case for all sentences with projective dependency trees.

The exact number of additional transitions needed to reach a terminal configuration is determined by the number of SWAP transitions. Since SWAP moves one node from Σ to B, there will be one additional SHIFT for every SWAP, which means that the total number of transitions is $2n + 2k$, where k is the number of SWAP transitions. Given the condition that SWAP can only apply in a configuration $c = ([\sigma|i, j], B, A)$ if $0 < i < j$, the number of SWAP transitions is bounded by $\frac{n(n-1)}{2}$, which means that $2n + n(n-1) = n + n^2$ is an upper bound on the number of transitions in a terminating sequence. Hence, the worst-case complexity of the deterministic parser is $O(n^2)$.

The running time of a deterministic transition-based parser using the system S_u is $O(n)$ in the best case and $O(n^2)$ in the worst case. But what about the average case? Empirical studies, based on data from a wide range of languages, have shown that dependency trees tend to be projective and that most non-projective trees only contain a small number of discontinuities (Nivre, 2006; Kuhlmann and Nivre, 2006; Havelka, 2007). This should mean that the expected number of swaps per sentence is small, and that the running time is linear on average for the range of inputs that occur in natural languages. This is a hypothesis that will be tested experimentally in the next section.

4 Experiments

Our experiments are based on five data sets from the CoNLL-X shared task: Arabic, Czech, Danish, Slovene, and Turkish (Buchholz and Marsi, 2006). These languages have been selected because the data come from genuine dependency treebanks, whereas all the other data sets are based on some kind of conversion from another type of representation, which could potentially distort the distribution of different types of structures in the data.

4.1 Running Time

In section 3.2, we hypothesized that the expected running time of a deterministic parser using the transition system S_u would be linear, rather than quadratic. To test this hypothesis, we examine how the number of transitions varies as a function of sentence length. We call this the *abstract running time*, since it abstracts over the actual time needed to compute each oracle prediction and transition, which is normally constant but dependent on the type of classifier used.

We first measured the abstract running time on the training sets, using the oracle to derive the transition sequence for every sentence, to see how many transitions are required in the ideal case. We then performed the same measurement on the test sets, using classifiers trained on the oracle transition sequences from the training sets (as described below in Section 4.2), to see whether the trained parsers deviate from the ideal case.

The result for Arabic and Danish can be seen

System	Arabic		Czech		Danish		Slovene		Turkish	
	AS	EM	AS	EM	AS	EM	AS	EM	AS	EM
S_u	67.1 (9.1)	**11.6**	**82.4** (**73.8**)	**35.3**	84.2 (22.5)	26.7	75.2 (23.0)	**29.9**	64.9 (**11.8**)	**21.5**
S_p	67.3 (**18.2**)	**11.6**	80.9 (3.7)	31.2	84.6 (0.0)	27.0	74.2 (3.4)	**29.9**	65.3 (6.6)	21.0
S_{pp}	67.2 (**18.2**)	**11.6**	82.1 (60.7)	34.0	84.7 (22.5)	28.9	74.8 (20.7)	26.9	65.5 (**11.8**)	20.7
Malt-06	66.7 (**18.2**)	11.0	78.4 (57.9)	27.4	84.8 (27.5)	26.7	70.3 (20.7)	19.7	65.7 (9.2)	19.3
MST-06	66.9 (0.0)	10.3	80.2 (61.7)	29.9	84.8 (**62.5**)	25.5	73.4 (26.4)	20.9	63.2 (**11.8**)	20.2
MST_{Malt}	**68.6** (9.4)	11.0	82.3 (69.2)	31.2	**86.7** (60.0)	**29.8**	**75.9** (**27.6**)	26.6	**66.3** (9.2)	18.6

Table 1: Labeled accuracy; AS = attachment score (non-projective arcs in brackets); EM = exact match.

in Figure 5, where black dots represent training sentences (parsed with the oracle) and white dots represent test sentences (parsed with a classifier). For Arabic there is a very clear linear relationship in both cases with very few outliers. Fitting the data with a linear function using the least squares method gives us $m = 2.06n$ ($R^2 = 0.97$) for the training data and $m = 2.02n$ ($R^2 = 0.98$) for the test data, where m is the number of transitions in parsing a sentence of length n. For Danish, there is clearly more variation, especially for the training data, but the least-squares approximation still explains most of the variance, with $m = 2.22n$ ($R^2 = 0.85$) for the training data and $m = 2.07n$ ($R^2 = 0.96$) for the test data. For both languages, we thus see that the classifier-based parsers have a lower mean number of transitions and less variance than the oracle parsers. And in both cases, the expected number of transitions is only marginally greater than the $2n$ of the strictly projective transition system S_p.

We have chosen to display results for Arabic and Danish because they are the two extremes in our sample. Arabic has the smallest variance and the smallest linear coefficients, and Danish has the largest variance and the largest coefficients. The remaining three languages all lie somewhere in the middle, with Czech being closer to Arabic and Slovene closer to Danish. Together, the evidence from all five languages strongly corroborates the hypothesis that the expected running time for the system S_u is linear in sentence length for naturally occurring data.

4.2 Parsing Accuracy

In order to assess the parsing accuracy that can be achieved with the new transition system, we trained a deterministic parser using the new transition system S_u for each of the five languages. For comparison, we also trained two parsers using S_p, one that is strictly projective and one that uses the pseudo-projective parsing technique to recover non-projective dependencies in a post-processing step (Nivre and Nilsson, 2005). We will refer to the latter system as S_{pp}. All systems use SVM classifiers with a polynomial kernel to approximate the oracle function, with features and parameters taken from Nivre et al. (2006), which was the best performing transition-based system in the CoNLL-X shared task.[2]

Table 1 shows the labeled parsing accuracy of the parsers measured in two ways: *attachment score* (AS) is the percentage of *tokens* with the correct head and dependency label; *exact match* (EM) is the percentage of *sentences* with a completely correct labeled dependency tree. The score in brackets is the attachment score for the (small) subset of tokens that are connected to their head by a non-projective arc in the gold standard parse. For comparison, the table also includes results for the two best performing systems in the original CoNLL-X shared task, Malt-06 (Nivre et al., 2006) and MST-06 (McDonald et al., 2006), as well as the integrated system MST_{Malt}, which is a graph-based parser guided by the predictions of a transition-based parser and currently has the best reported results on the CoNLL-X data sets (Nivre and McDonald, 2008).

Looking first at the overall attachment score, we see that S_u gives a substantial improvement over S_p (and outperforms S_{pp}) for Czech and Slovene, where the scores achieved are rivaled only by the combo system MST_{Malt}. For these languages, there is no statistical difference between S_u and MST_{Malt}, which are both significantly better than all the other parsers, except S_{pp} for Czech (McNemar's test, $\alpha = .05$). This is accompanied by an improvement on non-projective arcs, where

[2]Complete information about experimental settings can be found at http://stp.lingfil.uu.se/∼nivre/exp/.

S_u outperforms all other systems for Czech and is second only to the two MST parsers (MST-06 and MST_{Malt}) for Slovene. It is worth noting that the percentage of non-projective arcs is higher for Czech (1.9%) and Slovene (1.9%) than for any of the other languages.

For the other three languages, S_u has a drop in overall attachment score compared to S_p, but none of these differences is statistically significant. In fact, the only significant differences in attachment score here are the positive differences between MST_{Malt} and all other systems for Arabic and Danish, and the negative difference between MST-06 and all other systems for Turkish. The attachment scores for non-projective arcs are generally very low for these languages, except for the two MST parsers on Danish, but S_u performs at least as well as S_{pp} on Danish and Turkish. (The results for Arabic are not very meaningful, given that there are only eleven non-projective arcs in the entire test set, of which the (pseudo-)projective parsers found two and S_u one, while MST_{Malt} and MST-06 found none at all.)

Considering the exact match scores, finally, it is very interesting to see that S_u almost consistently outperforms all other parsers, including the combo system MST_{Malt}, and sometimes by a fairly wide margin (Czech, Slovene). The difference is statistically significant with respect to all other systems except MST_{Malt} for Slovene, all except MST_{Malt} and S_{pp} for Czech, and with respect to MST_{Malt} for Turkish. For Arabic and Danish, there are no significant differences in the exact match scores. We conclude that S_u may increase the probability of finding a completely correct analysis, which is sometimes reflected also in the overall attachment score, and we conjecture that the strength of the positive effect is dependent on the frequency of non-projective arcs in the language.

5 Related Work

Processing non-projective trees by swapping the order of words has recently been proposed by both Nivre (2008b) and Titov et al. (2009), but these systems cannot handle unrestricted non-projective trees. It is worth pointing out that, although the system described in Nivre (2008b) uses four transitions bearing the same names as the transitions of S_u, the two systems are not equivalent. In particular, the system of Nivre (2008b) is sound but not complete for the class of all dependency trees.

There are also affinities to the system of Attardi (2006), which combines non-adjacent nodes on the stack instead of swapping nodes and is equivalent to a restricted version of our system, where no more than two consecutive SWAP transitions are permitted. This restriction preserves linear worst-case complexity at the expense of completeness. Finally, the algorithm first described by Covington (2001) and used for data-driven parsing by Nivre (2007), is complete but has quadratic complexity even in the best case.

6 Conclusion

We have presented a novel transition system for dependency parsing that can handle unrestricted non-projective trees. The system reuses standard techniques for building projective trees by combining adjacent nodes (representing subtrees with adjacent yields), but adds a simple mechanism for swapping the order of nodes on the stack, which gives a system that is sound and complete for the set of all dependency trees over a given label set but behaves exactly like the standard system for the subset of projective trees. As a result, the time complexity of deterministic parsing is $O(n^2)$ in the worst case, which is rare, but $O(n)$ in the best case, which is common, and experimental results on data from five languages support the conclusion that expected running time is linear in the length of the sentence. Experimental results also show that parsing accuracy is competitive, especially for languages like Czech and Slovene where non-projective dependency structures are common, and especially with respect to the exact match score, where it has the best reported results for four out of five languages. Finally, the simplicity of the system makes it very easy to implement.

Future research will include an in-depth error analysis to find out why the system works better for some languages than others and why the exact match score improves even when the attachment score goes down. In addition, we want to explore alternative oracle functions, which try to minimize the number of swaps by allowing the stack to be temporarily "unsorted".

Acknowledgments

Thanks to Johan Hall and Jens Nilsson for help with implementation and evaluation, and to Marco Kuhlmann and three anonymous reviewers for useful comments.

References

Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of CoNLL*, pages 166–170.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*, pages 149–164.

Michael A. Covington. 2001. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102.

Carlos Gómez-Rodríguez, David Weir, and John Carroll. 2009. Parsing mildly non-projective dependency structures. In *Proceedings of EACL*, pages 291–299.

Keith Hall and Vaclav Novák. 2005. Corrective modeling for non-projective dependency parsing. In *Proceedings of IWPT*, pages 42–52.

Jiri Havelka. 2007. Beyond projectivity: Multilingual evaluation of constraints and measures on non-projective structures. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 608–615.

Richard Johansson and Pierre Nugues. 2007. Incremental dependency parsing using online learning. In *Proceedings of the Shared Task of EMNLP-CoNLL*, pages 1134–1138.

Marco Kuhlmann and Joakim Nivre. 2006. Mildly non-projective dependency structures. In *Proceedings of the COLING/ACL Main Conference Poster Sessions*, pages 507–514.

Marco Kuhlmann and Giorgio Satta. 2009. Treebank grammar techniques for non-projective dependency parsing. In *Proceedings of EACL*, pages 478–486.

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

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of IWPT*, pages 122–131.

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

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP*, pages 523–530.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of CoNLL*, pages 216–220.

Peter Neuhaus and Norbert Bröker. 1997. The complexity of recognition of linguistically adequate dependency grammars. In *Proceedings of ACL/EACL*, pages 337–343.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL*, pages 950–958.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of ACL*, pages 99–106.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of CoNLL*, pages 49–56.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülsen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of CoNLL*, pages 221–225.

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

Joakim Nivre. 2006. Constraints on non-projective dependency graphs. In *Proceedings of EACL*, pages 73–80.

Joakim Nivre. 2007. Incremental non-projective dependency parsing. In *Proceedings of NAACL HLT*, pages 396–403.

Joakim Nivre. 2008a. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34:513–553.

Joakim Nivre. 2008b. Sorting out dependency parsing. In *Proceedings of the 6th International Conference on Natural Language Processing (GoTAL)*, pages 16–27.

Ivan Titov and James Henderson. 2007. A latent variable model for generative dependency parsing. In *Proceedings of IWPT*, pages 144–155.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarization for synchronous parsing of semantic and syntactic dependencies. In *Proceedings of IJCAI*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, pages 195–206.

Semi-supervised Learning of Dependency Parsers using Generalized Expectation Criteria

Gregory Druck
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
gdruck@cs.umass.edu

Gideon Mann
Google, Inc.
76 9th Ave.
New York, NY 10011
gideon.mann@gmail.com

Andrew McCallum
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
mccallum@cs.umass.edu

Abstract

In this paper, we propose a novel method for semi-supervised learning of non-projective log-linear dependency parsers using directly expressed linguistic prior knowledge (e.g. a *noun*'s parent is often a *verb*). Model parameters are estimated using a *generalized expectation* (GE) objective function that penalizes the mismatch between model predictions and linguistic expectation constraints. In a comparison with two prominent "unsupervised" learning methods that require indirect biasing toward the correct syntactic structure, we show that GE can attain better accuracy with as few as 20 intuitive constraints. We also present positive experimental results on longer sentences in multiple languages.

1 Introduction

Early approaches to parsing assumed a grammar provided by human experts (Quirk et al., 1985). Later approaches avoided grammar writing by learning the grammar from sentences explicitly annotated with their syntactic structure (Black et al., 1992). While such supervised approaches have yielded accurate parsers (Charniak, 2001), the syntactic annotation of corpora such as the Penn Treebank is extremely costly, and consequently there are few treebanks of comparable size.

As a result, there has been recent interest in unsupervised parsing. However, in order to attain reasonable accuracy, these methods have to be carefully biased towards the desired syntactic structure. This weak supervision has been encoded using priors and initializations (Klein and Manning, 2004; Smith, 2006), specialized models (Klein and Manning, 2004; Seginer, 2007; Bod, 2006), and implicit negative evidence (Smith, 2006). These indirect methods for leveraging prior knowledge can be cumbersome and unintuitive for a non-machine-learning expert.

This paper proposes a method for directly guiding the learning of dependency parsers with naturally encoded linguistic insights. *Generalized expectation* (GE) (Mann and McCallum, 2008; Druck et al., 2008) is a recently proposed framework for incorporating prior knowledge into the learning of conditional random fields (CRFs) (Lafferty et al., 2001). GE criteria express a preference on the value of a model expectation. For example, we know that "in English, when a *determiner* is directly to the left of a *noun*, the *noun* is usually the parent of the *determiner*". With GE we may add a term to the objective function that encourages a feature-rich CRF to match this expectation on unlabeled data, and in the process learn about related features. In this paper we use a non-projective dependency tree CRF (Smith and Smith, 2007).

While a complete exploration of linguistic prior knowledge for dependency parsing is beyond the scope of this paper, we provide several promising demonstrations of the proposed method. On the English WSJ10 data set, GE training outperforms two prominent unsupervised methods using only 20 constraints either elicited from a human or provided by an "oracle" simulating a human. We also present experiments on longer sentences in Dutch, Spanish, and Turkish in which we obtain accuracy comparable to supervised learning with tens to hundreds of complete parsed sentences.

2 Related Work

This work is closely related to the *prototype-driven* grammar induction method of Haghighi and Klein (2006), which uses prototype phrases to guide the EM algorithm in learning a PCFG. Direct comparison with this method is not possible because we are interested in dependency syntax rather than phrase structure syntax. However, the approach we advocate has several significant

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 360–368,
Suntec, Singapore, 2-7 August 2009.

advantages. GE is more general than prototype-driven learning because GE constraints can be uncertain. Additionally prototype-driven grammar induction needs to be used in conjunction with other unsupervised methods (distributional similarity and CCM (Klein and Manning, 2004)) to attain reasonable accuracy, and is only evaluated on length 10 or less sentences with no lexical information. In contrast, GE uses only the provided constraints and unparsed sentences, and is used to train a feature-rich discriminative model.

Conventional semi-supervised learning requires parsed sentences. Kate and Mooney (2007) and McClosky et al. (2006) both use modified forms of self-training to bootstrap parsers from limited labeled data. Wang et al. (2008) combine a structured loss on parsed sentences with a least squares loss on unlabeled sentences. Koo et al. (2008) use a large unlabeled corpus to estimate cluster features which help the parser generalize with fewer examples. Smith and Eisner (2007) apply entropy regularization to dependency parsing. The above methods can be applied to small seed corpora, but McDonald[1] has criticized such methods as working from an unrealistic premise, as a significant amount of the effort required to build a treebank comes in the first 100 sentences (both because of the time it takes to create an appropriate rubric and to train annotators).

There are also a number of methods for unsupervised learning of dependency parsers. Klein and Manning (2004) use a carefully initialized and structured generative model (DMV) in conjunction with the EM algorithm to get the first positive results on unsupervised dependency parsing. As empirical evidence of the sensitivity of DMV to initialization, Smith (2006) (pg. 37) uses three different initializations, and only one, the method of Klein and Manning (2004), gives accuracy higher than 31% on the WSJ10 corpus (see Section 5). This initialization encodes the prior knowledge that long distance attachments are unlikely.

Smith and Eisner (2005) develop *contrastive estimation* (CE), in which the model is encouraged to move probability mass away from implicit negative examples defined using a carefully chosen neighborhood function. For instance, Smith (2006) (pg. 82) uses eight different neighborhood functions to estimate parameters for the DMV model. The best performing neighborhood function DEL1ORTRANS1 provides accuracy of 57.6% on WSJ10 (see Section 5). Another neighborhood, DEL1ORTRANS2, provides accuracy of 51.2%. The remaining six neighborhood functions provide accuracy below 50%. This demonstrates that constructing an appropriate neighborhood function can be delicate and challenging.

Smith and Eisner (2006) propose *structural annealing* (SA), in which a strong bias for local dependency attachments is enforced early in learning, and then gradually relaxed. This method is sensitive to the annealing schedule. Smith (2006) (pg. 136) use 10 annealing schedules in conjunction with three initializers. The best performing combination attains accuracy of 66.7% on WSJ10, but the worst attains accuracy of 32.5%.

Finally, Seginer (2007) and Bod (2006) approach unsupervised parsing by constructing novel syntactic models. The development and tuning of the above methods constitute the encoding of prior domain knowledge about the desired syntactic structure. In contrast, our framework provides a straightforward and explicit method for incorporating prior knowledge.

Ganchev et al. (2009) propose a related method that uses posterior constrained EM to learn a projective target language parser using only a source language parser and word alignments.

3 Generalized Expectation Criteria

Generalized expectation criteria (Mann and McCallum, 2008; Druck et al., 2008) are terms in a parameter estimation objective function that express a preference on the value of a model expectation. Let $\mathbf{x}$ represent input variables (i.e. a sentence) and $\mathbf{y}$ represent output variables (i.e. a parse tree). A generalized expectation term $\mathcal{G}(\lambda)$ is defined by a constraint function $G(\mathbf{y}, \mathbf{x})$ that returns a non-negative real value given input and output variables, an empirical distribution $\tilde{p}(\mathbf{x})$ over input variables (i.e. unlabeled data), a model distribution $p_\lambda(\mathbf{y}|\mathbf{x})$, and a score function S:

$$\mathcal{G}(\lambda) = S(E_{\tilde{p}(\mathbf{x})}[E_{p_\lambda(\mathbf{y}|\mathbf{x})}[G(\mathbf{y}, \mathbf{x})]]).$$

In this paper, we use a score function that is the squared difference of the model expectation of G and some target expectation $\tilde{G}$:

$$S_{sq} = -(\tilde{G} - E_{\tilde{p}(\mathbf{x})}[E_{p_\lambda(\mathbf{y}|\mathbf{x})}[G(\mathbf{y}, \mathbf{x})]])^2 \quad (1)$$

We can incorporate prior knowledge into the training of $p_\lambda(\mathbf{y}|\mathbf{x})$ by specifying the from of the constraint function G and the target expectation $\tilde{G}$.

[1] R. McDonald, personal communication, 2007

Importantly, G does not need to match a particular feature in the underlying model.

The complete objective function[2] includes multiple GE terms and a prior on parameters[3], $p(\lambda)$

$$\mathcal{O}(\lambda; \mathcal{D}) = p(\lambda) + \sum_{\mathcal{G}} \mathcal{G}(\lambda)$$

GE has been applied to logistic regression models (Mann and McCallum, 2007; Druck et al., 2008) and linear chain CRFs (Mann and McCallum, 2008). In the following sections we apply GE to non-projective CRF dependency parsing.

3.1 GE in General CRFs

We first consider an arbitrarily structured conditional random field (Lafferty et al., 2001) $p_\lambda(\mathbf{y}|\mathbf{x})$. We describe the CRF for non-projective dependency parsing in Section 3.2. The probability of an output $\mathbf{y}$ conditioned on an input $\mathbf{x}$ is

$$p_\lambda(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_\mathbf{x}} \exp\Big(\sum_j \lambda_j F_j(\mathbf{y}, \mathbf{x})\Big),$$

where F_j are feature functions over the cliques of the graphical model and $Z(\mathbf{x})$ is a normalizing constant that ensures $p_\lambda(\mathbf{y}|\mathbf{x})$ sums to 1. We are interested in the expectation of constraint function $G(\mathbf{x}, \mathbf{y})$ under this model. We abbreviate this *model expectation* as:

$$G_\lambda = E_{\tilde{p}(\mathbf{x})}[E_{p_\lambda(\mathbf{y}|\mathbf{x})}[G(\mathbf{y}, \mathbf{x})]]$$

It can be shown that partial derivative of $\mathcal{G}(\lambda)$ using S_{sq}[4] with respect to model parameter λ_j is

$$\frac{\partial}{\partial \lambda_j}\mathcal{G}(\lambda) = 2(\tilde{G} - G_\lambda) \quad (2)$$
$$\Big(E_{\tilde{p}(\mathbf{x})}\Big[E_{p_\lambda(\mathbf{y}|\mathbf{x})}\left[G(\mathbf{y}, \mathbf{x})F_j(\mathbf{y}, \mathbf{x})\right] - E_{p_\lambda(\mathbf{y}|\mathbf{x})}\left[G(\mathbf{y}, \mathbf{x})\right] E_{p_\lambda(\mathbf{y}|\mathbf{x})}\left[F_j(\mathbf{y}, \mathbf{x})\right]\Big]\Big).$$

Equation 2 has an intuitive interpretation. The first term (on the first line) is the difference between the model and target expectations. The second term (the rest of the equation) is the predicted covariance between the constraint function G and the model feature function F_j. Therefore, if the constraint is not satisfied, GE updates parameters for features that the model predicts are related to the constraint function.

If there are constraint functions G for all model feature functions F_j, and the target expectations $\tilde{G}$ are estimated from labeled data, then the globally optimal parameter setting under the GE objective function is equivalent to the maximum likelihood solution. However, GE does not require such a one-to-one correspondence between constraint functions and model feature functions. This allows bootstrapping of feature-rich models with a small number of prior expectation constraints.

3.2 Non-Projective Dependency Tree CRFs

We now define a CRF $p_\lambda(\mathbf{y}|\mathbf{x})$ for unlabeled, non-projective[5] dependency parsing. The tree $\mathbf{y}$ is represented as a vector of the same length as the sentence, where y_i is the index of the parent of word i. The probability of a tree $\mathbf{y}$ given sentence $\mathbf{x}$ is

$$p_\lambda(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_\mathbf{x}} \exp\Big(\sum_{i=1}^{n}\sum_j \lambda_j f_j(x_i, x_{y_i}, \mathbf{x})\Big),$$

where f_j are *edge-factored* feature functions that consider the child input (word, tag, or other feature), the parent input, and the rest of the sentence. This factorization implies that dependency decisions are independent conditioned on the input sentence $\mathbf{x}$ if $\mathbf{y}$ is a tree. Computing $Z_\mathbf{x}$ and the edge expectations needed for partial derivatives requires summing over all possible trees for $\mathbf{x}$.

By relating the sum of the scores of all possible trees to counting the number of spanning trees in a graph, it can be shown that $Z_\mathbf{x}$ is the determinant of the *Kirchoff matrix* K, which is constructed using the scores of possible edges. (McDonald and Satta, 2007; Smith and Smith, 2007). Computing the determinant takes $O(n^3)$ time, where n is the length of the sentence. To compute the marginal probability of a particular edge $k \rightarrow i$ (i.e. $y_i = k$), the score of any edge $k' \rightarrow i$ such that $k' \neq k$ is set to 0. The determinant of the resulting modified Kirchoff matrix $K_{k \rightarrow i}$ is then the sum of the scores of all trees that include the edge $k \rightarrow i$. The

[2] In general, the objective function could also include the likelihood of available labeled data, but throughout this paper we assume we have no parsed sentences.

[3] Throughout this paper we use a Gaussian prior on parameters with $\sigma^2 = 10$.

[4] In previous work, S was the KL-divergence from the target expectation. The partial derivative of the KL divergence score function includes the same covariance term as above but substitutes a different multiplicative term: $\tilde{G}/G_\lambda$.

[5] Note that we could instead define a CRF for projective dependency parse trees and use a variant of the inside outside algorithm for inference. We choose non-projective because it is the more general case.

marginal $p(y_i=k|\mathbf{x};\theta)$ can be computed by dividing this score by $Z_{\mathbf{x}}$ (McDonald and Satta, 2007). Computing all edge expectations with this algorithm takes $O(n^5)$ time. Smith and Smith (2007) describe a more efficient algorithm that can compute all edge expectations in $O(n^3)$ time using the inverse of the Kirchoff matrix K^{-1}.

3.3 GE for Non-Projective Dependency Tree CRFs

While in general constraint functions G may consider multiple edges, in this paper we use edge-factored constraint functions. In this case $E_{p_\lambda(\mathbf{y}|\mathbf{x})}[G(\mathbf{y},\mathbf{x})]E_{p_\lambda(\mathbf{y}|\mathbf{x})}[F_j(\mathbf{y},\mathbf{x})]$, the second term of the covariance in Equation 2, can be computed using the edge marginal distributions $p_\lambda(y_i|\mathbf{x})$. The first term of the covariance $E_{p_\lambda(\mathbf{y}|\mathbf{x})}[G(\mathbf{y},\mathbf{x})F_j(\mathbf{y},\mathbf{x})]$ is more difficult to compute because it requires the marginal probability of two edges $p_\lambda(y_i, y_{i'}|\mathbf{x})$. It is important to note that the model p_λ is still edge-factored.

The sum of the scores of all trees that contain edges $k \rightarrow i$ and $k' \rightarrow i'$ can be computed by setting the scores of edges $j \rightarrow i$ such that $j \neq k$ and $j' \rightarrow i'$ such that $j' \neq k'$ to 0, and computing the determinant of the resulting modified Kirchoff matrix $K_{k\rightarrow i,k'\rightarrow i'}$. There are $O(n^4)$ pairs of possible edges, and the determinant computation takes time $O(n^3)$, so this naive algorithm takes $O(n^7)$ time.

An improved algorithm computes, for each possible edge $k \rightarrow i$, a modified Kirchoff matrix $K_{k\rightarrow i}$ that requires the presence of that edge. Then, the method of Smith and Smith (2007) can be used to compute the probability of every possible edge conditioned on the presence of $k \rightarrow i$, $p_\lambda(y_{i'}=k'|y_i=k,\mathbf{x})$, using $K^{-1}_{k\rightarrow i}$. Multiplying this probability by $p_\lambda(y_i{=}k|\mathbf{x})$ yields the desired two edge marginal. Because this algorithm pulls the $O(n^3)$ matrix operation out of the inner loop over edges, the run time is reduced to $O(n^5)$.

If it were possible to perform only one $O(n^3)$ matrix operation per sentence, then the gradient computation would take only $O(n^4)$ time, the time required to consider all pairs of edges. Unfortunately, there is no straightforward generalization of the method of Smith and Smith (2007) to the two edge marginal problem. Specifically, *Laplace expansion* generalizes to second-order matrix minors, but it is not clear how to compute second-order cofactors from the inverse Kirchoff matrix alone (c.f. (Smith and Smith, 2007)).

Consequently, we also propose an approximation that can be used to speed up GE training at the expense of a less accurate covariance computation. We consider different cases of the edges $k \rightarrow i$, and $k' \rightarrow i'$.

- $p_\lambda(y_i{=}k, y_{i'}{=}k'|\mathbf{x}){=}0$ when $i{=}i'$ and $k{\neq}k'$ (different parent for the same word), or when $i{=}k'$ and $k{=}i'$ (cycle), because these pairs of edges break the tree constraint.
- $p_\lambda(y_i{=}k, y_{i'}{=}k'|\mathbf{x}){=}p_\lambda(y_i{=}k|\mathbf{x})$ when $i{=}i', k{=}k'$.
- $p_\lambda(y_i=k, y_{i'}=k'|\mathbf{x}) \approx p_\lambda(y_i=k|\mathbf{x})p_\lambda(y_{i'}=k'|\mathbf{x})$ when $i \neq i'$ and $i \neq k'$ or $i' \neq k$ (different words, do not create a cycle). This approximation assumes that pairs of edges that do not fall into one of the above cases are conditionally independent given $\mathbf{x}$. This is not true because there are partial trees in which $k \rightarrow i$ and $k' \rightarrow i'$ can appear separately, but not together (for example if $i = k'$ and the partial tree contains $i' \rightarrow k$).

Using this approximation, the covariance for one sentence is approximately equal to

$$\sum_i^n E_{p_\lambda(y_i|\mathbf{x})}[f_j(x_i, x_{y_i}, \mathbf{x})g(x_i, x_{y_i}, \mathbf{x})]$$
$$-\sum_i^n E_{p_\lambda(y_i|\mathbf{x})}[f_j(x_i, x_{y_i}, \mathbf{x})]E_{p_\lambda(y_i|\mathbf{x})}[g(x_i, x_{y_i}, \mathbf{x})]$$
$$-\sum_{i,k}^n p_\lambda(y_i{=}k|\mathbf{x})p_\lambda(y_k{=}i|\mathbf{x})f_j(x_i, x_k, \mathbf{x})g(x_k, x_i, \mathbf{x}).$$

Intuitively, the first and second terms compute a covariance over possible parents for a single word, and the third term accounts for cycles. Computing the above takes $O(n^3)$ time, the time required to compute single edge marginals. In this paper, we use the $O(n^5)$ exact method, though we find that the accuracy attained by approximate training is usually within 5% of the exact method.

If G is not edge-factored, then we need to compute a marginal over three or more edges, making exact training intractable. An appealing alternative to a similar approximation to the above would use loopy belief propagation to efficiently approximate the marginals (Smith and Eisner, 2008).

In this paper g is binary and normalized by its total count in the corpus. The expectation of g is then the probability that it indicates a true edge.

4 Linguistic Prior Knowledge

Training parsers using GE with the aid of linguists is an exciting direction for future work. In this paper, we use constraints derived from several basic types of linguistic knowledge.

One simple form of linguistic knowledge is the set of possible parent tags for a given child tag. This type of constraint was used in the development of a rule-based dependency parser (Debusmann et al., 2004). Additional information can be obtained from small grammar fragments. Haghighi and Klein (2006) provide a list of prototype phrase structure rules that can be augmented with dependencies and used to define constraints involving parent and child tags, surrounding or interposing tags, direction, and distance. Finally there are well known hypotheses about the direction and distance of attachments that can be used to define constraints. Eisner and Smith (2005) use the fact that short attachments are more common to improve unsupervised parsing accuracy.

4.1 "Oracle" constraints

For some experiments that follow we use "oracle" constraints that are estimated from labeled data. This involves choosing feature templates (motivated by the linguistic knowledge described above) and estimating target expectations. Oracle methods used in this paper consider three simple statistics of candidate constraint functions: count $\tilde{c}(g)$, edge count $\tilde{c}_{edge}(g)$, and edge probability $\tilde{p}(edge|g)$. Let $\mathcal{D}$ be the labeled corpus.

$$\tilde{c}(g) = \sum_{\mathbf{x} \in \mathcal{D}} \sum_{i} \sum_{j} g(x_i, x_j, \mathbf{x})$$

$$\tilde{c}_{edge}(g) = \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}} \sum_{i} g(x_i, x_{y_i}, \mathbf{x})$$

$$\tilde{p}(edge|g) = \frac{\tilde{c}_{edge}(g)}{\tilde{c}(g)}$$

Constraint functions are selected according to some combination of the above statistics. In some cases we additionally prune the candidate set by considering only certain templates. To compute the target expectation, we simply use $\mathrm{bin}(\tilde{p}(edge|g))$, where bin returns the closest value in the set $\{0, 0.1, 0.25, 0.5, 0.75, 1\}$. This can be viewed as specifying that g is *very indicative of edge*, *somewhat indicative of edge*, etc.

5 Experimental Comparison with Unsupervised Learning

In this section we compare GE training with methods for unsupervised parsing. We use the WSJ10 corpus (as processed by Smith (2006)), which is comprised of English sentences of ten words or fewer (after stripping punctuation) from the WSJ portion of the Penn Treebank. As in previous work sentences contain only part-of-speech tags.

We compare GE and supervised training of an edge-factored CRF with unsupervised learning of a DMV model (Klein and Manning, 2004) using EM and contrastive estimation (CE) (Smith and Eisner, 2005). We also report the accuracy of an attach-right baseline[6]. Finally, we report the accuracy of a constraint baseline that assigns a score to each possible edge that is the sum of the target expectations for all constraints on that edge. Possible edges without constraints receive a score of 0. These scores are used as input to the maximum spanning tree algorithm, which returns the best tree. Note that this is a strong baseline because it can handle uncertain constraints, and the tree constraint imposed by the MST algorithm helps information propagate across edges.

We note that there are considerable differences between the DMV and CRF models. The DMV model is more expressive than the CRF because it can model the arity of a head as well as sibling relationships. Because these features consider multiple edges, including them in the CRF model would make exact inference intractable (McDonald and Satta, 2007). However, the CRF may consider the distance between head and child, whereas DMV does not model distance. The CRF also models non-projective trees, which when evaluating on English is likely a disadvantage.

Consequently, we experiment with two sets of features for the CRF model. The first, *restricted* set includes features that consider the head and child tags of the dependency conjoined with the direction of the attachment, (*parent-POS,child-POS,direction*). With this feature set, the CRF model is less expressive than DMV. The second *full* set includes standard features for edge-factored dependency parsers (McDonald et al., 2005), though still unlexicalized. The CRF cannot consider valency even with the *full* feature set, but this is balanced by the ability to use distance.

[6]The reported accuracies with the DMV model and the attach-right baseline are taken from (Smith, 2006).

feature	ex.	feature	ex.
MD → VB	1.00	NNS ← **VBD**	0.75
POS ← **NN**	0.75	PRP ← **VBD**	0.75
JJ ← **NNS**	0.75	**VBD** → TO	1.00
NNP ← **POS**	0.75	**VBD** → VBN	0.75
ROOT → MD	0.75	NNS ← **VBP**	0.75
ROOT → VBD	1.00	PRP ← **VBP**	0.75
ROOT → VBP	0.75	**VBP** → VBN	0.75
ROOT → VBZ	0.75	PRP ← **VBZ**	0.75
TO → VB	1.00	NN ← **VBZ**	0.75
VBN → IN	0.75	**VBZ** → VBN	0.75

Table 1: 20 constraints that give 61.3% accuracy on WSJ10. Tags are grouped according to heads, and are in the order they appear in the sentence, with the arrow pointing from head to modifier.

We generate constraints in two ways. First, we use oracle constraints of the form (*parent-POS,child-POS,direction*) such that $\tilde{c}(g) \geq 200$. We choose constraints in descending order of $\tilde{p}(edge|g)$. The first 20 constraints selected using this method are displayed in Table 1.

Although the reader can verify that the constraints in Table 1 are reasonable, we additionally experiment with human-provided constraints. We use the prototype phrase-structure constraints provided by Haghighi and Klein (2006), and with the aid of head-finding rules, extract 14 (*parent-pos,child-pos,direction*) constraints.[7] We then estimated target expectations for these constraints using our prior knowledge, without looking at the training data. We also created a second constraint set with an additional six constraints for tag pairs that were previously underrepresented.

5.1 Results

We present results varying the number of constraints in Figures 1 and 2. Figure 1 compares supervised and GE training of the CRF model, as well as the feature constraint baseline. First we note that GE training using the *full* feature set substantially outperforms the *restricted* feature set, despite the fact that the same set of constraints is used for both experiments. This result demonstrates GE's ability to learn about related but non-constrained features. GE training also outperforms the baseline[8].

We compare GE training of the CRF model with unsupervised learning of the DMV model in Figure 2[9]. Despite the fact that the *restricted* CRF is less expressive than DMV, GE training of this model outperforms EM with 30 constraints and CE with 50 constraints. GE training of the *full* CRF outperforms EM with 10 constraints and CE with 20 constraints (those displayed in Table 1). GE training of the *full* CRF with the set of 14 constraints from (Haghighi and Klein, 2006), gives accuracy of 53.8%, which is above the interpolated oracle constraints curve (43.5% accuracy with 10 constraints, 61.3% accuracy with 20 constraints). With the 6 additional constraints, we obtain accuracy of 57.7% and match CE.

Recall that CE, EM, and the DMV model incorporate prior knowledge indirectly, and that the reported results are heavily-tuned ideal cases (see Section 2). In contrast, GE provides a method to directly encode intuitive linguistic insights.

Finally, note that structural annealing (Smith and Eisner, 2006) provides 66.7% accuracy on WSJ10 when choosing the best performing annealing schedule (Smith, 2006). As noted in Section 2 other annealing schedules provide accuracy as low as 32.5%. GE training of the *full* CRF attains accuracy of 67.0% with 30 constraints.

6 Experimental Comparison with Supervised Training on Long Sentences

Unsupervised parsing methods are typically evaluated on short sentences, as in Section 5. In this section we show that GE can be used to train parsers for longer sentences that provide comparable accuracy to supervised training with tens to hundreds of parsed sentences.

We use the standard train/test splits of the Spanish, Dutch, and Turkish data from the 2006 CoNLL Shared Task. We also use standard edge-factored feature templates (McDonald et al., 2005)[10]. We experiment with versions of the dat-

[7] Because the CFG rules in (Haghighi and Klein, 2006) are "flattened" and in some cases do not generate appropriate dependency constraints, we only used a subset.

[8] The baseline eventually matches the accuracy of the restricted CRF but this is understandable because GE's ability to bootstrap is greatly reduced with the restricted feature set.

[9] Klein and Manning (2004) report 43.2% accuracy for DMV with EM on WSJ10. When jointly modeling constituency and dependencies, Klein and Manning (2004) report accuracy of 47.5%. Seginer (2007) and Bod (2006) propose unsupervised phrase structure parsing methods that give better unlabeled F-scores than DMV with EM, but they do not report directed dependency accuracy.

[10] Typical feature processing uses only *supported* features, or those features that occur on at least one true edge in the training data. Because we assume that the data is unlabeled, we instead use features on all possible edges. This generates tens of millions features, so we prune those features that occur fewer than 10 total times, as in (Smith and Eisner, 2007).

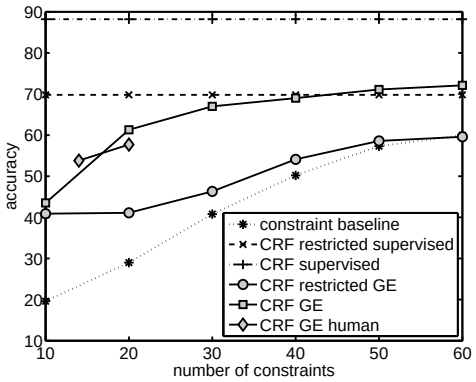


Figure 1: Comparison of the constraint baseline and both GE and supervised training of the *restricted* and *full* CRF. Note that supervised training uses 5,301 parsed sentences. GE with human provided constraints closely matches the oracle results.

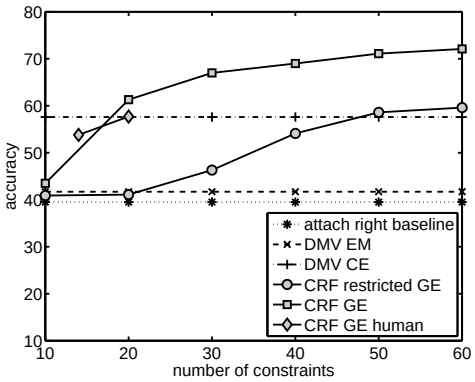


Figure 2: Comparison of GE training of the *restricted* and *full* CRFs with unsupervised learning of DMV. GE training of the *full* CRF outperforms CE with just 20 constraints. GE also matches CE with 20 human provided constraints.

sets in which we remove sentences that are longer than 20 words and 60 words.

For these experiments, we use an oracle constraint selection method motivated by the linguistic prior knowledge described in Section 4. The first set of constraints specify the most frequent head tag, attachment direction, and distance combinations for each child tag. Specifically, we select oracle constraints of the type (*parent-CPOS,child-CPOS,direction,distance*)[11]. We add constraints for every g such that $\tilde{c}_{edge}(g) > 100$ for max length 60 data sets, and $\tilde{c}_{edge}(g) > 10$ times for max length 20 data sets.

In some cases, the *possible parent* constraints described above will not be enough to provide high accuracy, because they do not consider other tags in the sentence (McDonald et al., 2005). Consequently, we experiment with adding an additional 25 *sequence* constraints (for what are often called "between" and "surrounding" features). The oracle feature selection method aims to choose such constraints that help to reduce uncertainty in the *possible parents* constraint set. Consequently, we consider sequence features g_s with $\tilde{p}(edge|g_s = 1) \geq 0.75$, and whose corresponding (*parent-CPOS,child-CPOS,direction,distance*) constraint g, has edge probability $\tilde{p}(edge|g) \leq 0.25$. Among these candidates, we sort by $\tilde{c}(g_s = 1)$, and select the top 25.

We compare with the constraint baseline described in Section 5. Additionally, we report the number of parsed sentences required for supervised CRF training (averaged over 5 random splits) to match the accuracy of GE training using the *possible parents* + *sequence* constraint set.

The results are provided in Table 2. We first observe that GE always beats the baseline, especially on parent decisions for which there are no constraints (not reported in Table 2, but for example 53.8% vs. 20.5% on Turkish 20). Second, we note that accuracy is always improved by adding *sequence* constraints. Importantly, we observe that GE gives comparable performance to supervised training with tens or hundreds of parsed sentences. These parsed sentences provide a tremendous amount of information to the model, as for example in 20 Spanish length ≤ 60 sentences, a total of 1,630,466 features are observed, 330,856 of them unique. In contrast, the constraint-based methods are provided at most a few hundred constraints. When comparing the human costs of parsing sentences and specifying constraints, remember that parsing sentences requires the development of detailed annotation guidelines, which can be extremely time-consuming (see also the discussion is Section 2).

Finally, we experiment with iteratively adding constraints. We sort constraints with $\tilde{c}(g) > 50$ by $\tilde{p}(edge|g)$, and ensure that 50% are (*parent-CPOS,child-CPOS,direction,distance*) constraints and 50% are *sequence* constraints. For lack of space, we only show the results for Spanish 60. In Figure 3, we see that GE beats the baseline more soundly than above, and that

[11] For these experiments we use coarse-grained part-of-speech tags in constraints.

	possible parent constraints		+ sequence constraints		complete trees
	baseline	GE	baseline	GE	
dutch 20	69.5	70.7	69.8	**71.8**	80-160
dutch 60	66.5	69.3	66.7	**69.8**	40-80
spanish 20	70.0	73.2	71.2	**75.8**	40-80
spanish 60	62.1	66.2	62.7	**66.9**	20-40
turkish 20	66.3	71.8	67.1	**72.9**	80-160
turkish 60	62.1	65.5	62.3	**66.6**	20-40

Table 2: Experiments on Dutch, Spanish, and Turkish with maximum sentence lengths of 20 and 60. Observe that GE outperforms the baseline, adding *sequence* constraints improves accuracy, and accuracy with GE training is comparable to supervised training with tens to hundreds of parsed sentences.

parent tag	true	predicted
det.	0.005	0.005
adv.	0.018	0.013
conj.	0.012	0.001
pron.	0.011	0.009
verb	0.355	0.405
adj.	0.067	0.075
punc.	0.031	0.013
noun	0.276	0.272
prep.	0.181	0.165

direction	true	predicted
right	0.621	0.598
left	0.339	0.362
distance	true	predicted
1	0.495	0.564
2	0.194	0.206
3	0.066	0.050
4	0.042	0.037
5	0.028	0.031
6-10	0.069	0.033
> 10	0.066	0.039

feature (distance)	false pos. occ.
verb $\rightarrow$ punc. ($>$10)	1183
noun $\rightarrow$ prep. (1)	1139
adj. $\rightarrow$ prep. (1)	855
verb $\rightarrow$ verb (6-10)	756
verb $\rightarrow$ verb ($>$10)	569
noun $\leftarrow$ punc. (1)	512
verb $\leftarrow$ punc. (2)	509
prep. $\leftarrow$ punc. (1)	476
verb $\rightarrow$ punc. (4)	427
verb $\rightarrow$ prep. (1)	422

Table 3: Error analysis for GE training with *possible parent* + *sequence* constraints on Spanish 60 data. On the left, the predicted and true distribution over parent coarse part-of-speech tags. In the middle, the predicted and true distributions over attachment directions and distances. On the right, common features on false positive edges.

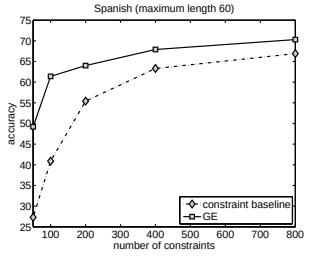


Figure 3: Comparing GE training of a CRF and constraint baseline while increasing the number of oracle constraints.

adding constraints continues to increase accuracy.

7 Error Analysis

In this section, we analyze the errors of the model learned with the *possible parent* + *sequence* constraints on the Spanish 60 data. In Table 3, we present four types of analysis. First, we present the predicted and true distributions over coarse-grained parent part of speech tags. We can see that verb is being predicted as a parent tag more often then it should be, while most other tags are predicted less often than they should be. Next, we show the predicted and true distributions over attachment direction and distance. From this we see that the model is often incorrectly predicting left attachments, and is predicting too many short attachments. Finally, we show the most common parent-child tag with direction and distance features that occur on false positive edges. From this table, we see that many errors concern the attachments of punctuation. The second line indicates a prepositional phrase attachment ambiguity.

This analysis could also be performed by a linguist by looking at predicted trees for selected sentences. Once errors are identified, GE constraints could be added to address these problems.

8 Conclusions

In this paper, we developed a novel method for the semi-supervised learning of a non-projective CRF dependency parser that directly uses linguistic prior knowledge as a training signal. It is our hope that this method will permit more effective leveraging of linguistic insight and resources and enable the construction of parsers in languages and domains where treebanks are not available.

Acknowledgments

We thank Ryan McDonald, Keith Hall, John Hale, Xiaoyun Wu, and David Smith for helpful discussions. This work was completed in part while Gregory Druck was an intern at Google. This work was supported in part by the Center for Intelligent Information Retrieval, The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0326249, and by the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-07-D-0185/0004. Any opinions, findings and conclusions or recommendations expressed in this material are the author's and do not necessarily reflect those of the sponsor.

References

E. Black, J. Lafferty, and S. Roukos. 1992. Development and evaluation of a broad-coverage probabilistic grammar of english language computer manuals. In *ACL*, pages 185–192.

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *ACL*, pages 865–872.

E. Charniak. 2001. Immediate-head parsing for language models. In *ACL*.

R. Debusmann, D. Duchier, A. Koller, M. Kuhlmann, G. Smolka, and S. Thater. 2004. A relational syntax-semantics interface based on dependency grammar. In *COLING*.

G. Druck, G. S. Mann, and A. McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *SIGIR*.

J. Eisner and N.A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *IWPT*.

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

A. Haghighi and D. Klein. 2006. Prototype-driven grammar induction. In *COLING*.

R. J. Kate and R. J. Mooney. 2007. Semi-supervised learning for semantic parsing using support vector machines. In *HLT-NAACL (Short Papers)*.

D. Klein and C. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL*.

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

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

G. Mann and A. McCallum. 2007. Simple, robust, scalable semi-supervised learning via expectation regularization. In *ICML*.

G. Mann and A. McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *ACL*.

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

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proc. of IWPT*, pages 121–132.

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

R. Quirk, S. Greenbaum, G. Leech, and J. Svartvik. 1985. *A Comprehensive Grammar of the English Language*. Longman.

Yoav Seginer. 2007. Fast unsupervised incremental parsing. In *ACL*, pages 384–391, Prague, Czech Republic.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: training log-linear models on unlabeled data. In *ACL*, pages 354–362.

Noah A. Smith and Jason Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *COLING-ACL*, pages 569–576.

David A. Smith and Jason Eisner. 2007. Bootstrapping feature-rich dependency parsers with entropic priors. In *EMNLP-CoNLL*, pages 667–677.

David A. Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *EMNLP*.

David A. Smith and Noah A. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *EMNLP-CoNLL*, pages 132–140.

Noah A. Smith. 2006. *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. Ph.D. thesis, Johns Hopkins University.

Qin Iris Wang, Dale Schuurmans, and Dekang Lin. 2008. Semi-supervised convex training for dependency parsing. In *ACL*, pages 532–540.

Dependency Grammar Induction via Bitext Projection Constraints

Kuzman Ganchev and **Jennifer Gillenwater** and **Ben Taskar**
Department of Computer and Information Science
University of Pennsylvania, Philadelphia PA, USA
{kuzman,jengi,taskar}@seas.upenn.edu

Abstract

Broad-coverage annotated treebanks necessary to train parsers do not exist for many resource-poor languages. The wide availability of parallel text and accurate parsers in English has opened up the possibility of grammar induction through partial transfer across bitext. We consider generative and discriminative models for dependency grammar induction that use word-level alignments and a source language parser (English) to constrain the space of possible target trees. Unlike previous approaches, our framework does not require full projected parses, allowing partial, approximate transfer through linear expectation constraints on the space of distributions over trees. We consider several types of constraints that range from generic dependency conservation to language-specific annotation rules for auxiliary verb analysis. We evaluate our approach on Bulgarian and Spanish CoNLL shared task data and show that we consistently outperform unsupervised methods and can outperform supervised learning for limited training data.

1 Introduction

For English and a handful of other languages, there are large, well-annotated corpora with a variety of linguistic information ranging from named entity to discourse structure. Unfortunately, for the vast majority of languages very few linguistic resources are available. This situation is likely to persist because of the expense of creating annotated corpora that require linguistic expertise (Abeillé, 2003). On the other hand, parallel corpora between many resource-poor languages and resource-rich languages are ample, motivating recent interest in transferring linguistic resources from one language to another via parallel text. For example, several early works (Yarowsky and Ngai, 2001; Yarowsky et al., 2001; Merlo et al., 2002) demonstrate transfer of shallow processing tools such as part-of-speech taggers and noun-phrase chunkers by using word-level alignment models (Brown et al., 1994; Och and Ney, 2000).

Alshawi et al. (2000) and Hwa et al. (2005) explore transfer of deeper syntactic structure: dependency grammars. Dependency and constituency grammar formalisms have long coexisted and competed in linguistics, especially beyond English (Mel'čuk, 1988). Recently, dependency parsing has gained popularity as a simpler, computationally more efficient alternative to constituency parsing and has spurred several supervised learning approaches (Eisner, 1996; Yamada and Matsumoto, 2003a; Nivre and Nilsson, 2005; McDonald et al., 2005) as well as unsupervised induction (Klein and Manning, 2004; Smith and Eisner, 2006). Dependency representation has been used for language modeling, textual entailment and machine translation (Haghighi et al., 2005; Chelba et al., 1997; Quirk et al., 2005; Shen et al., 2008), to name a few tasks.

Dependency grammars are arguably more robust to transfer since syntactic relations between aligned words of parallel sentences are better conserved in translation than phrase structure (Fox, 2002; Hwa et al., 2005). Nevertheless, several challenges to accurate training and evaluation from aligned bitext remain: (1) partial word alignment due to non-literal or distant translation; (2) errors in word alignments and source language parses, (3) grammatical annotation choices that differ across languages and linguistic theories (e.g., how to analyze auxiliary verbs, conjunctions).

In this paper, we present a flexible learning

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 369–377,
Suntec, Singapore, 2-7 August 2009.

framework for transferring dependency grammars via bitext using the posterior regularization framework (Graça et al., 2008). In particular, we address challenges (1) and (2) by avoiding commitment to an entire projected parse tree in the target language during training. Instead, we explore formulations of both generative and discriminative probabilistic models where projected syntactic relations are constrained to hold approximately and only in expectation. Finally, we address challenge (3) by introducing a very small number of language-specific constraints that disambiguate arbitrary annotation choices.

We evaluate our approach by transferring from an English parser trained on the Penn treebank to Bulgarian and Spanish. We evaluate our results on the Bulgarian and Spanish corpora from the CoNLL X shared task. We see that our transfer approach consistently outperforms unsupervised methods and, given just a few (2 to 7) language-specific constraints, performs comparably to a supervised parser trained on a very limited corpus (30 - 140 training sentences).

2 Approach

At a high level our approach is illustrated in Figure 1(a). A parallel corpus is word-level aligned using an alignment toolkit (Graça et al., 2009) and the source (English) is parsed using a dependency parser (McDonald et al., 2005). Figure 1(b) shows an aligned sentence pair example where dependencies are perfectly conserved across the alignment. An edge from English parent p to child c is called conserved if word p aligns to word p' in the second language, c aligns to c' in the second language, and p' is the parent of c'. Note that we are not restricting ourselves to one-to-one alignments here; p, c, p', and c' can all also align to other words. After filtering to identify well-behaved sentences and high confidence projected dependencies, we learn a probabilistic parsing model using the posterior regularization framework (Graça et al., 2008). We estimate both generative and discriminative models by constraining the posterior distribution over possible target parses to approximately respect projected dependencies and other rules which we describe below. In our experiments we evaluate the learned models on dependency treebanks (Nivre et al., 2007).

Unfortunately the sentence in Figure 1(b) is highly unusual in its amount of dependency conservation. To get a feel for the typical case, we used off-the-shelf parsers (McDonald et al., 2005) for English, Spanish and Bulgarian on two bitexts (Koehn, 2005; Tiedemann, 2007) and compared several measures of dependency conservation. For the English-Bulgarian corpus, we observed that 71.9% of the edges we projected were edges in the corpus, and we projected on average 2.7 edges per sentence (out of 5.3 tokens on average). For Spanish, we saw conservation of 64.4% and an average of 5.9 projected edges per sentence (out of 11.5 tokens on average).

As these numbers illustrate, directly transferring information one dependency edge at a time is unfortunately error prone for two reasons. First, parser and word alignment errors cause much of the transferred information to be wrong. We deal with this problem by constraining groups of edges rather than a single edge. For example, in some sentence pair we might find 10 edges that have both end points aligned and can be transferred. Rather than requiring our target language parse to contain each of the 10 edges, we require that the expected number of edges from this set is at least 10η, where η is a strength parameter. This gives the parser freedom to have some uncertainty about which edges to include, or alternatively to choose to exclude some of the transferred edges.

A more serious problem for transferring parse information across languages are structural differences and grammar annotation choices between the two languages. For example dealing with auxiliary verbs and reflexive constructions. Hwa et al. (2005) also note these problems and solve them by introducing dozens of rules to transform the transferred parse trees. We discuss these differences in detail in the experimental section and use our framework introduce a very small number of rules to cover the most common structural differences.

3 Parsing Models

We explored two parsing models: a generative model used by several authors for unsupervised induction and a discriminative model used for fully supervised training.

The discriminative parser is based on the edge-factored model and features of the MST-Parser (McDonald et al., 2005). The parsing model defines a conditional distribution $p_\theta(\mathbf{z} \mid \mathbf{x})$ over each projective parse tree $\mathbf{z}$ for a particular sentence $\mathbf{x}$, parameterized by a vector θ. The prob-

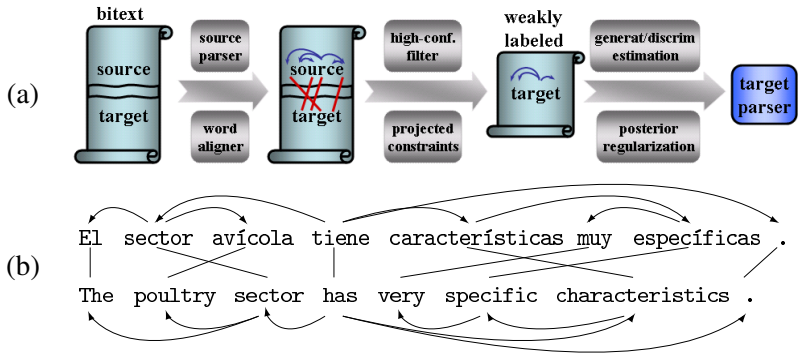


Figure 1: (a) Overview of our grammar induction approach via bitext: the source (English) is parsed and word-aligned with target; after filtering, projected dependencies define constraints over target parse tree space, providing weak supervision for learning a target grammar. (b) An example word-aligned sentence pair with perfectly projected dependencies.

ability of any particular parse is

$$p_\theta(\mathbf{z} \mid \mathbf{x}) \propto \prod_{z \in \mathbf{z}} e^{\theta \cdot \phi(z, \mathbf{x})}, \quad (1)$$

where z is a directed edge contained in the parse tree $\mathbf{z}$ and ϕ is a feature function. In the fully supervised experiments we run for comparison, parameter estimation is performed by stochastic gradient ascent on the conditional likelihood function, similar to maximum entropy models or conditional random fields. One needs to be able to compute expectations of the features $\phi(z, \mathbf{x})$ under the distribution $p_\theta(z \mid \mathbf{x})$. A version of the inside-outside algorithm (Lee and Choi, 1997) performs this computation. Viterbi decoding is done using Eisner's algorithm (Eisner, 1996).

We also used a generative model based on dependency model with valence (Klein and Manning, 2004). Under this model, the probability of a particular parse $\mathbf{z}$ and a sentence with part of speech tags $\mathbf{x}$ is given by

$$p_\theta(\mathbf{z}, \mathbf{x}) = p_{\text{root}}(r(\mathbf{x})) \cdot \quad (2)$$
$$\Big(\prod_{z \in \mathbf{z}} p_{\neg\text{stop}}(z_p, z_d, v_z)\, p_{\text{child}}(z_p, z_d, z_c) \Big) \cdot$$
$$\Big(\prod_{x \in \mathbf{x}} p_{\text{stop}}(x, \text{left}, v_l)\, p_{\text{stop}}(x, \text{right}, v_r) \Big)$$

where $r(\mathbf{x})$ is the part of speech tag of the root of the parse tree $\mathbf{z}$, z is an edge from parent z_p to child z_c in direction z_d, either left or right, and v_z indicates valency—false if z_p has no other children further from it in direction z_d than z_c, true otherwise. The valencies v_r / v_l are marked as true if x has any children on the left/right in $\mathbf{z}$, false otherwise.

4 Posterior Regularization

Graça et al. (2008) introduce an estimation framework that incorporates side-information into unsupervised problems in the form of linear constraints on posterior expectations. In grammar transfer, our basic constraint is of the form: the expected proportion of conserved edges in a sentence pair is at least η (the exact proportion we used was 0.9, which was determined using unlabeled data as described in Section 5). Specifically, let $C_\mathbf{x}$ be the set of directed edges projected from English for a given sentence $\mathbf{x}$, then given a parse $\mathbf{z}$, the proportion of conserved edges is $f(\mathbf{x}, \mathbf{z}) = \frac{1}{|C_\mathbf{x}|} \sum_{z \in \mathbf{z}} \mathbf{1}(z \in C_\mathbf{x})$ and the expected proportion of conserved edges under distribution $p(\mathbf{z} \mid \mathbf{x})$ is

$$\mathbf{E}_p[f(\mathbf{x}, \mathbf{z})] = \frac{1}{|C_\mathbf{x}|} \sum_{z \in C_\mathbf{x}} p(z \mid \mathbf{x}).$$

The posterior regularization framework (Graça et al., 2008) was originally defined for generative unsupervised learning. The standard objective is to minimize the negative marginal log-likelihood of the data : $\widehat{\mathbf{E}}[-\log p_\theta(\mathbf{x})] = \widehat{\mathbf{E}}[-\log \sum_\mathbf{z} p_\theta(\mathbf{z}, \mathbf{x})]$ over the parameters θ (we use $\widehat{\mathbf{E}}$ to denote expectation over the sample sentences $\mathbf{x}$). We typically also add standard regularization term on θ, resulting from a parameter prior $-\log p(\theta) = R(\theta)$, where $p(\theta)$ is Gaussian for the MST-Parser models and Dirichlet for the valence model.

To introduce supervision into the model, we define a set $\mathcal{Q}_\mathbf{x}$ of distributions over the hidden variables $\mathbf{z}$ satisfying the desired posterior constraints in terms of linear equalities or inequalities on feature expectations (we use inequalities in this paper):

$$\mathcal{Q}_\mathbf{x} = \{q(\mathbf{z}) : \mathbf{E}[\mathbf{f}(\mathbf{x}, \mathbf{z})] \leq \mathbf{b}\}.$$

Basic Uni-gram Features
x_i-word, x_i-pos
x_i-word
x_i-pos
x_j-word, x_j-pos
x_j-word
x_j-pos

Basic Bi-gram Features
x_i-word, x_i-pos, x_j-word, x_j-pos
x_i-pos, x_j-word, x_j-pos
x_i-word, x_j-word, x_j-pos
x_i-word, x_i-pos, x_j-pos
x_i-word, x_i-pos, x_j-word
x_i-word, x_j-word
x_i-pos, x_j-pos

In Between POS Features
x_i-pos, b-pos, x_j-pos

Surrounding Word POS Features
x_i-pos, x_i-pos+1, x_j-pos-1, x_j-pos
x_i-pos-1, x_i-pos, x_j-pos-1, x_j-pos
x_i-pos, x_i-pos+1, x_j-pos, x_j-pos+1
x_i-pos-1, x_i-pos, x_j-pos, x_j-pos+1

Table 1: Features used by the MSTParser. For each edge (i, j), x_i-word is the parent word and x_j-word is the child word, analogously for POS tags. The +1 and -1 denote preceeding and following tokens in the sentence, while b denotes tokens between x_i and x_j.

In this paper, for example, we use the conserved-edge-proportion constraint as defined above. The marginal log-likelihood objective is then modified with a penalty for deviation from the desired set of distributions, measured by KL-divergence from the set $\mathcal{Q}_{\mathbf{x}}$, $\mathrm{KL}(\mathcal{Q}_{\mathbf{x}}||p_\theta(\mathbf{z}|\mathbf{x})) = \min_{q\in\mathcal{Q}_{\mathbf{x}}} \mathrm{KL}(q(\mathbf{z})||p_\theta(\mathbf{z}|\mathbf{x}))$. The generative learning objective is to minimize:

$$\widehat{\mathbf{E}}[-\log p_\theta(\mathbf{x})] + R(\theta) + \widehat{\mathbf{E}}[\mathrm{KL}(\mathcal{Q}_{\mathbf{x}}||p_\theta(\mathbf{z} \mid \mathbf{x}))].$$

For discriminative estimation (Ganchev et al., 2008), we do not attempt to model the marginal distribution of $\mathbf{x}$, so we simply have the two regularization terms:

$$R(\theta) + \widehat{\mathbf{E}}[\mathrm{KL}(\mathcal{Q}_{\mathbf{x}}||p_\theta(\mathbf{z} \mid \mathbf{x}))].$$

Note that the idea of regularizing moments is related to generalized expectation criteria algorithm of Mann and McCallum (2007), as we discuss in the related work section below. In general, the objectives above are not convex in θ. To optimize these objectives, we follow an Expectation Maximization-like scheme. Recall that standard EM iterates two steps. An E-step computes a probability distribution over the model's hidden variables (posterior probabilities) and an M-step that updates the model's parameters based on that distribution. The posterior-regularized EM algorithm leaves the M-step unchanged, but involves projecting the posteriors onto a constraint set after they are computed for each sentence $\mathbf{x}$:

$$\begin{aligned} &\arg\min_q \mathrm{KL}(q(\mathbf{z}) \parallel p_\theta(\mathbf{z}|\mathbf{x})) \\ &\quad \text{s.t. } \mathbf{E}_q[\mathbf{f}(\mathbf{x}, \mathbf{z})] \leq \mathbf{b}, \end{aligned} \tag{3}$$

where $p_\theta(\mathbf{z}|\mathbf{x})$ are the posteriors. The new posteriors $q(\mathbf{z})$ are used to compute sufficient statistics for this instance and hence to update the model's parameters in the M-step for either the generative or discriminative setting.

The optimization problem in Equation 3 can be efficiently solved in its dual formulation:

$$\arg\min_{\lambda\geq 0} \mathbf{b}^\top\lambda + \log \sum_{\mathbf{z}} p_\theta(\mathbf{z} \mid \mathbf{x}) \exp\{-\lambda^\top \mathbf{f}(\mathbf{x}, \mathbf{z})\}. \tag{4}$$

Given λ, the primal solution is given by: $q(\mathbf{z}) = p_\theta(\mathbf{z} \mid \mathbf{x}) \exp\{-\lambda^\top \mathbf{f}(\mathbf{x}, \mathbf{z})\}/Z$, where Z is a normalization constant. There is one dual variable per expectation constraint, and we can optimize them by projected gradient descent, similar to log-linear model estimation. The gradient with respect to λ is given by: $\mathbf{b} - \mathbf{E}_q[\mathbf{f}(\mathbf{x}, \mathbf{z})]$, so it involves computing expectations under the distribution $q(\mathbf{z})$. This remains tractable as long as features factor by edge, $f(\mathbf{x}, \mathbf{z}) = \sum_{z\in\mathbf{z}} f(\mathbf{x}, z)$, because that ensures that $q(\mathbf{z})$ will have the same form as $p_\theta(\mathbf{z} \mid \mathbf{x})$. Furthermore, since the constraints are per instance, we can use incremental or online version of EM (Neal and Hinton, 1998), where we update parameters θ after posterior-constrained E-step on each instance $\mathbf{x}$.

5 Experiments

We conducted experiments on two languages: Bulgarian and Spanish, using each of the parsing models. The Bulgarian experiments transfer a parser from English to Bulgarian, using the Open-Subtitles corpus (Tiedemann, 2007). The Spanish experiments transfer from English to Spanish using the Spanish portion of the Europarl corpus (Koehn, 2005). For both corpora, we performed word alignments with the open source PostCAT (Graça et al., 2009) toolkit. We used the Tokyo tagger (Tsuruoka and Tsujii, 2005) to POS tag the English tokens, and generated parses using the first-order model of McDonald et al. (2005) with projective decoding, trained on sections 2-21 of the Penn treebank with dependencies extracted using the head rules of Yamada and Matsumoto (2003b). For Bulgarian we trained the Stanford POS tagger (Toutanova et al., 2003) on the Bul-

	Discriminative model					Generative model				
	Bulgarian			Spanish		Bulgarian			Spanish	
	no rules	2 rules	7 rules	no rules	3 rules	no rules	2 rules	7 rules	no rules	3 rules
Baseline	63.8	72.1	72.6	67.6	69.0	66.5	69.1	**71.0**	68.2	71.3
Post.Reg.	**66.9**	**77.5**	**78.3**	**70.6**	**72.3**	**67.8**	**70.7**	70.8	**69.5**	**72.8**

Table 2: Comparison between transferring a single tree of edges and transferring all possible projected edges. The transfer models were trained on 10k sentences of length up to 20, all models tested on CoNLL train sentences of up to 10 words. Punctuation was stripped at train time.

gtreebank corpus from CoNLL X. The Spanish Europarl data was POS tagged with the FreeLing language analyzer (Atserias et al., 2006). The discriminative model used the same features as MSTParser, summarized in Table 1.

In order to evaluate our method, we a baseline inspired by Hwa et al. (2005). The baseline constructs a full parse tree from the incomplete and possibly conflicting transferred edges using a simple random process. We start with no edges and try to add edges one at a time verifying at each step that it is possible to complete the tree. We first try to add the transferred edges in random order, then for each orphan node we try all possible parents (both in random order). We then use this full labeling as supervision for a parser. Note that this baseline is very similar to the first iteration of our model, since for a large corpus the different random choices made in different sentences tend to smooth each other out. We also tried to create rules for the adoption of orphans, but the simple rules we tried added bias and performed worse than the baseline we report. Table 2 shows attachment accuracy of our method and the baseline for both language pairs under several conditions. By attachment accuracy we mean the fraction of words assigned the correct parent. The experimental details are described in this section. Link-left baselines for these corpora are much lower: 33.8% and 27.9% for Bulgarian and Spanish respectively.

5.1 Preprocessing

Preliminary experiments showed that our word alignments were not always appropriate for syntactic transfer, even when they were correct for translation. For example, the English "bike/V" could be translated in French as "aller/V en vélo/N", where the word "bike" would be aligned with "vélo". While this captures some of the semantic shared information in the two languages, we have no expectation that the noun "vélo" will have a similar syntactic behavior to the verb "bike". To prevent such false transfer, we filter out alignments between incompatible POS tags. In both language pairs, filtering out noun-verb alignments gave the biggest improvement.

Both corpora also contain sentence fragments, either because of question responses or fragmented speech in movie subtitles or because of voting announcements and similar formulaic sentences in the parliamentary proceedings. We overcome this problem by filtering out sentences that do not have a verb as the English root or for which the English root is not aligned to a verb in the target language. For the subtitles corpus we also remove sentences that end in an ellipsis or contain more than one comma. Finally, following (Klein and Manning, 2004) we strip out punctuation from the sentences. For the discriminative model this did not affect results significantly but improved them slightly in most cases. We found that the generative model gets confused by punctuation and tends to predict that periods at the end of sentences are the parents of words in the sentence.

Our basic model uses constraints of the form: the expected proportion of conserved edges in a sentence pair is at least $\eta = 90\%$.[1]

5.2 No Language-Specific Rules

We call the generic model described above "no-rules" to distinguish it from the language-specific constraints we introduce in the sequel. The no rules columns of Table 2 summarize the performance in this basic setting. Discriminative models outperform the generative models in the majority of cases. The left panel of Table 3 shows the most common errors by child POS tag, as well as by true parent and guessed parent POS tag.

Figure 2 shows that the discriminative model continues to improve with more transfer-type data

[1] We chose η in the following way: we split the unlabeled parallel text into two portions. We trained a models with different η on one portion and ran it on the other portion. We chose the model with the highest fraction of conserved constraints on the second portion.

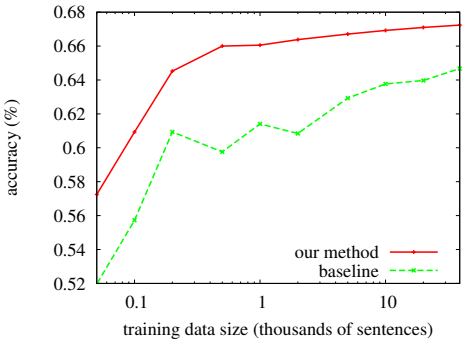


Figure 2: Learning curve of the discriminative no-rules transfer model on Bulgarian bitext, testing on CoNLL train sentences of up to 10 words.

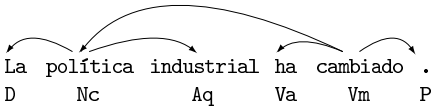


Figure 3: A Spanish example where an auxiliary verb dominates the main verb.

up to at least 40 thousand sentences.

5.3 Annotation guidelines and constraints

Using the straightforward approach outlined above is a dramatic improvement over the standard link-left baseline (and the unsupervised generative model as we discuss below), however it doesn't have any information about the annotation guidelines used for the testing corpus. For example, the Bulgarian corpus has an unusual treatment of nonfinite clauses. Figure 4 shows an example. We see that the "да" is the parent of both the verb and its object, which is different than the treatment in the English corpus.

We propose to deal with these annotation dissimilarities by creating very simple rules. For Spanish, we have three rules. The first rule sets main verbs to dominate auxiliary verbs. Specifically, whenever an auxiliary precedes a main verb the main verb becomes its parent and adopts its children; if there is only one main verb it becomes the root of the sentence; main verbs also become

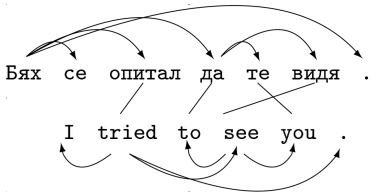


Figure 4: An example where transfer fails because of different handling of reflexives and nonfinite clauses. The alignment links provide correct glosses for Bulgarian words. "Бях" is a past tense marker while "се" is a reflexive marker.

parents of pronouns, adverbs, and common nouns that directly preceed auxiliary verbs. By adopting children we mean that we change the parent of transferred edges to be the adopting node. The second Spanish rule states that the first element of an adjective-noun or noun-adjective pair dominates the second; the first element also adopts the children of the second element. The third and final Spanish rule sets all prepositions to be children of the first main verb in the sentence, unless the preposition is a "de" located between two noun phrases. In this later case, we set the closest noun in the first of the two noun phrases as the preposition's parent.

For Bulgarian the first rule is that "да" should dominate all words until the next verb and adopt their noun, preposition, particle and adverb children. The second rule is that auxiliary verbs should dominate main verbs and adopt their children. We have a list of 12 Bulgarian auxiliary verbs. The "seven rules" experiments add rules for 5 more words similar to the rule for "да", specifically "че", "ли", "какво", "не", "за". Table 3 compares the errors for different linguistic rules. When we train using the "да" rule and the rules for auxiliary verbs, the model learns that main verbs attach to auxiliary verbs and that "да" dominates its nonfinite clause. This causes an improvement in the attachment of verbs, and also drastically reduces words being attached to verbs instead of particles. The latter is expected because "да" is analyzed as a particle in the Bulgarian POS tagset. We see an improvement in root/verb confusions since "да" is sometimes errenously attached to a the following verb rather than being the root of the sentence.

The rightmost panel of Table 3 shows similar analysis when we also use the rules for the five other closed-class words. We see an improvement in attachments in all categories, but no qualitative change is visible. The reason for this is probably that these words are relatively rare, but by encouraging the model to add an edge, it also rules out incorrect edges that would cross it. Consequently we are seeing improvements not only directly from the constraints we enforce but also indirectly as types of edges that tend to get ruled out.

5.4 Generative parser

The generative model we use is a state of the art model for unsupervised parsing and is our only

No Rules

child POS	acc(%)	errors	parent POS	errors
V	65.2	2237	T/V	2175
N	73.8	1938	V/V	1305
P	58.5	1705	N/V	1112
R	70.3	961	root/V	555

Two Rules

child POS	acc(%)	errors	parent POS	errors
N	78.7	1572	N/V	938
P	70.2	1224	V/V	734
V	84.4	1002	V/N	529
R	79.3	670	N/N	376

Seven Rules

child POS	acc(%)	errors	parent POS	errors
N	79.3	1532	N/V	1116
P	75.7	998	V/V	560
R	69.3	993	V/N	507
V	86.2	889	N/N	450

Table 3: Top 4 discriminative parser errors by child POS tag and true/guess parent POS tag in the Bulgarian CoNLL train data of length up to 10. Training with no language-specific rules (left); two rules (center); and seven rules (right). POS meanings: V verb, N noun, P pronoun, R preposition, T particle. Accuracies are by child or parent truth/guess POS tag.

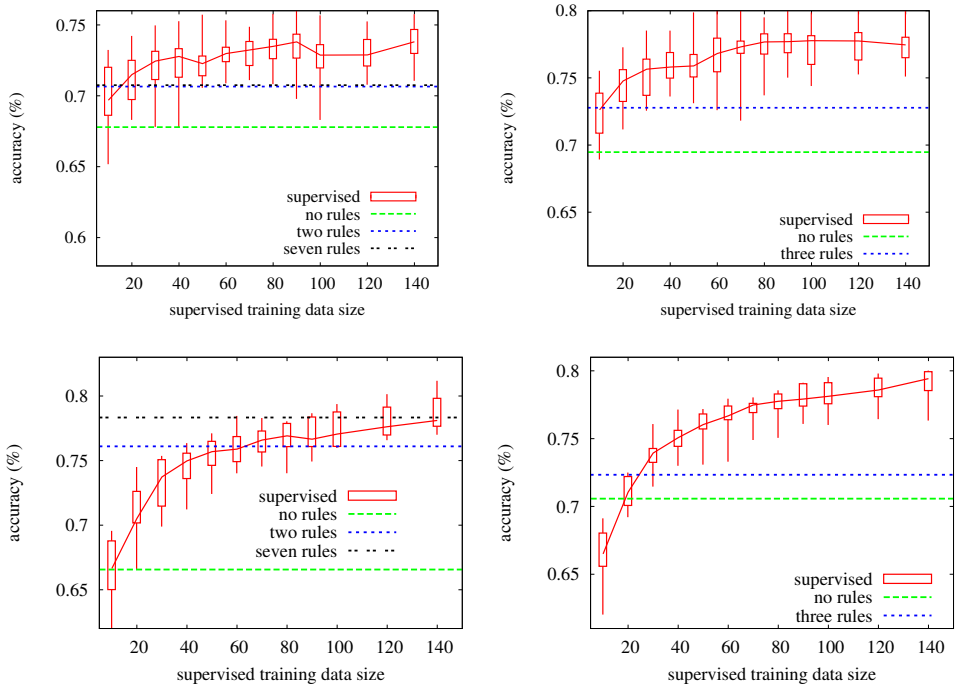


Figure 5: Comparison to parsers with supervised estimation and transfer. Top: Generative. Bottom: Discriminative. Left: Bulgarian. Right: Spanish. The transfer models were trained on 10k sentences all of length at most 20, all models tested on CoNLL train sentences of up to 10 words. The x-axis shows the number of examples used to train the supervised model. Boxes show first and third quartile, whiskers extend to max and min, with the line passing through the median. Supervised experiments used 30 random samples from CoNLL train.

fully unsupervised baseline. As smoothing we add a very small backoff probability of 4.5×10^{-5} to each learned paramter. Unfortunately, we found generative model performance was disappointing overall. The maximum unsupervised accuracy it achieved on the Bulgarian data is 47.6% with initialization from Klein and Manning (2004) and this result is not stable. Changing the initialization parameters, training sample, or maximum sentence length used for training drastically affected the results, even for samples with several thousand sentences. When we use the transferred information to constrain the learning, EM stabilizes and achieves much better performance. Even setting all parameters equal at the outset does not prevent the model from learning the dependency structure of the aligned language. The top panels in Figure 5 show the results in this setting. We see that performance is still always below the accuracy achieved by supervised training on 20 annotated sentences. However, the improvement in stability makes the algorithm much more usable. As we shall see below, the discriminative parser performs even better than the generative model.

5.5 Discriminative parser

We trained our discriminative parser for 100 iterations of online EM with a Gaussian prior variance of 100. Results for the discriminative parser are shown in the bottom panels of Figure 5. The supervised experiments are given to provide context for the accuracies. For Bulgarian, we see that without any hints about the annotation guidelines, the transfer system performs better than an unsu-

pervised parser, comparable to a supervised parser trained on 10 sentences. However, if we specify just the two rules for "да" and verb conjugations performance jumps to that of training on 60-70 fully labeled sentences. If we have just a little more prior knowledge about how closed-class words are handled, performance jumps above 140 fully labeled sentence equivalent.

We observed another desirable property of the discriminative model. While the generative model can get confused and perform poorly when the training data contains very long sentences, the discriminative parser does not appear to have this drawback. In fact we observed that as the maximum training sentence length increased, the parsing performance also improved.

6 Related Work

Our work most closely relates to Hwa et al. (2005), who proposed to learn generative dependency grammars using Collins' parser (Collins, 1999) by constructing full target parses via projected dependencies and completion/transformation rules. Hwa et al. (2005) found that transferring dependencies directly was not sufficient to get a parser with reasonable performance, even when both the source language parses and the word alignments are performed by hand. They adjusted for this by introducing on the order of one or two dozen language-specific transformation rules to complete target parses for unaligned words and to account for diverging annotation rules. Transferring from English to Spanish in this way, they achieve 72.1% and transferring to Chinese they achieve 53.9%.

Our learning method is very closely related to the work of (Mann and McCallum, 2007; Mann and McCallum, 2008) who concurrently developed the idea of using penalties based on posterior expectations of features not necessarily in the model in order to guide learning. They call their method generalized expectation constraints or alternatively expectation regularization. In this volume (Druck et al., 2009) use this framework to train a dependency parser based on constraints stated as corpus-wide expected values of linguistic rules. The rules select a class of edges (e.g. auxiliary verb to main verb) and require that the expectation of these be close to some value. The main difference between this work and theirs is the source of the information (a linguistic informant vs. cross-lingual projection). Also, we define our regularization with respect to inequality constraints (the model is not penalized for exceeding the required model expectations), while they require moments to be close to an estimated value. We suspect that the two learning methods could perform comparably when they exploit similar information.

7 Conclusion

In this paper, we proposed a novel and effective learning scheme for transferring dependency parses across bitext. By enforcing projected dependency constraints approximately and in expectation, our framework allows robust learning from noisy partially supervised target sentences, instead of committing to entire parses. We show that discriminative training generally outperforms generative approaches even in this very weakly supervised setting. By adding easily specified language-specific constraints, our models begin to rival strong supervised baselines for small amounts of data. Our framework can handle a wide range of constraints and we are currently exploring richer syntactic constraints that involve conservation of multiple edge constructions as well as constraints on conservation of surface length of dependencies.

Acknowledgments

This work was partially supported by an Integrative Graduate Education and Research Traineeship grant from National Science Foundation (NSFIGERT 0504487), by ARO MURI SUBTLE W911NF-07-1-0216 and by the European Projects AsIsKnown (FP6-028044) and LTfLL (FP7-212578).

References

A. Abeillé. 2003. *Treebanks: Building and Using Parsed Corpora*. Springer.

H. Alshawi, S. Bangalore, and S. Douglas. 2000. Learning dependency translation models as collections of finite state head transducers. *Computational Linguistics*, 26(1).

J. Atserias, B. Casas, E. Comelles, M. González, L. Padró, and M. Padró. 2006. Freeling 1.3: Syntactic and semantic services in an open-source nlp library. In *Proc. LREC*, Genoa, Italy.

P. F. Brown, S. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1994. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

C. Chelba, D. Engle, F. Jelinek, V. Jimenez, S. Khudanpur, L. Mangu, H. Printz, E. Ristad, R. Rosenfeld, A. Stolcke, and D. Wu. 1997. Structure and performance of a dependency language model. In *Proc. Eurospeech*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

G. Druck, G. Mann, and A. McCallum. 2009. Semi-supervised learning of dependency parsers using generalized expectation criteria. In *Proc. ACL*.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: an exploration. In *Proc. CoLing*.

H. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proc. EMNLP*, pages 304–311.

K. Ganchev, J. Graca, J. Blitzer, and B. Taskar. 2008. Multi-view learning over structured and non-identical outputs. In *Proc. UAI*.

J. Graça, K. Ganchev, and B. Taskar. 2008. Expectation maximization and posterior constraints. In *Proc. NIPS*.

J. Graça, K. Ganchev, and B. Taskar. 2009. Postcat - posterior constrained alignment toolkit. In *The Third Machine Translation Marathon*.

A. Haghighi, A. Ng, and C. Manning. 2005. Robust textual inference via graph matching. In *Proc. EMNLP*.

R. Hwa, P. Resnik, A. Weinberg, C. Cabezas, and O. Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11:11–311.

D. Klein and C. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proc. of ACL*.

P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*.

S. Lee and K. Choi. 1997. Reestimation and best-first parsing algorithm for probabilistic dependency grammar. In *In WVLC-5*, pages 41–55.

G. Mann and A. McCallum. 2007. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proc. ICML*.

G. Mann and A. McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proc. ACL*, pages 870 – 878.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. ACL*, pages 91–98.

I. Mel'čuk. 1988. *Dependency syntax: theory and practice*. SUNY. inci.

P. Merlo, S. Stevenson, V. Tsang, and G. Allaria. 2002. A multilingual paradigm for automatic verb classification. In *Proc. ACL*.

R. M. Neal and G. E. Hinton. 1998. A new view of the EM algorithm that justifies incremental, sparse and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer.

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. ACL*.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. EMNLP-CoNLL*.

F. J. Och and H. Ney. 2000. Improved statistical alignment models. In *Proc. ACL*.

C. Quirk, A. Menezes, and C. Cherry. 2005. Dependency treelet translation: syntactically informed phrasal smt. In *Proc. ACL*.

L. Shen, J. Xu, and R. Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. of ACL*.

N. Smith and J. Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proc. ACL*.

J. Tiedemann. 2007. Building a multilingual parallel subtitle corpus. In *Proc. CLIN*.

K. Toutanova, D. Klein, C. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. HLT-NAACL*.

Y. Tsuruoka and J. Tsujii. 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Proc. HLT/EMNLP*.

H. Yamada and Y. Matsumoto. 2003a. Statistical dependency analysis with support vector machines. In *Proc. IWPT*, pages 195–206.

H. Yamada and Y. Matsumoto. 2003b. Statistical dependency analysis with support vector machines. In *Proc. IWPT*.

D. Yarowsky and G. Ngai. 2001. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proc. NAACL*.

D. Yarowsky, G. Ngai, and R. Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proc. HLT*.

Cross-Domain Dependency Parsing Using a Deep Linguistic Grammar

Yi Zhang
LT-Lab, DFKI GmbH and
Dept of Computational Linguistics
Saarland University
D-66123 Saarbrücken, Germany
yzhang@coli.uni-sb.de

Rui Wang
Dept of Computational Linguistics
Saarland University
66123 Saarbrücken, Germany
rwang@coli.uni-sb.de

Abstract

Pure statistical parsing systems achieves high in-domain accuracy but performs poorly out-domain. In this paper, we propose two different approaches to produce syntactic dependency structures using a large-scale hand-crafted HPSG grammar. The dependency backbone of an HPSG analysis is used to provide general linguistic insights which, when combined with state-of-the-art statistical dependency parsing models, achieves performance improvements on out-domain tests.[†]

1 Introduction

Syntactic dependency parsing is attracting more and more research focus in recent years, partially due to its theory-neutral representation, but also thanks to its wide deployment in various NLP tasks (machine translation, textual entailment recognition, question answering, information extraction, etc.). In combination with machine learning methods, several statistical dependency parsing models have reached comparable high parsing accuracy (McDonald et al., 2005b; Nivre et al., 2007b). In the meantime, successful continuation of CoNLL Shared Tasks since 2006 (Buchholz and Marsi, 2006; Nivre et al., 2007a; Surdeanu et al., 2008) have witnessed how easy it has become to train a statistical syntactic dependency parser provided that there is annotated treebank.

While the dissemination continues towards various languages, several issues arise with such purely data-driven approaches. One common observation is that statistical parser performance drops significantly when tested on a dataset *different* from the training set. For instance, when using the Wall Street Journal (WSJ) sections of the Penn Treebank (Marcus et al., 1993) as training set, tests on BROWN Sections typically result in a 6-8% drop in labeled attachment scores, although the average sentence length is much shorter in BROWN than that in WSJ. The common interpretation is that the test set is heterogeneous to the training set, hence in a different "domain" (in a loose sense). The typical cause of this is that the model overfits the training domain. The concerns over random choice of training corpus leading to linguistically inadequate parsing systems increase over time.

While the statistical revolution in the field of computational linguistics gaining high publicity, the conventional symbolic grammar-based parsing approaches have undergone a quiet period of development during the past decade, and reemerged very recently with several large scale grammar-driven parsing systems, benefiting from the combination of well-established linguistic theories and data-driven stochastic models. The obvious advantage of such systems over pure statistical parsers is their usage of hand-coded linguistic knowledge irrespective of the training data. A common problem with grammar-based parser is the lack of robustness. Also it is difficult to derive grammar compatible annotations to train the statistical components.

2 Parser Domain Adaptation

In recent years, two statistical dependency parsing systems, MaltParser (Nivre et al., 2007b) and MSTParser (McDonald et al., 2005b), representing different threads of research in data-driven machine learning approaches have obtained high publicity, for their state-of-the-art performances in open competitions such as CoNLL Shared Tasks. MaltParser follows the *transition-based* approach, where parsing is done through a series of actions deterministically predicted by an *oracle* model. MSTParser, on the other hand, follows

[†]The first author thanks the German Excellence Cluster of Multimodal Computing and Interaction for the support of the work. The second author is funded by the PIRE PhD scholarship program.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 378–386,
Suntec, Singapore, 2-7 August 2009.

the *graph-based* approach where the best parse tree is acquired by searching for a spanning tree which maximizes the score on either a partially or a fully connected graph with all words in the sentence as nodes (Eisner, 1996; McDonald et al., 2005b).

As reported in various evaluation competitions, the two systems achieved comparable performances. More recently, approaches of combining these two parsers achieved even better dependency accuracy (Nivre and McDonald, 2008). Granted for the differences between their approaches, both systems heavily rely on machine learning methods to estimate the parsing model from an annotated corpus as training set. Due to the heavy cost of developing high quality large scale syntactically annotated corpora, even for a resource-rich language like English, only very few of them meets the criteria for training a general purpose statistical parsing model. For instance, the text style of WSJ is newswire, and most of the sentences are statements. Being lack of non-statements in the training data could cause problems, when the testing data contain many interrogative or imperative sentences as in the BROWN corpus. Therefore, the unbalanced distribution of linguistic phenomena in the training data leads to inadequate parser output structures. Also, the financial domain specific terminology seen in WSJ can skew the interpretation of daily life sentences seen in BROWN.

There has been a substantial amount of work on parser adaptation, especially from WSJ to BROWN. Gildea (2001) compared results from different combinations of the training and testing data to demonstrate that the size of the feature model can be reduced via excluding "domain-dependent" features, while the performance could still be preserved. Furthermore, he also pointed out that if the additional training data is heterogeneous from the original one, the parser will not obtain a substantially better performance. Bacchiani et al. (2006) generalized the previous approaches using a maximum a posteriori (MAP) framework and proposed both supervised and unsupervised adaptation of statistical parsers. McClosky et al. (2006) and McClosky et al. (2008) have shown that out-domain parser performance can be improved with self-training on a large amount of unlabeled data. Most of these approaches focused on the machine learning perspective instead of the linguistic knowledge embraced in the parsers. Little study has been reported on approaches of incorporating linguistic features to make the parser less dependent on the nature of training and testing dataset, without resorting to huge amount of unlabeled out-domain data. In addition, most of the previous work have been focusing on constituent-based parsing, while the domain adaptation of the dependency parsing has not been fully explored.

Taking a different approach towards parsing, grammar-based parsers appear to have much linguistic knowledge encoded within the grammars. In recent years, several of these linguistically motivated grammar-driven parsing systems achieved high accuracy which are comparable to the treebank-based statistical parsers. Notably are the constraint-based linguistic frameworks with mathematical rigor, and provide grammatical analyses for a large variety of phenomena. For instance, the Head-Driven Phrase Structure Grammar (Pollard and Sag, 1994) has been successfully applied in several parsing systems for more than a dozen of languages. Some of these grammars, such as the English Resource Grammar (ERG; Flickinger (2002)), have undergone over decades of continuous development, and provide precise linguistic analyses for a broad range of phenomena. These linguistic knowledge are encoded in highly generalized form according to linguists' reflection for the target languages, and tend to be largely independent from any specific domain.

The main issue of parsing with precision grammars is that broad coverage and high precision on linguistic phenomena do not directly guarantee robustness of the parser with noisy real world texts. Also, the detailed linguistic analysis is not always of the highest interest to all NLP applications. It is not always straightforward to scale down the detailed analyses embraced by deep grammars to a shallower representation which is more accessible for specific NLP tasks. On the other hand, since the dependency representation is relatively theory-neutral, it is possible to convert from other frameworks into its backbone representation in dependencies. For HPSG, this is further assisted by the clear marking of head daughters in headed phrases. Although the statistical components of the grammar-driven parser might be still biased by the training domain, the hand-coded grammar rules guarantee the basic linguistic constraints to be met. This not to say that domain adaptation is

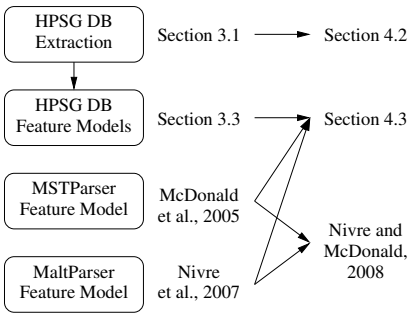


Figure 1: Different dependency parsing models and their combinations. DB stands for dependency backbone.

not an issue for grammar-based parsing systems, but the built-in linguistic knowledge can be explored to reduce the performance drop in pure statistical approaches.

3 Dependency Parsing with HPSG

In this section, we explore two possible applications of the HPSG parsing onto the syntactic dependency parsing task. One is to extract dependency backbone from the HPSG analyses of the sentences and directly convert them into the target representation; the other way is to encode the HPSG outputs as additional features into the existing statistical dependency parsing models. In the previous work, Nivre and McDonald (2008) have integrated MSTParser and MaltParser by feeding one parser's output as features into the other. The relationships between our work and their work are roughly shown in Figure 1.

3.1 Extracting Dependency Backbone from HPSG Derivation Tree

Given a sentence, each parse produced by the parser is represented by a typed feature structure, which recursively embeds smaller feature structures for lower level phrases or words. For the purpose of dependency backbone extraction, we only look at the derivation tree which corresponds to the constituent tree of an HPSG analysis, with all non-terminal nodes labeled by the names of the grammar rules applied. Figure 2 shows an example. Note that all grammar rules in ERG are either unary or binary, giving us relatively *deep* trees when compared with annotations such as Penn Treebank. Conceptually, this conversion is similar to the conversions from deeper structures to GR reprsentations reported by Clark and Curran (2007) and Miyao et al. (2007).

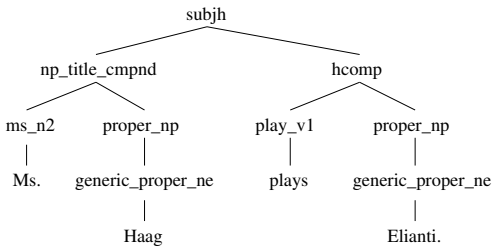


Figure 2: An example of an HPSG derivation tree with ERG

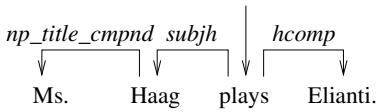


Figure 3: An HPSG dependency backbone structure

The dependency backbone extraction works by first identifying the head daughter for each binary grammar rule, and then propagating the head word of the head daughter upwards to their parents, and finally creating a dependency relation, labeled with the HPSG rule name of the parent node, from the head word of the parent to the head word of the non-head daughter. See Figure 3 for an example of such an extracted backbone.

For the experiments in this paper, we used July-08 version of the ERG, which contains in total 185 grammar rules (morphological rules are not counted). Among them, 61 are unary rules, and 124 are binary. Many of the binary rules are clearly marked as headed phrases. The grammar also indicates whether the head is on the left (*head-initial*) or on the right (*head-final*). However, there are still quite a few binary rules which are not marked as headed-phrases (according to the linguistic theory), e.g. rules to handle coordinations, appositions, compound nouns, etc. For these rules, we refer to the conversion of the Penn Treebank into dependency structures used in the CoNLL 2008 Shared Task, and mark the heads of these rules in a way that will arrive at a compatible dependency backbone. For instance, the left most daughters of coordination rules are marked as heads. In combination with the right-branching analysis of coordination in ERG, this leads to the same dependency attachment in the CoNLL syntax. Eventually, 37 binary rules are marked with a head daughter on the left, and 86 with a head daughter on the right.

Although the extracted dependency is similar to

the CoNLL shared task dependency structures, minor systematic differences still exist for some phenomena. For example, the possessive "'s" is annotated to be governed by its preceding word in CoNLL dependency; while in HPSG, it is treated as the head of a "specifier-head" construction, hence governing the preceding word in the dependency backbone. With several simple tree rewriting rules, we are able to fix the most frequent inconsistencies. With the rule-based backbone extraction and repair, we can finally turn our HPSG parser outputs into dependency structures[1]. The unlabeled attachment agreement between the HPSG backbone and CoNLL dependency annotation will be shown in Section 4.2.

3.2 Robust Parsing with HPSG

As mentioned in Section 2, one pitfall of using a precision-oriented grammar in parsing is its lack of robustness. Even with a large scale broad coverage grammar like ERG, using our settings we only achieved 75% of sentential coverage[2]. Given that the grammar has never been fine-tuned for the financial domain, such coverage is very encouraging. But still, the remaining unparsed sentences comprise a big coverage gap.

Different strategies can be taken here. One can either keep the high precision by only looking at full parses from the HPSG parser, of which the analyses are completely admitted by grammar constraints. Or one can trade precision for extra robustness by looking at the most probable incomplete analysis. Several partial parsing strategies have been proposed (Kasper et al., 1999; Zhang and Kordoni, 2008) as the robust fallbacks for the parser when no available analysis can be derived. In our experiment, we select the sequence of most likely fragment analyses according to their local disambiguation scores as the partial parse. When combined with the dependency backbone extraction, partial parses generate disjoint tree fragments. We simply attach all fragments onto the virtual root node.

[1]It is also possible map from HPSG rule names (together with the part-of-speech of head and dependent) to CoNLL dependency labels. This remains to be explored in the future.

[2]More recent study shows that with carefully designed retokenization and preprocessing rules, over 80% sentential coverage can be achieved on the WSJ sections of the Penn Treebank data using the same version of ERG. The numbers reported in this paper are based on a simpler preprocessor, using rather strict time/memory limits for the parser. Hence the coverage number reported here should not be taken as an absolute measure of grammar performance.

3.3 Using Feature-Based Models

Besides directly using the dependency backbone of the HPSG output, we could also use it for building feature-based models of statistical dependency parsers. Since we focus on the domain adaptation issue, we incorporate a less domain dependent language resource (i.e. the HPSG parsing outputs using ERG) into the features models of statistical parsers. As mordern grammar-based parsers has achieved high runtime efficency (with our HPSG parser parsing at an average speed of $\sim$3 sentences per second), this adds up to an acceptable overhead.

3.3.1 Feature Model with MSTParser

As mentioned before, MSTParser is a graph-based statistical dependency parser, whose learning procedure can be viewed as the assignment of different weights to all kinds of dependency arcs. Therefore, the feature model focuses on each kind of head-child pair in the dependency tree, and mainly contains four categories of features (Mcdonald et al., 2005a): basic uni-gram features, basic bi-gram features, in-between POS features, and surrounding POS features. It is emphasized by the authors that the last two categories contribute a large improvement to the performance and bring the parser to the state-of-the-art accuracy.

Therefore, we extend this feature set by adding four more feature categories, which are similar to the original ones, but the dependency relation was replaced by the dependency backbone of the HPSG outputs. The extended feature set is shown in Table 1.

3.3.2 Feature Model with MaltParser

MaltParser is another trend of dependency parser, which is based on transitions. The learning procedure is to train a statistical model, which can help the parser to decide which operation to take at each parsing status. The basic data structures are a stack, where the constructed dependency graph is stored, and an input queue, where the unprocessed data are put. Therefore, the feature model focuses on the tokens close to the top of the stack and also the head of the queue.

Provided with the original features used in MaltParser, we add extra ones about the top token in the stack and the head token of the queue derived from the HPSG dependency backbone. The extended feature set is shown in Table 2 (the new features are listed separately).

Uni-gram Features: *h-w,h-p; h-w; h-p; c-w,c-p; c-w; c-p*
Bi-gram Features: *h-w,h-p,c-w,c-p; h-p,c-w,c-p; h-w,c-w,c-p; h-w,h-p,c-p; h-w,h-p,c-w; h-w,c-w; h-p,c-p*
POS Features of words in between: *h-p,b-p,c-p*
POS Features of words surround: *h-p,h-p+1,c-p-1,c-p; h-p-1,h-p,c-p-1,c-p; h-p,h-p+1,c-p,c-p+1; h-p-1,h-p,c-p,c-p+1*

Table 1: The Extra Feature Set for MSTParser. *h*: the HPSG head of the current token; *c*: the current token; *b*: each token in between; *-1/+1*: the previous/next token; *w*: word form; *p*: POS

POS Features: *s[0]-p; s[1]-p; i[0]-p; i[1]-p; i[2]-p; i[3]-p*
Word Form Features: *s[0]-h-w; s[0]-w; i[0]-w; i[1]-w*
Dependency Features: *s[0]-lmc-d; s[0]-d; s[0]-rmc-d; i[0]-lmc-d*
New Features: *s[0]-hh-w; s[0]-hh-p; s[0]-hr; i[0]-hh-w; i[0]-hh-p; i[0]-hr*

Table 2: The Extended Feature Set for MaltParser. *s[0]/s[1]*: the first and second token on the top of the stack; *i[0]/i[1]/i[2]/i[3]*: front tokens in the input queue; *h*: head of the token; *hh*: HPSG DB head of the token; *w*: word form; *p*: POS; *d*: dependency relation; *hr*: HPSG rule; *lmc/rmc*: left-/right-most child

With the extra features, we hope that the training of the statistical model will not overfit the in-domain data, but be able to deal with domain independent linguistic phenomena as well.

4 Experiment Results & Error Analyses

To evaluate the performance of our different dependency parsing models, we tested our approaches on several dependency treebanks for English in a similar spirit to the CoNLL 2006-2008 Shared Tasks. In this section, we will first describe the datasets, then present the results. An error analysis is also carried out to show both pros and cons of different models.

4.1 Datasets

In previous years of CoNLL Shared Tasks, several datasets have been created for the purpose of dependency parser evaluation. Most of them are converted automatically from existing treebanks in various forms. Our experiments adhere to the CoNLL 2008 dependency syntax (Yamada et al. 2003, Johansson et al. 2007) which was used to convert Penn-Treebank constituent trees into single-head, single-root, traceless and non-projective dependencies.

WSJ This dataset comprises of three portions. The larger part is converted from the Penn Treebank Wall Street Journal Sections #2–#21, and is used for training statistical dependency parsing models; the smaller part, which covers sentences from Section #23, is used for testing.

Brown This dataset contains a subset of converted sentences from BROWN sections of the Penn Treebank. It is used for the out-domain test.

PChemtb This dataset was extracted from the PennBioIE CYP corpus, containing 195 sentences from biomedical domain. The same dataset has been used for the domain adaptation track of the CoNLL 2007 Shared Task. Although the original annotation scheme is similar to the Penn Treebank, the dependency extraction setting is slightly different to the CoNLLWSJ dependencies (e.g. the coordinations).

Childes This is another out-domain test set from the children language component of the TalkBank, containing dialogs between parents and children. This is the other datasets used in the domain adaptation track of the CoNLL 2007 Shared Task. The dataset is annotated with unlabeled dependencies. As have been reported by others, several systematic differences in the original CHILDES annotation scheme has led to the poor system performances on this track of the Shared Task in 2007. Two main differences concern a) root attachments, and b) coordinations. With several simple heuristics, we change the annotation scheme of the original dataset to match the Penn Treebank-based datasets. The new dataset is referred to as CHILDES*.

4.2 HPSG Backbone as Dependency Parser

First we test the agreement between HPSG dependency backbone and CoNLL dependency. While approximating a target dependency structure with rule-based conversion is not the main focus of this work, the agreement between two representations gives indication on how similar and consistent the two representations are, and a rough impression of whether the feature-based models can benefit from the HPSG backbone.

	# sentence	ϕ w/s	DB(F)%	DB(P)%
WSJ	2399	24.04	50.68	63.85
BROWN	425	16.96	66.36	76.25
PCHEMTB	195	25.65	50.27	61.60
CHILDES*	666	7.51	67.37	70.66
WSJ-P	1796 (75%)	22.25	71.33	–
BROWN-P	375 (88%)	15.74	80.04	–
PCHEMTB-P	147 (75%)	23.99	69.27	–
CHILDES*-P	595 (89%)	7.49	73.91	–

Table 3: Agreement between HPSG dependency backbone and CoNLL 2008 dependency in unlabeled attachment score. DB(F): full parsing mode; DB(P): partial parsing mode; Punctuations are excluded from the evaluation.

The PET parser, an efficient parser HPSG parser is used in combination with ERG to parse the test sets. Note that the training set is not used. The grammar is not adapted for any of these specific domain. To pick the most probable reading from HPSG parsing outputs, we used a discriminative parse selection model as described in (Toutanova et al., 2002) trained on the LOGON Treebank (Oepen et al., 2004), which is significantly different from any of the test domain. The treebank contains about 9K sentences for which HPSG analyses are manually disambiguated. The difference in annotation make it difficult to simply merge this HPSG treebank into the training set of the dependency parser. Also, as Gildea (2001) suggests, adding such heterogeneous data to the training set will not automatically lead to performance improvement. It should be noted that domain adaptation also presents a challenge to the disambiguation model of the HPSG parser. All datasets we use in our should be considered out-domain to the HPSG disambiguation model.

Table 3 shows the agreement between the HPSG backbone and CoNLL dependency in unlabeled attachment score (UAS). The parser is set in either full parsing or partial parsing mode. Partial parsing is used as a fallback when full parse is not available. UAS are reported on all complete test sets, as well as fully parsed subsets (suffixed with *"-p"*).

It is not surprising to see that, without a decent fallback strategy, the full parse HPSG backbone suffers from insufficient coverage. Since the grammar coverage is statistically correlated to the average sentence length, the worst performance is observed for the PCHEMTB. Although sentences in CHILDES* are significantly shorter than those in BROWN, there is a fairly large amount of less well-formed sentences (either as a nature of child language, or due to the transcription from spoken dialogs). This leads to the close performance between these two datasets. PCHEMTB appears to be the most difficult one for the HPSG parser. The partial parsing fallback sets up a good safe net for sentences that fail to parse. Without resorting to any external resource, the performance was significantly improved on all complete test sets.

When we set the coverage of the HPSG grammar aside and only compare performance on the subsets of these datasets which are fully parsed by the HPSG grammar, the unlabeled attachment score jumps up significantly. Most notable is that the dependency backbone achieved over 80% UAS on BROWN, which is close to the performance of state-of-the-art statistical dependency parsing systems trained on WSJ (see Table 5 and Table 4). The performance difference across data sets correlates to varying levels of difficulties in linguists' view. Our error analysis does confirm that frequent errors occur in WSJ test with financial terminology missing from the grammar lexicon. The relative performance difference between the WSJ and BROWN test is contrary to the results observed for statistical parsers trained on WSJ.

To further investigate the effect of HPSG parse disambiguation model on the dependency backbone accuracy, we used a set of 222 sentences from section of WSJ which have been parsed with ERG and manually disambiguated. Comparing to the WSJ-P result in Table 3, we improved the agreement with CoNLL dependency by another 8% (an upper-bound in case of a perfect disambiguation model).

4.3 Statistical Dependency Parsing with HPSG Features

Similar evaluations were carried out for the statistical parsers using extra HPSG dependency backbone as features. It should be noted that the performance comparison between MSTParser and MaltParser is not the aim of this experiment, and the difference might be introduced by the specific settings we use for each parser. Instead, performance variance using different feature models is the main subject. Also, performance drop on out-domain tests shows how domain dependent the feature models are.

For MaltParser, we use Arc-Eager algo-

rithm, and polynomial kernel with $d = 2$. For MSTParser, we use 1st order features and a projective decoder (Eisner, 1996).

When incorporating HPSG features, two settings are used. The PARTIAL model is derived by robust-parsing the entire training data set and extract features from every sentence to train a unified model. When testing, the PARTIAL model is used alone to determine the dependency structures of the input sentences. The FULL model, on the other hand is only trained on the full parsed subset of sentences, and only used to predict dependency structures for sentences that the grammar parses. For the unparsed sentences, the original models without HPSG features are used.

Parser performances are measured using both labeled and unlabeled attachment scores (LAS/UAS). For unlabeled CHILDES* data, only UAS numbers are reported. Table 4 and 5 summarize results for MSTParser and MaltParser, respectively.

With both parsers, we see slight performance drops with both HPSG feature models on in-domain tests (WSJ), compared with the original models. However, on out-domain tests, full-parse HPSG feature models consistently outperform the original models for both parsers. The difference is even larger when only the HPSG fully parsed subsets of the test sets are concerned. When we look at the performance difference between in-domain and out-domain tests for each feature model, we observe that the drop is significantly smaller for the extended models with HPSG features.

We should note that we have not done any feature selection for our HPSG feature models. Nor have we used the best known configurations of the existing parsers (e.g. second order features in MSTParser). Admittedly the results on PCHEMTB are lower than the best reported results in CoNLL 2007 Shared Task, we shall note that we are not using any in-domain unlabeled data. Also, the poor performance of the HPSG parser on this dataset indicates that the parser performance drop is more related to domain-specific phenomena and not general linguistic knowledge. Nevertheless, the drops when compared to in-domain tests are constantly decreased with the help of HPSG analyses features. With the results on BROWN, the performance of our HPSG feature models will rank 2^{nd} on the out-domain test for the CoNLL 2008 Shared Task.

Unlike the observations in Section 4.2, the partial parsing mode does not work well as a fall-back in the feature models. In most cases, its performances are between the original models and the full-parse HPSG feature models. The partial parsing features obscure the linguistic certainty of grammatical structures produced in the full model. When used as features, such uncertainty leads to further confusion. Practically, falling back to the original models works better when HPSG full parse is not available.

4.4 Error Analyses

Qualitative error analysis is also performed. Since our work focuses on the domain adaptation, we manually compare the outputs of the original statistical models, the dependency backbone, and the feature-based models on the out-domain data, i.e. the BROWN data set (both labeled and unlabeled results) and the CHILDES* data set (only unlabeled results).

For the dependency attachment (i.e. unlabeled dependency relation), fine-grained HPSG features do help the parser to deal with colloquial sentences, such as "What's wrong with you?". The original parser wrongly takes "what" as the root of the dependency tree and "'s" is attached to "what". The dependency backbone correctly finds out the root, and thus guide the extended model to make the right prediction. A correct structure of "..., were now neither active nor really relaxed." is also predicted by our model, while the original model wrongly attaches "really" to "nor" and "relaxed" to "were". The rich linguistic knowledge from the HPSG outputs also shows its usefulness. For example, in a sentence from the CHILDES* data, "Did you put dolly's shoes on?", the verb phrase "put on" can be captured by the HPSG backbone, while the original model attaches "on" to the adjacent token "shoes".

For the dependency labels, the most difficulty comes from the prepositions. For example, "Scotty drove home alone in the Plymouth", all the systems get the head of "in" correct, which is "drove". However, none of the dependency labels is correct. The original model predicts the "DIR" relation, the extended feature-based model says "TMP", but the gold standard annotation is "LOC". This is because the HPSG dependency backbone knows that "in the Plymouth" is an adjunct of "drove", but whether it is a temporal or

	Original		PARTIAL		FULL	
	LAS%	UAS%	LAS%	UAS%	LAS%	UAS%
WSJ	87.38	90.35	87.06	90.03	86.87	89.91
BROWN	80.46 (-6.92)	86.26 (-4.09)	80.55 (-6.51)	86.17 (-3.86)	**80.92 (-5.95)**	**86.58 (-3.33)**
PCHEMTB	53.37 (-33.8)	62.11 (-28.24)	54.69 (-32.37)	64.09 (-25.94)	56.45 (-30.42)	65.77 (-24.14)
CHILDES*	–	72.17 (-18.18)	–	74.91 (-15.12)	–	**75.64 (-14.27)**
WSJ-P	87.86	90.88	87.78	90.85	87.12	90.25
BROWN-P	81.58 (-6.28)	87.41 (-3.47)	81.92 (-5.86)	87.51 (-3.34)	**82.14 (-4.98)**	**87.80 (-2.45)**
PCHEMTB-P	56.32 (-31.54)	65.26 (-25.63)	59.36 (-28.42)	69.20 (-21.65)	60.69 (-26.43)	70.45 (-19.80)
CHILDES*-P	–	72.88 (-18.00)	–	76.02 (-14.83)	–	**76.76 (-13.49)**

Table 4: Performance of the MSTParser with different feature models. Numbers in parentheses are performance drops in out-domain tests, comparing to in-domain results. The upper part represents the results on the complete data sets, and the lower part is on the fully parsed subsets, indicated by "-P".

	Original		PARTIAL		FULL	
	LAS%	UAS%	LAS%	UAS%	LAS%	UAS%
WSJ	86.47	88.97	85.39	88.10	85.66	88.40
BROWN	79.41 (-7.06)	84.75 (-4.22)	79.10 (-6.29)	84.58 (-3.52)	**79.56 (-6.10)**	**85.24 (-3.16)**
PCHEMTB	61.05 (-25.42)	71.32 (-17.65)	61.01 (-24.38)	70.99 (-17.11)	60.93 (-24.73)	70.89 (-17.51)
CHILDES*	–	74.97 (-14.00)	–	75.64 (-12.46)	–	**76.18 (-12.22)**
WSJ-P	86.99	89.58	86.09	88.83	85.82	88.76
BROWN-P	80.43 (-6.56)	85.78 (-3.80)	80.46 (-5.63)	85.94 (-2.89)	**80.62 (-5.20)**	**86.38 (-2.38)**
PCHEMTB-P	63.33 (-23.66)	73.54 (-16.04)	63.27 (-22.82)	73.31 (-15.52)	63.16 (-22.66)	73.06 (-15.70)
CHILDES*-P	–	75.95 (-13.63)	–	77.05 (-11.78)	–	**77.30 (-11.46)**

Table 5: Performance of the MaltParser with different feature models.

locative expression cannot be easily predicted at the pure syntactic level. This also suggests a joint learning of syntactic and semantic dependencies, as proposed in the CoNLL 2008 Shared Task.

Instances of wrong HPSG analyses have also been observed as one source of errors. For most of the cases, a correct reading exists, but not picked by our parse selection model. This happens more often with the WSJ test set, partially contributing to the low performance.

5 Conclusion & Future Work

Similar to our work, Sagae et al. (2007) also considered the combination of dependency parsing with an HPSG parser, although their work was to use statistical dependency parser outputs as soft constraints to improve the HPSG parsing. Nevertheless, a similar backbone extraction algorithm was used to map between different representations. Similar work also exists in the constituent-based approaches, where CFG backbones were used to improve the efficiency and robustness of HPSG parsers (Matsuzaki et al., 2007; Zhang and Kordoni, 2008).

In this paper, we restricted our investigation on the syntactic evaluation using labeled/unlabeled attachment scores. Recent discussions in the parsing community about meaningful cross-framework evaluation metrics have suggested to use measures that are semantically informed. In this spirit, Zhang et al. (2008) showed that the semantic outputs of the same HPSG parser helps in the semantic role labeling task. Consistent with the results reported in this paper, more improvement was achieved on the out-domain tests in their work as well.

Although the experiments presented in this paper were carried out on a HPSG grammar for English, the method can be easily adapted to work with other grammar frameworks (e.g. LFG, CCG, TAG, etc.), as well as on langugages other than English. We chose to use a hand-crafted grammar, so that the effect of training corpus on the deep parser is minimized (with the exception of the lexical coverage and disambiguation model).

As mentioned in Section 4.4, the performance of our HPSG parse selection model varies across different domains. This indicates that, although the deep grammar embraces domain independent linguistic knowledge, the lexical coverage and the disambiguation process among permissible readings is still domain dependent. With the mapping between HPSG analyses and their dependency backbones, one can potentially use existing dependency treebanks to help overcome the insufficient data problem for deep parse selection models.

References

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. Map adaptation of stochastic grammars. *Computer speech and language*, 20(1):41–68.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X)*, New York City, USA.

Stephen Clark and James Curran. 2007. Formalism-independent parser evaluation with ccg and depbank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 248–255, Prague, Czech Republic.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 340–345, Copenhagen, Denmark.

Dan Flickinger. 2002. On building a more efficient grammar by exploiting types. In Stephan Oepen, Dan Flickinger, Jun'ichi Tsujii, and Hans Uszkoreit, editors, *Collaborative Language Engineering*, pages 1–17. CSLI Publications.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 167–202, Pittsburgh, USA.

Walter Kasper, Bernd Kiefer, Hans-Ulrich Krieger, C.J. Rupp, and Karsten Worm. 1999. Charting the depths of robust speech processing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL 1999)*, pages 405–412, Maryland, USA.

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

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Efficient HPSG parsing with supertagging and CFG-filtering. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 1671–1676, Hyderabad, India.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 337–344, Sydney, Australia.

David McClosky, Eugene Charniak, and Mark Johnson. 2008. When is self-training effective for parsing? In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 561–568, Manchester, UK.

Ryan Mcdonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98, Ann Arbor, Michigan.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of HLT-EMNLP 2005*, pages 523–530, Vancouver, Canada.

Yusuke Miyao, Kenji Sagae, and Jun'ichi Tsujii. 2007. Towards framework-independent evaluation of deep linguistic parsers. In *Proceedings of the GEAF07 Workshop*, pages 238–258, Stanford, CA.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio, June.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007a. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic.

Joakim Nivre, Jens Nilsson, Johan Hall, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007b. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(1):1–41.

Stephan Oepen, Helge Dyvik, Jan Tore Lønning, Erik Velldal, Dorothee Beermann, John Carroll, Dan Flickinger, Lars Hellan, Janne Bondi Johannessen, Paul Meurer, Torbjørn Nordgård, and Victoria Rosén. 2004. Som å kapp-ete med trollet? Towards MRS-Based Norwegian–English Machine Translation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation*, Baltimore, USA.

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

Kenji Sagae, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Hpsg parsing with shallow dependency constraints. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 624–631, Prague, Czech Republic.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*, Manchester, UK.

Kristina Toutanova, Christoper D. Manning, Stuart M. Shieber, Dan Flickinger, and Stephan Oepen. 2002. Parse ranking for a rich HPSG grammar. In *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories (TLT 2002)*, pages 253–263, Sozopol, Bulgaria.

Yi Zhang and Valia Kordoni. 2008. Robust Parsing with a Large HPSG Grammar. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

Yi Zhang, Rui Wang, and Hans Uszkoreit. 2008. Hybrid Learning of Dependency Structures from Heterogeneous Linguistic Resources. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL 2008)*, pages 198–202, Manchester, UK.

A Chinese-English Organization Name Translation System Using Heuristic Web Mining and Asymmetric Alignment

Fan Yang, Jun Zhao, Kang Liu
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
{fyang,jzhao,kliu}@nlpr.ia.ac.cn

Abstract

In this paper, we propose a novel system for translating organization names from Chinese to English with the assistance of web resources. Firstly, we adopt a chunking-based segmentation method to improve the segmentation of Chinese organization names which is plagued by the OOV problem. Then a heuristic query construction method is employed to construct an efficient query which can be used to search the bilingual Web pages containing translation equivalents. Finally, we align the Chinese organization name with English sentences using the asymmetric alignment method to find the best English fragment as the translation equivalent. The experimental results show that the proposed method outperforms the baseline statistical machine translation system by 30.42%.

1 Introduction

The task of Named Entity (NE) translation is to translate a named entity from the source language to the target language, which plays an important role in machine translation and cross-language information retrieval (CLIR). The organization name (ON) translation is the most difficult subtask in NE translation. The structure of ON is complex and usually nested, including person name, location name and sub-ON etc. For example, the organization name "北京诺基亚通信有限公司 (Beijing Nokia Communication Ltd.)" contains a company name (诺基亚/Nokia) and a location name (北京/Beijing). Therefore, the translation of organization names should combine transliteration and translation together.

Many previous researchers have tried to solve ON translation problem by building a statistical model or with the assistance of web resources. The performance of ON translation using web knowledge is determined by the solution of the following two problems:

- *The efficiency of web page searching*: how can we find the web pages which contain the translation equivalent when the amount of the returned web pages is limited?
- *The reliability of the extraction method*: how reliably can we extract the translation equivalent from the web pages that we obtained in the searching phase?

For solving these two problems, we propose a Chinese-English organization name translation system using heuristic web mining and asymmetric alignment, which has three innovations.

1) *Chunking-based segmentation*: A Chinese ON is a character sequences, we need to segment it before translation. But the OOV words always make the ON segmentation much more difficult. We adopt a new two-phase method here. First, the Chinese ON is chunked and each chunk is classified into four types. Then, different types of chunks are segmented separately using different strategies. Through chunking the Chinese ON first, the OOVs can be partitioned into one chunk which will not be segmented in the next phase. In this way, the performance of segmentation is improved.

2) *Heuristic Query construction*: We need to obtain the bilingual web pages that contain both the input Chinese ON and its translation equivalent. But in most cases, if we just send the Chinese ON to the search engine, we will always get the Chinese monolingual web pages which don't contain any English word sequences, let alone the English translation equivalent. So we propose a heuristic query construction method to generate an efficient bilingual query. Some words in the Chinese ON are selected and their translations are added into the query. These English words will act as clues for searching

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 387–395,
Suntec, Singapore, 2-7 August 2009.

bilingual web pages. The selection of the Chinese words to be translated will take into consideration both the translation confidence of the words and the information contents that they contain for the whole ON.

3) *Asymmetric alignment*: When we extract the translation equivalent from the web pages, the traditional method should recognize the named entities in the target language sentence first, and then the extracted NEs will be aligned with the source ON. However, the named entity recognition (NER) will always introduce some mistakes. In order to avoid NER mistakes, we propose an asymmetric alignment method which align the Chinese ON with an English sentence directly and then extract the English fragment with the largest alignment score as the equivalent. The asymmetric alignment method can avoid the influence of improper results of NER and generate an explicit matching between the source and the target phrases which can guarantee the precision of alignment.

In order to illustrate the above ideas clearly, we give an example of translating the Chinese ON "中国华融资产管理公司 (China Huarong Asset Management Corporation)".

*Step*1: We first chunk the ON, where "LC", "NC", "MC" and "KC" are the four types of chunks defined in Section 4.2.

中国(China)/LC 华融(Huarong)/NC 资产管理(asset management)/MC 公司(corporation)/KC

*Step*2: We segment the ON based on the chunking results.

中国(china) 华融(Huarong) 资产(asset) 管理(management) 公司(corporation)

If we do not chunk the ON first, the OOV word "华融(Huarong)" may be segmented as "华融". This result will certainly lead to translation errors.

Step 3: Query construction:

We select the words "资产" and "管理" to translate and a bilingual query is constructed as: "中国华融资产管理公司" + asset + management

If we don't add some English words into the query, we may not obtain the web pages which contain the English phrase "*China Huarong Asset Management Corporation*". In that case, we can not extract the translation equivalent.

Step 4: Asymmetric Alignment: We extract a sentence "…*President of China Huarong Asset Management Corporation*…" from the returned snippets. Then the best fragment of the sentence "*China Huarong Asset Management Corporation*" will be extracted as the translation equivalent. We don't need to implement English NER process which may make mistakes.

The remainder of the paper is structured as follows. Section 2 reviews the related works. In Section 3, we present the framework of our system. We discuss the details of the ON chunking in Section 4. In Section 5, we introduce the approach of heuristic query construction. In section 6, we will analyze the asymmetric alignment method. The experiments are reported in Section 7. The last section gives the conclusion and future work.

2 Related Work

In the past few years, researchers have proposed many approaches for organization translation. There are three main types of methods. The first type of methods translates ONs by building a statistical translation model. The model can be built on the granularity of word [Stalls et al., 1998], phrase [Min Zhang et al., 2005] or structure [Yufeng Chen et al., 2007]. The second type of methods finds the translation equivalent based on the results of alignment from the source ON to the target ON [Huang et al., 2003; Feng et al., 2004; Lee et al., 2006]. The ONs are extracted from two corpora. The corpora can be parallel corpora [Moore et al., 2003] or content-aligned corpora [Kumano et al., 2004]. The third type of methods introduces the web resources into ON translation. [Al-Onaizan et al., 2002] uses the web knowledge to assist NE translation and [Huang et al., 2004; Zhang et al., 2005; Chen et al., 2006] extracts the translation equivalents from web pages directly.

The above three types of methods have their advantages and shortcomings. The statistical translation model can give an output for any input. But the performance is not good enough on complex ONs. The method of extracting translation equivalents from bilingual corpora can obtain high-quality translation equivalents. But the quantity of the results depends heavily on the amount and coverage of the corpora. So this kind of method is fit for building a reliable ON dictionary. In the third type of method, with the assistance of web pages, the task of ON translation can be viewed as a two-stage process. Firstly, the web pages that may contain the target translation are found through a search engine. Then the translation equivalent will be extracted from the web pages based on the alignment score with the original ON. This method will not

depend on the quantity and quality of the corpora and can be used for translating complex ONs.

3 The Framework of Our System

The Framework of our ON translation system shown in Figure 1 has four modules.

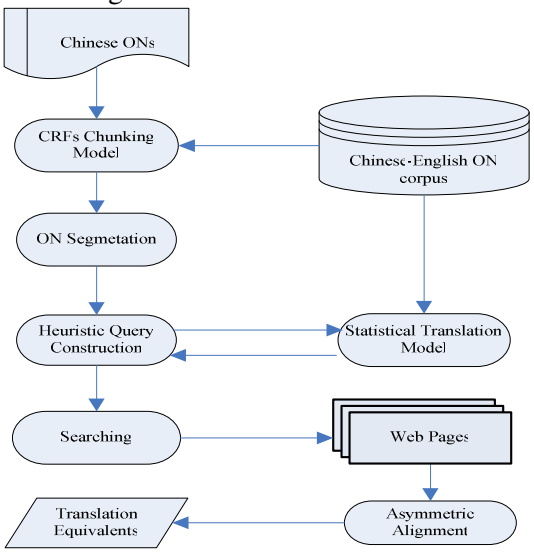


Figure 1. System framework

1) *Chunking-based ON Segmentation Module*: The input of this module is a Chinese ON. The Chunking model will partition the ON into chunks, and label each chunk using one of four classes. Then, different segmentation strategies will be executed for different types of chunks.

2) *Statistical Organization Translation Module*: The input of the module is a word set in which the words are selected from the Chinese ON. The module will output the translation of these words.

3) *Web Retrieval Module*: When input a Chinese ON, this module generates a query which contains both the ON and some words' translation output from the translation module. Then we can obtain the snippets that may contain the translation of the ON from the search engine. The English sentences will be extracted from these snippets.

4) *NE Alignment Module*: In this module, the asymmetric alignment method is employed to align the Chinese ON with these English sentences obtained in Web retrieval module. The best part of the English sentences will be extracted as the translation equivalent.

4 The Chunking-based Segmentation for Chinese ONs

In this section, we will illustrate a chunking-based Chinese ON segmentation method, which can efficiently deal with the ONs containing OOVs.

4.1 The Problems in ON Segmentation

The performance of the statistical ON translation model is dependent on the precision of the Chinese ON segmentation to some extent. When Chinese words are aligned with English words, the mistakes made in Chinese segmentation may result in wrong alignment results. We also need correct segmentation results when decoding. But Chinese ONs usually contain some OOVs that are hard to segment, especially the ONs containing names of people or brand names. To solve this problem, we try to chunk Chinese ONs firstly and the OOVs will be partitioned into one chunk. Then the segmentation will be executed for every chunk except the chunks containing OOVs.

4.2 Four Types of Chunks

We define the following four types of chunks for Chinese ONs:

- *Location Chunk (LC)*: LC contains the location information of an ON.
- *Name Chunk (NC)*: NC contains the name or brand information of an ON. In most cases, Name chunks should be transliterated.
- *Modification Chunk (MC)*: MC contains the modification information of an ON.
- *Key word Chunk (KC)*: KC contains the type information of an ON.

The following is an example of an ON containing these four types of chunks.

北京(Beijing)/LC 百富勤(Peregrine)/NC 投资咨询(investment consulting)/MC 有限公司(co.)/KC

In the above example, the OOV "百富勤(Peregrine)" is partitioned into name chunk. Then the name chunk will not be segmented.

4.3 The CRFs Model for Chunking

Considered as a discriminative probabilistic model for sequence joint labeling and with the advantage of flexible feature fusion ability, Conditional Random Fields (CRFs) [J.Lafferty et al., 2001] is believed to be one of the best probabilistic models for sequence labeling tasks. So the CRFs model is employed for chunking.

We select 6 types of features which are proved to be efficient for chunking through experiments. The templates of features are shown in Table 1,

Description	Features
current/previous/success character	C_0、C_{-1}、C_1
whether the characters is a word	$W(C_{-2}C_{-1}C_0)$、$W(C_0C_1C_2)$、$W(C_{-1}C_0C_1)$
whether the characters is a location name	$L(C_{-2}C_{-1}C_0)$、$L(C_0C_1C_2)$、$L(C_{-1}C_0C_1)$
whether the characters is an ON suffix	$SK(C_{-2}C_{-1}C_0)$、$SK(C_0C_1C_2)$、$SK(C_{-1}C_0C_1)$
whether the characters is a location suffix	$SL(C_{-2}C_{-1}C_0)$、$SL(C_0C_1C_2)$、$SL(C_{-1}C_0C_1)$
relative position in the sentence	$POS(C_0)$

Table 1. Features used in CRFs model

where C_i denotes a Chinese character, i denotes the position relative to the current character. We also use bigram and unigram features but only show trigram templates in Table 1.

5 Heuristic Query Construction

In order to use the web information to assist Chinese-English ON translation, we must firstly retrieve the bilingual web pages effectively. So we should develop a method to construct efficient queries which are used to obtain web pages through the search engine.

5.1 The Limitation of Monolingual Query

We expect to find the web pages where the Chinese ON and its translation equivalent co-occur. If we just use a Chinese ON as the query, we will always obtain the monolingual web pages only containing the Chinese ON. In order to solve the problem, some words in the Chinese ON can be translated into English, and the English words will be added into the query as the clues to search the bilingual web pages.

5.2 The Strategy of Query Construction

We use the metric of precision here to evaluate the possibility in which the translation equivalent is contained in the snippets returned by the search engine. That means, on the condition that we obtain a fixed number of snippets, the more the snippets which contain the translation equivalent are obtained, the higher the precision is. There are two factors to be considered. The first is how efficient the added English words can improve the precision. The second is how to avoid adding wrong translations which may bring down the precision. The first factor means that we should select the most informative words in the Chinese ON. The second factor means that we should consider the confidence of the SMT model at the same time. For example:

天津/LC 本田/NC 摩托车/MC 有限公司/KC
(Tianjin Honda motor co. ltd.)

There are three strategies of constructing queries as follows:

Q1.“天津本田摩托车有限公司” Honda

Q2.“天津本田摩托车有限公司” Ltd.

Q3.“天津本田摩托车有限公司” Motor Tianjin

In the first strategy, we translate the word “本田(Honda)” which is the most informative word in the ON. But its translation confidence is very low, which means that the statistical model gives wrong results usually. The mistakes in translation will mislead the search engine. In the second strategy, we translate the word which has the largest translation confidence. Unfortunately the word is so common that it can’t give any help in filtering out useless web pages. In the third strategy, the words which have sufficient translation confidence and information content are selected.

5.3 Heuristically Selecting the Words to be Translated

The mutual information is used to evaluate the importance of the words in a Chinese ON. We calculate the mutual information on the granularity of words in formula 1 and chunks in formula 2. The integration of the two kinds of mutual information is in formula 3.

$$MIW(x,Y)=\sum_{y\in Y}\log\frac{p(x,y)}{p(x)p(y)} \quad (1)$$

$$MIC(c,Y)=\sum_{y\in Y}\log\frac{p(y,c)}{p(y)p(c)} \quad (2)$$

$$IC(x,Y)=\alpha MIW(x,Y)+(1-\alpha)MIC(c_x,Y) \quad (3)$$

Here, $MIW(x,Y)$ denotes the mutual information of word x with ON Y. That is the summation of the mutual information of x with every word in Y. $MIC(c,Y)$ is similar. c_x denotes the label of the chunk containing x.

We should also consider the risk of obtaining wrong translation results. We can see that the name chunk usually has the largest mutual information. However, the name chunk always needs to be transliterated, and transliteration is often more difficult than translation by lexicon. So we set a threshold T_c for translation confidence. We only select the words whose translation confidences are higher than T_c, with their mutual information from high to low.

6 Asymmetric Alignment Method for Equivalent Extraction

After we have obtained the web pages with the assistant of search engine, we extract the equivalent candidates from the bilingual web pages. So we first extract the pure English sentences and then an asymmetric alignment method is executed to find the best fragment of the English sentences as the equivalent candidate.

6.1 Traditional Alignment Method

To find the translation candidates, the traditional method has three main steps.

1) The NEs in the source and the target language sentences are extracted separately. The NE collections are S_{ne} and T_{ne}.

2) For each NE in S_{ne}, calculate the alignment probability with every NE in T_{ne}.

3) For each NE in S_{ne}, the NE in T_{ne} which has the highest alignment probability will be selected as its translation equivalent.

This method has two main shortcomings:

1) Traditional alignment method needs the NER process in both sides, but the NER process may often bring in some mistakes.

2) Traditional alignment method evaluates the alignment probability coarsely. In other words, we don't know exactly which target word(s) should be aligned to for the source word. A coarse alignment method may have negative effect on translation equivalent extraction.

6.2 The Asymmetric Alignment Method

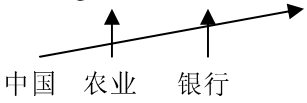
To solve the above two problems, we propose an asymmetric alignment method. The alignment method is so called "asymmetric" for that it aligns a phrase with a sentence, in other words, the alignment is conducted between two objects with different granularities. The NER process is not necessary for that we align the Chinese ON with English sentences directly.

[Wai Lam et al., 2007] proposed a method which uses the KM algorithm to find the optimal explicit matching between a Chinese ON and a given English ON. KM algorithm [Kuhn, 1955] is a traditional graphic algorithm for finding the maximum matching in bipartite weighted graph. In this paper, the KM algorithm is extended to be an asymmetric alignment method. So we can obtain an explicit matching between a Chinese ON and a fragment of English sentence.

A Chinese NE $CO=\{CW_1, CW_2, \ldots, CW_n\}$ is a sequence of Chinese words CW_i and the English sentence $ES=\{EW_1, EW_2, \ldots, EW_m\}$ is a sequence of English words EW_i. Our goal is to find a fragment $EW_{i,i+n}=\{EW_i, \ldots, EW_{i+n}\}$ in ES, which has the highest alignment score with CO. Through executing the extended KM algorithm, we can obtain an explicit matching L. For any CW_i, we can get its corresponding English word EW_j, written as $L(CW_i)=EW_j$ and vice versa. We find the optimal matching L between two phrases, and calculate the alignment score based on L. An example of the asymmetric alignment will be given in Fig2.

Step 1:

… [(The) Agriculture Bank (of) China] (is) (the) four



Step 2:

(The) Agriculture [Bank (of) China] (is) (the) four]…

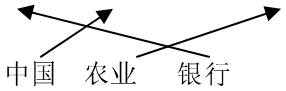


Fig2. An example of asymmetric alignment

In Fig2, the Chinese ON "中国农业银行" is aligned to an English sentence "… the Agriculture Bank of China is the four…". The stop words in parentheses are deleted for they have no meaning in Chinese. In *step* 1, the English fragment contained in the square brackets is aligned with the Chinese ON. We can obtain an explicit matching L_1, shown by arrows, and an alignment score. In *step* 2, the square brackets move right by one word, we can obtain a new matching L_2 and its corresponding alignment score, and so on. When we have calculated every consequent fragment in English sentence, we can find the best fragment "the Agriculture Bank of China" according to the alignment score as the translation equivalent.

The algorithm is shown in Fig3. Where, m is the number of words in an English sentence and n is the number of words in a Chinese ON. KM algorithm will generate an equivalent sub-graph by setting a value to each vertex. The edge whose weight is equal to the summation of the values of its two vertexes will be added into the sub-graph. Then the Hungary algorithm will be executed in the equivalent sub-graph to find the optimal matching. We find the optimal matching between $CW_{1,n}$ and $EW_{1,n}$ first. Then we move the window right and find the optimal matching between $CW_{1,n}$ and $EW_{2,n+1}$. The process will continue until the window arrives at the right most of the

English sentence. When the window moves right, we only need to find a new matching for the new added English vertex EW_{end} and the Chinese vertex C_{drop} which has been matched with EW_{start} in the last step. In the Hungary algorithm, the matching is added through finding an augmenting path. So we only need to find one augmenting path each time. The time complexity of finding an augmenting path is $O(n^3)$. So the whole complexity of asymmetric alignment is $O(m*n^3)$.

Algorithm: Asymmetric Alignment Algorithm

Input: A segmented Chinese ON *CO* and an English sentence ES.
Output: an English fragment $EW_{k,k+n}$

1. Let *start*=1, *end*=*n*, L_0=*null*
2. Using KM algorithm to find the optimal matching between two phrases $CW_{1,n}$ and $EW_{start,end}$ based on the previous matching $L_{start-1}$. We obtain a matching L_{start} and calculate the alignment score S_{start} based on L_{start}.
3. $CW_{drop} = L(EW_{start})$ $L(CW_{drop})$=*null*.
4. If (*end*==*m*) go to 7, else *start*=*start*+1, *end*=*end*+1.
5. Calculate the feasible vertex labeling for the vertexes CW_{drop} and EW_{end}
6. Go to 2.
7. The fragment $EW_{k,k+n-1}$ which has the highest alignment score will be returned.

Fig3. The asymmetric alignment algorithm

6.3 Obtain the Translation Equivalent

For each English sentence, we can obtain a fragment $ES_{i,i+n}$ which has the highest alignment score. We will also take into consideration the frequency information of the fragment and its distance away from the Chinese ON. We use formula (4) to obtain a final score for each translation candidate ET_i and select the largest one as translation result.

$$S(ET_i) = \alpha SA_i + \beta \log(C_i + 1) + \gamma \log(1/D_i + 1) \quad (4)$$

Where C_i denotes the frequency of ET_i, and D_i denotes the nearest distance between ET_i and the Chinese ON.

7 Experiments

We carried out experiments to investigate the performance improvement of ON translation under the assistance of web knowledge.

7.1 Experimental Data

Our experiment data are extracted from LDC2005T34. There are two corpora, ldc_propernames_org_ce_v1.beta (Indus_corpus for short) and ldc_propernames_industry_ce_v1.beta (Org_corpus for short). Some pre-process will be executed to filter out some noisy translation pairs. For example, the translation pairs involving other languages such as Japanese and Korean will be filtered out. There are 65,835 translation pairs that we used as the training corpus and the chunk labels are added manually.

We randomly select 250 translation pairs from the Org_corpus and 253 translation pairs from the Indus_corpus. Altogether, there are 503 translation pairs as the testing set.

7.2 The Effect of Chunking-based Segmentation upon ON Translation

In order to evaluate the influence of segmentation results upon the statistical ON translation system, we compare the results of two translation models. One model uses chunking-based segmentation results as input, while the other uses traditional segmentation results.

To train the CRFs-chunking model, we randomly selected 59,200 pairs of equivalent translations from Indus_corpus and org_corpus. We tested the performance on the set which contains 6,635 Chinese ONs and the results are shown as Table-2.

For constructing a statistical ON translation model, we use *GIZA++*[1] to align the Chinese NEs and the English NEs in the training set. Then the phrase-based machine translation system *MOSES*[2] is adopted to translate the 503 Chinese NEs in testing set into English.

	Precision	Recall	F-measure
LC	0.8083	0.7973	0.8028
NC	0.8962	0.8747	0.8853
MC	0.9104	0.9073	0.9088
KC	0.9844	0.9821	0.9833
All	0.9437	0.9372	0.9404

Table 2. The test results of CRFs-chunking model

We have two metrics to evaluate the translation results. The first metric *L*1 is used to evaluate whether the translation result is exactly the same as the answer. The second metric *L*2 is used to evaluate whether the translation result contains almost the same words as the answer,

[1] http://www.fjoch.com/GIZA++.html
[2] http://www.statmt.org/moses/

without considering the order of words. The results are shown in Table-3:

	chunking-based segmentation	traditional segmentation
*L*1	21.47%	18.29%
*L*2	40.76%	36.78%

Table 3. Comparison of segmentation influence

From the above experimental data, we can see that the chunking-based segmentation improves *L*1 precision from 18.29% to 21.47% and *L*2 precision from 36.78% to 40.76% in comparison with the traditional segmentation method. Because the segmentation results will be used in alignment, the errors will affect the computation of alignment probability. The chunking based segmentation can generate better segmentation results; therefore better alignment probabilities can be obtained.

7.3 The Efficiency of Query Construction

The heuristic query construction method aims to improve the efficiency of Web searching. The performance of searching for translation equivalents mostly depends on how to construct the query. To test its validity, we design four kinds of queries and evaluate their ability using the metric of average precision in formula 5 and macro average precision (MAP) in formula 6,

$$Average\,\mathrm{Precision} = \frac{1}{N}\sum_{i=1}^{N}\frac{H_i}{S_i} \qquad (5)$$

where H_i is the count of snippets that contain at least one equivalent for the *i*th query. And S_i is the total number of snippets we got for the *i*th query,

$$MAP = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{H_j}\sum_{i=1}^{H_j}\frac{i}{R(i)} \qquad (6)$$

where *R*(i) is the order of snippet where the *i*th equivalent occurs. We construct four kinds of queries for the 503 Chinese ONs in testing set as follows:

*Q*1: only the Chinese ON.

*Q*2: the Chinese ON and the results of the statistical translation model.

*Q*3: the Chinese ON and some parts' translation selected by the heuristic query construction method.

*Q*4: the Chinese ON and its correct English translation equivalent.

We obtain at most 100 snippets from *Google* for every query. Sometimes there are not enough snippets as we expect. We set α in formula 4 at 0.7，and the threshold of translation confidence at 0.05. The results are shown as Table 4.

	Average precision	MAP
*Q*1	0.031	0.0527
*Q*2	0.187	0.2061
*Q*3	0.265	0.3129
*Q*4	1.000	1.0000

Table 4. Comparison of four types query

Here we can see that, the result of *Q*4 is the upper bound of the performance, and the *Q*1 is the lower bound of the performance. We concentrate on the comparison between *Q*2 and *Q*3. *Q*2 contains the translations of every word in a Chinese ON, while *Q*3 just contains the translations of the words we select using the heuristic method. *Q*2 may give more information to search engine about which web pages we expect to obtain, but it also brings in translation mistakes that may mislead the search engine. The results show that *Q*3 is better than *Q*2, which proves that a careful clue selection is needed.

7.4 The Effect of Asymmetric Alignment Algorithm

The asymmetric alignment method can avoid the mistakes made in the NER process and give an explicit alignment matching. We will compare the asymmetric alignment algorithm with the traditional alignment method on performance. We adopt two methods to align the Chinese NE with the English sentences. The first method has two phases, the English ONs are extracted from English sentences firstly, and then the English ONs are aligned with the Chinese ON. Lastly, the English ON with the highest alignment score will be selected as the translation equivalent. We use the software *Lingpipe*[3] to recognize NEs in the English sentences. The alignment probability can be calculated as formula 7:

$$Score(C,E) = \sum_i \sum_j p(e_i \mid c_j) \qquad (7)$$

The second method is our asymmetric alignment algorithm. Our method is different from the one in [Wai Lam et al., 2007] which segmented a Chinese ON using an English ON as suggestion. We segment the Chinese ON using the chunking-based segmentation method. The English sentences extracted from snippets will be preprocessed. Some stop words will be deleted, such as "*the*", "*of*", "*on*" etc. To execute the extended KM algorithm for finding the best alignment matching, we must assure that the vertex number in each side of the bipartite is the

[3] http://www.alias-i.com/lingpipe/

same. So we will execute a phrase combination process before alignment, which combines some frequently occurring consequent English words into single vertex, such as "*limited company*" etc. The combination is based on the phrase pair table which is generated from phrase-based SMT system. The results are shown in Table 5:

	Asymmetric Alignment	Traditional method	Statistical model
Top1	**48.71%**	36.18%	**18.29%**
Top5	53.68%	46.12%	**--**

Table 5. Comparison the precision of alignment method

From the results (column 1 and column 2) we can see that, the Asymmetric alignment method outperforms the traditional alignment method. Our method can overcome the mistakes introduced in the NER process. On the other hand, in our asymmetric alignment method, there are two main reasons which may result in mistakes, one is that the correct equivalent doesn't occur in the snippet; the other is that some English ONs can't be aligned to the Chinese ON word by word.

7.5 Comparison between Statistical ON Translation Model and Our Method

Compared with the statistical ON translation model, we can see that the performance is improved from 18.29% to 48.71% (the bold data shown in column 1 and column 3 of Table 5) by using our Chinese-English ON translation system. Transforming the translation problem into the problem of searching for the correct translation equivalent in web pages has three advantages. First, word order determination is difficult in statistical machine translation (SMT), while search engines are insensitive to this problem. Second, SMT often loses some function word such as "*the*", "*a*", "*of*", etc, while our method can avoid this problem because such words are stop words in search engines. Third, SMT often makes mistakes in the selection of synonyms. This problem can be solved by the fuzzy matching of search engines. In summary, web assistant method makes Chinese ON translation easier than traditional SMT method.

8 Conclusion

In this paper, we present a new approach which translates the Chinese ON into English with the assistance of web resources. We first adopt the chunking-based segmentation method to improve the ON segmentation. Then a heuristic query construction method is employed to construct a query which can search translation equivalent more efficiently. At last, the asymmetric alignment method aligns the Chinese ON with English sentences directly. The performance of ON translation is improved from 18.29% to 48.71%. It proves that our system can work well on the Chinese-English ON translation task. In the future, we will try to apply this method in mining the NE translation equivalents from monolingual web pages. In addition, the asymmetric alignment algorithm also has some space to be improved.

Acknowledgement

The work is supported by the National High Technology Development 863 Program of China under Grants no. 2006AA01Z144, and the National Natural Science Foundation of China under Grants no. 60673042 and 60875041.

References

Yaser Al-Onaizan and Kevin Knight. 2002. Translating named entities using monolingual and bilingual resources. In Proc of ACL-2002.

Yufeng Chen, Chenqing Zong. 2007. A Structure-Based Model for Chinese Organization Name Translation. In Proc. of ACM Transactions on Asian Language Information Processing (TALIP)

Donghui Feng, Yajuan Lv, Ming Zhou. 2004. A new approach for English-Chinese named entity alignment. In Proc. of EMNLP 2004.

Fei Huang, Stephan Vogal. 2002. Improved named entity translation and bilingual named entity extraction. In Proc. of the 4th IEEE International Conference on Multimodal Interface.

Fei Huang, Stephan Vogal, Alex Waibel. 2003. Automatic extraction of named entity translingual equivalence based on multi-feature cost minimization. In Proc. of the 2003 Annual Conference of the ACL, Workshop on Multilingual and Mixed-language Named Entity Recognition

Masaaki Nagata, Teruka Saito, and Kenji Suzuki. 2001. Using the Web as a Bilingual Dictionary. In Proc. of ACL 2001 Workshop on Data-driven Methods in Machine Translation.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In Proc. of ACL 2005.

Conrad Chen, Hsin-His Chen. 2006. A High-Accurate Chinese-English NE Backward Translation System Combining Both Lexical Information and Web Statistics. In Proc. of ACL 2006.

Wai Lam, Shing-Kit Chan. 2007. Named Entity Translation Matching and Learning: With Application for Mining Unseen Translations. In Proc. of ACM Transactions on Information Systems.

Chun-Jen Lee, Jason S. Chang, Jyh-Shing R. Jang. 2006. Alignment of bilingual named entities in parallel corpora using statistical models and multiple knowledge sources. In Proc. of ACM Transactions on Asian Language Information Processing (TALIP).

Kuhn, H. 1955. The Hungarian method for the assignment problem. Naval Rese. Logist. Quart 2,83-97.

Min Zhang., Haizhou Li, Su Jian, Hendra Setiawan. 2005. A phrase-based context-dependent joint probability model for named entity translation. In Proc. of the 2nd International Joint Conference on Natural Language Processing(IJCNLP)

Ying Zhang, Fei Huang, Stephan Vogel. 2005. Mining translations of OOV terms from the web through cross-lingual query expansion. In Proc. of the 28th ACM SIGIR.

Bonnie Glover Stalls and Kevin Knight. 1998. Translating names and technical terms in Arabic text. In Proc. of the COLING/ACL Workshop on Computational Approaches to Semitic Language.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proc. ICML-2001.

Tadashi Kumano, Hideki Kashioka, Hideki Tanaka and Takahiro Fukusima. 2004. Acquiring bilingual named entity translations from content-aligned corpora. In Proc. IJCNLP-04.

Robert C. Moore. 2003. Learning translation of named-entity phrases from parallel corpora. In Proc. of 10th conference of the European chapter of ACL.

Reducing semantic drift with bagging and distributional similarity

Tara McIntosh and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{tara,james}@it.usyd.edu.au

Abstract

Iterative bootstrapping algorithms are typically compared using a single set of hand-picked seeds. However, we demonstrate that performance varies greatly depending on these seeds, and favourable seeds for one algorithm can perform very poorly with others, making comparisons unreliable. We exploit this wide variation with bagging, sampling from automatically extracted seeds to reduce semantic drift.

However, semantic drift still occurs in later iterations. We propose an integrated distributional similarity filter to identify and censor potential semantic drifts, ensuring over 10% higher precision when extracting large semantic lexicons.

1 Introduction

Iterative bootstrapping algorithms have been proposed to extract semantic lexicons for NLP tasks with limited linguistic resources. Bootstrapping was initially proposed by Riloff and Jones (1999), and has since been successfully applied to extracting general semantic lexicons (Riloff and Jones, 1999; Thelen and Riloff, 2002), biomedical entities (Yu and Agichtein, 2003), facts (Paşca et al., 2006), and coreference data (Yang and Su, 2007).

Bootstrapping approaches are attractive because they are domain and language independent, require minimal linguistic pre-processing and can be applied to raw text, and are efficient enough for tera-scale extraction (Paşca et al., 2006).

Bootstrapping is minimally supervised, as it is initialised with a small number of seed instances of the information to extract. For semantic lexicons, these seeds are terms from the category of interest. The seeds identify contextual patterns that express a particular semantic category, which in turn recognise new terms (Riloff and Jones, 1999). Unfortunately, *semantic drift* often occurs when ambiguous or erroneous terms and/or patterns are introduced into and then dominate the iterative process (Curran et al., 2007).

Bootstrapping algorithms are typically compared using only a single set of hand-picked seeds. We first show that different seeds cause these algorithms to generate diverse lexicons which vary greatly in precision. This makes evaluation unreliable – seeds which perform well on one algorithm can perform surprisingly poorly on another. In fact, random gold-standard seeds often outperform seeds carefully chosen by domain experts.

Our second contribution exploits this diversity we have identified. We present an unsupervised bagging algorithm which samples from the extracted lexicon rather than relying on existing gazetteers or hand-selected seeds. Each sample is then fed back as seeds to the bootstrapper and the results combined using voting. This both improves the precision of the lexicon and the robustness of the algorithms to the choice of initial seeds.

Unfortunately, semantic drift still dominates in later iterations, since erroneous extracted terms and/or patterns eventually shift the category's direction. Our third contribution focuses on detecting and censoring the terms introduced by semantic drift. We integrate a distributional similarity filter directly into WMEB (McIntosh and Curran, 2008). This filter judges whether a new term is more similar to the earlier or most recently extracted terms, a sign of potential semantic drift.

We demonstrate these methods for extracting biomedical semantic lexicons using two bootstrapping algorithms. Our unsupervised bagging approach outperforms carefully hand-picked seeds by $\sim 10\%$ in later iterations. Our distributional similarity filter gives a similar performance improvement. This allows us to produce large lexicons accurately and efficiently for domain-specific language processing.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 396–404,
Suntec, Singapore, 2-7 August 2009.

2 Background

Hearst (1992) exploited patterns for information extraction, to acquire *is-a* relations using manually devised patterns like *such Z as X and/or Y* where *X* and *Y* are hyponyms of *Z*. Riloff and Jones (1999) extended this with an automated bootstrapping algorithm, *Multi-level Bootstrapping* (MLB), which iteratively extracts semantic lexicons from text.

In MLB, bootstrapping alternates between two stages: pattern extraction and selection, and term extraction and selection. MB is seeded with a small set of user selected *seed* terms. These seeds are used to identify contextual patterns they appear in, which in turn identify new lexicon entries. This process is repeated with the new lexicon terms identifying new patterns. In each iteration, the top-n candidates are selected, based on a metric scoring their membership in the category and suitability for extracting additional terms and patterns.

Bootstrapping eventually extracts polysemous terms and patterns which weakly constrain the semantic class, causing the lexicon's meaning to shift, called *semantic drift* by Curran et al. (2007). For example, female firstnames may drift into flowers when *Iris* and *Rose* are extracted. Many variations on bootstrapping have been developed to reduce semantic drift.[1]

One approach is to extract multiple semantic categories simultaneously, where the individual bootstrapping instances compete with one another in an attempt to actively direct the categories away from each other. Multi-category algorithms outperform MLB (Thelen and Riloff, 2002), and we focus on these algorithms in our experiments.

In BASILISK, MEB, and WMEB, each competing category iterates simultaneously between the term and pattern extraction and selection stages. These algorithms differ in how terms and patterns selected by multiple categories are handled, and their scoring metrics. In BASILISK (Thelen and Riloff, 2002), candidate terms are ranked highly if they have strong evidence for a category and little or no evidence for other categories. This typically favours less frequent terms, as they will match far fewer patterns and are thus more likely to belong to one category. Patterns are selected similarly, however patterns may also be selected by different categories in later iterations.

Curran et al. (2007) introduced *Mutual Exclusion Bootstrapping* (MEB) which forces stricter boundaries between the competing categories than BASILISK. In MEB, the key assumptions are that terms only belong to a category and that patterns only extract terms of a single category. Semantic drift is reduced by eliminating patterns that collide with multiple categories in an iteration and by ignoring colliding candidate terms (for the current iteration). This excludes generic patterns that can occur frequently with multiple categories, and reduces the chance of assigning ambiguous terms to their less dominant sense.

2.1 Weighted MEB

The scoring of candidate terms and patterns in MEB is naïve. Candidates which 1) match the most input instances; and 2) have the potential to generate the most new candidates, are preferred (Curran et al., 2007). This second criterion aims to increase recall. However, the selected instances are highly likely to introduce drift.

Our *Weighted* MEB algorithm (McIntosh and Curran, 2008), extends MEB by incorporating term and pattern weighting, and a cumulative pattern pool. WMEB uses the χ^2 statistic to identify patterns and terms that are strongly associated with the growing lexicon terms and their patterns respectively. The terms and patterns are then ranked first by the number of input instances they match (as in MEB), but then by their weighted score.

In MEB and BASILISK[2], the top-k patterns for each iteration are used to extract new candidate terms. As the lexicons grow, general patterns can drift into the top-k and as a result the earlier precise patterns lose their extracting influence. In WMEB, the pattern pool accumulates all top-k patterns from previous iterations, to ensure previous patterns can contribute.

2.2 Distributional Similarity

Distributional similarity has been used to extract semantic lexicons (Grefenstette, 1994), based on the *distributional hypothesis* that semantically similar words appear in similar contexts (Harris, 1954). Words are represented by context vectors, and words are considered similar if their context vectors are similar.

Patterns and distributional methods have been combined previously. Pantel and Ravichandran

[1] Komachi et al. (2008) used graph-based algorithms to reduce semantic drift for Word Sense Disambiguation.

[2] In BASILISK, k is increased by one in each iteration, to ensure at least one new pattern is introduced.

TYPE (#)	MEDLINE
Terms	1 347 002
Contexts	4 090 412
5-grams	72 796 760
Unfiltered tokens	6 642 802 776

Table 1: Filtered 5-gram dataset statistics.

(2004) used lexical-syntactic patterns to label clusters of distributionally similar terms. Mirkin et al. (2006) used 11 patterns, and the distributional similarity score of each pair of terms, to construct features for lexical entailment. Paşca et al. (2006) used distributional similarity to find similar terms for verifying the names in date-of-birth facts for their tera-scale bootstrapping system.

2.3 Selecting seeds

For the majority of bootstrapping tasks, there is little or no guidance on how to select seeds which will generate the most accurate lexicons. Most previous works used seeds selected based on a user's or domain expert's intuition (Curran et al., 2007), which may then have to meet a frequency criterion (Riloff et al., 2003).

Eisner and Karakos (2005) focus on this issue by considering an approach called *strapping* for word sense disambiguation. In strapping, semi-supervised bootstrapping instances are used to train a meta-classifier, which given a bootstrapping instance can predict the usefulness (*fertility*) of its seeds. The most fertile seeds can then be used in place of hand-picked seeds.

The design of a strapping algorithm is more complex than that of a supervised learner (Eisner and Karakos, 2005), and it is unclear how well strapping will generalise to other bootstrapping tasks. In our work, we build upon bootstrapping using unsupervised approaches.

3 Experimental setup

In our experiments we consider the task of extracting biomedical semantic lexicons from raw text using BASILISK and WMEB.

3.1 Data

We compared the performance of BASILISK and WMEB using 5-grams (t_1, t_2, t_3, t_4, t_5) from raw MEDLINE abstracts[3]. In our experiments, the candidate terms are the middle tokens (t_3), and the patterns are a tuple of the surrounding tokens (t_1, t_2, t_4, t_5). Unlike Riloff and Jones (1999) and Yangarber (2003), we do not use syntactic knowledge, as we aim to take a language independent approach.

[3]The set contains all MEDLINE abstracts available up to Oct 2007 (16 140 000 abstracts).

CAT	DESCRIPTION
ANTI	Antibodies: Immunoglobulin molecules that react with a specific antigen that induced its synthesis *MAb IgG IgM rituximab infliximab* (κ_1:0.89, κ_2:1.0)
CELL	Cells: A morphological or functional form of a cell *RBC HUVEC BAEC VSMC SMC* (κ_1:0.91, κ_2:1.0)
CLNE	Cell lines: A population of cells that are totally derived from a single common ancestor cell *PC12 CHO HeLa Jurkat COS* (κ_1:0.93, κ_2: 1.0)
DISE	Diseases: A definite pathological process that affects humans, animals and or plants *asthma hepatitis tuberculosis HIV malaria* (κ_1:0.98, κ_2:1.0)
DRUG	Drugs: A pharmaceutical preparation *acetylcholine carbachol heparin penicillin tetracyclin* (κ_1:0.86, κ_2:0.99)
FUNC	Molecular functions and processes *kinase ligase acetyltransferase helicase binding* (κ_1:0.87, κ_2:0.99)
MUTN	Mutations: Gene and protein mutations, and mutants *Leiden C677T C282Y 35delG null* (κ_1:0.89, κ_2:1.0)
PROT	Proteins and genes *p53 actin collagen albumin IL-6* (κ_1:0.99, κ_2:1.0)
SIGN	Signs and symptoms of diseases *anemia hypertension hyperglycemia fever cough* (κ_1:0.96, κ_2:0.99)
TUMR	Tumors: Types of tumors *lymphoma sarcoma melanoma neuroblastoma osteosarcoma* (κ_1:0.89, κ_2:0.95)

Table 2: The MEDLINE semantic categories.

The 5-grams were extracted from the MEDLINE abstracts following McIntosh and Curran (2008). The abstracts were tokenised and split into sentences using bio-specific NLP tools (Grover et al., 2006). The 5-grams were filtered to remove patterns appearing with less than 7 terms[4]. The statistics of the resulting dataset are shown in Table 1.

3.2 Semantic Categories

The semantic categories we extract from MEDLINE are shown in Table 2. These are a subset of the TREC Genomics 2007 entities (Hersh et al., 2007). Categories which are predominately multi-term entities, e.g. *Pathways* and *Toxicities*, were excluded.[5] *Genes* and *Proteins* were merged into PROT as they have a high degree of metonymy, particularly out of context. The *Cell or Tissue Type* category was split into two fine grained classes, CELL and CLNE (*cell line*).

[4]This frequency was selected as it resulted in the largest number of patterns and terms loadable by BASILISK

[5]Note that polysemous terms in these categories may be correctly extracted by another category. For example, all *Pathways* also belong to FUNC.

The five hand-picked seeds used for each category are shown in italics in Table 2. These were carefully chosen based on the evaluators' intuition, and are as unambiguous as possible with respect to the other categories.

We also utilised terms in *stop categories* which are known to cause semantic drift in specific classes. These extra categories bound the lexical space and reduce ambiguity (Yangarber, 2003; Curran et al., 2007). We used four stop categories introduced in McIntosh and Curran (2008): AMINO ACID, ANIMAL, BODY and ORGANISM.

3.3 Lexicon evaluation

The evaluation involves manually inspecting each extracted term and judging whether it was a member of the semantic class. This manual evaluation is extremely time consuming and is necessary due to the limited coverage of biomedical resources. To make later evaluations more efficient, all evaluators' decisions for each category are cached.

Unfamiliar terms were checked using online resources including MEDLINE, Medical Subject Headings (MeSH), Wikipedia. Each ambiguous term was counted as correct if it was classified into one of its correct categories, such as *lymphoma* which is a TUMR and DISE. If a term was unambiguously part of a multi-word term we considered it correct. Abbreviations, acronyms and typographical variations were included. We also considered obvious spelling mistakes to be correct, such as *nuetrophils* instead of *neutrophils* (a type of CELL). Non-specific modifiers are marked as incorrect, for example, *gastrointestinal* may be incorrectly extracted for TUMR, as part of the entity *gastrointestinal carcinoma*. However, the modifier may also be used for DISE (*gastrointestinal infection*) and CELL.

The terms were evaluated by two domain experts. Inter-annotator agreement was measured on the top-100 terms extracted by BASILISK and WMEB with the hand-picked seeds for each category. All disagreements were discussed, and the kappa scores, before (κ_1) and after (κ_2) the discussions, are shown in Table 2. Each score is above 0.8 which reflects an agreement strength of "almost perfect" (Landis and Koch, 1977).

For comparing the accuracy of the systems we evaluated the precision of samples of the lexicons extracted for each category. We report average precision over the 10 semantic categories on the 1-200, 401-600 and 801-1000 term samples, and over the first 1000 terms. In each algorithm, each category is initialised with 5 seed terms, and the number of patterns, k, is set to 5. In each iteration, 5 lexicon terms are extracted by each category. Each algorithm is run for 200 iterations.

4 Seed diversity

The first step in bootstrapping is to select a set of seeds by hand. These *hand-picked* seeds are typically chosen by a domain expert who selects a reasonably unambiguous representative sample of the category with high coverage by introspection.

To improve the seeds, the frequency of the potential seeds in the corpora is often considered, on the assumption that highly frequent seeds are better (Thelen and Riloff, 2002). Unfortunately, these seeds may be too general and extract many non-specific patterns. Another approach is to identify seeds using hyponym patterns like, * is a [NAMED ENTITY] (Meij and Katrenko, 2007).

This leads us to our first investigation of seed variability and the methodology used to compare bootstrapping algorithms. Typically algorithms are compared using one set of hand-picked seeds for each category (Pennacchiotti and Pantel, 2006; McIntosh and Curran, 2008). This approach does not provide a fair comparison or any detailed analysis of the algorithms under investigation. As we shall see, it is possible that the seeds achieve the maximum precision for one algorithm and the minimum for another, and thus the single comparison is inappropriate. Even evaluating on multiple categories does not ensure the robustness of the evaluation. Secondly, it provides no insight into the sensitivity of an algorithm to different seeds.

4.1 Analysis with random gold seeds

Our initial analysis investigated the sensitivity and variability of the lexicons generated using different seeds. We instantiated each algorithm 10 times with different random gold seeds (S_{gold}) for each category. We randomly sample S_{gold} from two sets of correct terms extracted from the evaluation cache. UNION: the correct terms extracted by BASILISK and WMEB; and UNIQUE: the correct terms uniquely identified by only one algorithm. The degree of ambiguity of each seed is unknown and term frequency is not considered during the random selection.

Firstly, we investigated the variability of the

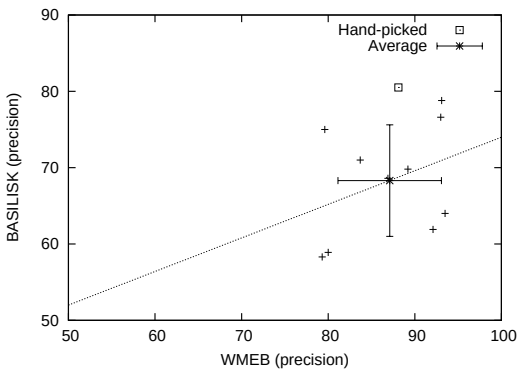


Figure 1: Performance relationship between WMEB and BASILISK on S_{gold} UNION

extracted lexicons using UNION. Each extracted lexicon was compared with the other 9 lexicons for each category and the term overlap calculated. For the top 100 terms, BASILISK had an overlap of 18% and WMEB 44%. For the top 500 terms, BASILISK had an overlap of 39% and WMEB 47%. Clearly BASILISK is far more sensitive to the choice of seeds – this also makes the cache a lot less valuable for the manual evaluation of BASILISK. These results match our annotators' intuition that BASILISK retrieved far more of the esoteric, rare and misspelt results. The overlap between algorithms was even worse: 6.3% for the top 100 terms and 9.1% for the top 500 terms.

The plot in Figure 1 shows the variation in precision between WMEB and BASILISK with the 10 seed sets from UNION. Precision is measured on the first 100 terms and averaged over the 10 categories. The S_{hand} is marked with a square, as well as each algorithms' average precision with 1 standard deviation (S.D.) error bars. The axes start at 50% precision. Visually, the scatter is quite obvious and the S.D. quite large. Note that on our S_{hand} evaluation, BASILISK performed significantly better than average.

We applied a linear regression analysis to identify any correlation between the algorithm's performances. The resulting regression line is shown in Figure 1. The regression analysis identified no correlation between WMEB and BASILISK ($R^2 = 0.13$). It is almost impossible to predict the performance of an algorithm with a given set of seeds from another's performance, and thus comparisons using only one seed set are unreliable.

Table 3 summarises the results on S_{gold}, including the minimum and maximum averages over the 10 categories. At only 100 terms, lexicon

S_{gold}	S_{hand}	Avg.	Min.	Max.	S.D.
UNION					
BASILISK	80.5	68.3	58.3	78.8	7.31
WMEB	88.1	87.1	79.3	93.5	5.97
UNIQUE					
BASILISK	80.5	67.1	56.7	83.5	9.75
WMEB	88.1	91.6	82.4	95.4	3.71

Table 3: Variation in precision with random gold seed sets

variations are already obvious. As noted above, S_{hand} on BASILISK performed better than average, whereas WMEB S_{gold} UNIQUE performed significantly better on average than S_{hand}. This clearly indicates the difficulty of picking the best seeds for an algorithm, and that comparing algorithms with only one set has the potential to penalise an algorithm. These results do show that WMEB is significantly better than BASILISK.

In the UNIQUE experiments, we hypothesized that each algorithm would perform well on its own set, but BASILISK performs significantly worse than WMEB, with a S.D. greater than 9.7. BASILISK's poor performance may be a direct result of it preferring low frequency terms, which are unlikely to be good seeds.

These experiments have identified previously unreported performance variations of these systems and their sensitivity to different seeds. The standard evaluation paradigm, using one set of hand-picked seeds over a few categories, does not provide a robust and informative basis for comparing bootstrapping algorithms.

5 Supervised Bagging

While the wide variation we reported in the previous section is an impediment to reliable evaluation, it presents an opportunity to improve the performance of bootstrapping algorithms. In the next section, we present a novel unsupervised bagging approach to reducing semantic drift. In this section, we consider the standard bagging approach introduced by Breiman (1996). Bagging was used by Ng and Cardie (2003) to create committees of classifiers for labelling unseen data for retraining.

Here, a bootstrapping algorithm is instantiated $n = 50$ times with random seed sets selected from the UNION evaluation cache. This generates n new lexicons $L_1, L_2, \ldots, L_n$ for each category. The next phase involves aggregating the predictions in L_{1-n} to form the final lexicon for each category, using a weighted voting function.

	1-200	401-600	801-1000	1-1000
S_{hand}				
BASILISK	76.3	67.8	58.3	66.7
WMEB	90.3	82.3	62.0	78.6
S_{gold} BAG				
BASILISK	84.2	80.2	58.2	78.2
WMEB	95.1	79.7	65.0	78.6

Table 4: Bagging with 50 gold seed sets

BAGGING	1-200	401-600	801-1000	1-1000
Top-100				
BASILISK	72.3	63.5	58.8	65.1
WMEB	90.2	78.5	66.3	78.5
Top-200				
BASILISK	70.7	60.7	45.5	59.8
WMEB	91.0	78.4	62.2	77.0
Top-500				
BASILISK	63.5	60.5	45.4	56.3
WMEB	92.5	80.9	59.1	77.2
PDF-500				
BASILISK	69.6	68.3	49.6	62.3
WMEB	92.9	80.7	72.1	81.0

Table 5: Bagging with 50 unsupervised seed sets

Our weighting function is based on two related hypotheses of terms in highly accurate lexicons: 1) the more category lexicons in L_{1-n} a term appears in, the more likely the term is a member of the category; 2) terms ranked higher in lexicons are more reliable category members. Firstly, we rank the aggregated terms by the number of lexicons they appear in, and to break ties, we take the term that was extracted in the earliest iteration across the lexicons.

5.1 Supervised results

Table 4 compares the average precisions of the lexicons for BASILISK and WMEB using just the hand-picked seeds (S_{hand}) and 50 sample supervised bagging (S_{gold} BAG).

Bagging with samples from S_{gold} successfully increased the performance of both BASILISK and WMEB in the top 200 terms. While the improvement continued for BASILISK in later sections, it had a more variable effect for WMEB. Overall, BASILISK gets the greater improvement in performance (a 12% gain), almost reaching the performance of WMEB across the top 1000 terms, while WMEB's performance is the same for both S_{hand} and S_{gold} BAG. We believe the greater variability in BASILISK meant it benefited from bagging with gold seeds.

6 Unsupervised bagging

A significant problem for supervised bagging approaches is that they require a larger set of gold-standard seed terms to sample from – either an existing gazetteer or a large hand-picked set. In our case, we used the evaluation cache which took considerable time to accumulate. This saddles the major application of bootstrapping, the quick construction of accurate semantic lexicons, with a chicken-and-egg problem.

However, we propose a novel solution – sampling from the terms extracted with the hand-picked seeds (L_{hand}). WMEB already has very high precision for the top extracted terms (88.1% for the top 100 terms) and may provide an acceptable source of seed terms. This approach now only requires the original 50 hand-picked seed terms across the 10 categories, rather than the 2100 terms used above. The process now uses two rounds of bootstrapping: first to create L_{hand} to sample from and then another round with the 50 sets of randomly unsupervised seeds, S_{rand}.

The next decision is how to sample S_{rand} from L_{hand}. One approach is to use uniform random sampling from restricted sections of L_{hand}. We performed random sampling from the top 100, 200 and 500 terms of L_{hand}. The seeds from the smaller samples will have higher precision, but less diversity.

In a truly unsupervised approach, it is impossible to know if and when semantic drift occurs and thus using arbitrary cut-offs can reduce the diversity of the selected seeds. To increase diversity we also sampled from the top n=500 using a probability density function (PDF) using rejection sampling, where r is the rank of the term in L_{hand}:

$$\text{PDF}(r) = \frac{\sum_{i=r}^{n} i^{-1}}{\sum_{i=1}^{n}\sum_{j=i}^{n} j^{-1}} \quad (1)$$

6.1 Unsupervised results

Table 5 shows the average precision of the lexicons after bagging on the unsupervised seeds, sampled from the top 100 – 500 terms from L_{hand}. Using the top 100 seed sample is much less effective than S_{gold} BAG for BASILISK but nearly as effective for WMEB. As the sample size increases, WMEB steadily improves with the increasing variability, however BASILISK is more effective when the more precise seeds are sampled from higher ranking terms in the lexicons.

Sampling with PDF-500 results in more accurate lexicons over the first 1000 terms than the other

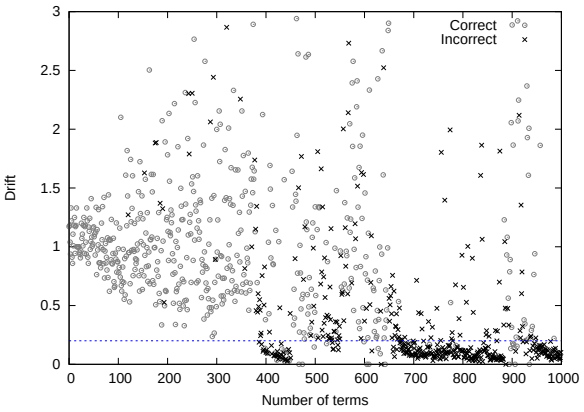


Figure 2: Semantic drift in CELL (n=20, m=20)

sampling methods for WMEB. In particular, WMEB is more accurate with the unsupervised seeds than the S_{gold} and S_{hand} (81.0% vs 78.6% and 78.6%). WMEB benefits from the larger variability introduced by the more diverse sets of seeds, and the greater variability available out-weighs the potential noise from incorrect seeds. The PDF-500 distribution allows some variability whilst still preferring the most reliable unsupervised seeds. In the critical later iterations, WMEB PDF-500 improves over supervised bagging (S_{gold} BAG) by 7% and the original hand-picked seeds (S_{hand}) by 10%.

7 Detecting semantic drift

As shown above, semantic drift still dominates the later iterations of bootstrapping even after bagging. In this section, we propose distributional similarity measurements over the extracted lexicon to detect semantic drift during the bootstrapping process. Our hypothesis is that semantic drift has occurred when a candidate term is more similar to recently added terms than to the seed and high precision terms added in the earlier iterations. We experiment with a range of values of both.

Given a growing lexicon of size N, L_N, let $L_{1...n}$ correspond to the first n terms extracted into L, and $L_{(N-m)...N}$ correspond to the last m terms added to L_N. In an iteration, let t be the next candidate term to be added to the lexicon.

We calculate the average distributional similarity (sim) of t with all terms in $L_{1...n}$ and those in $L_{(N-m)...N}$ and call the ratio the *drift* for term t:

$$\text{drift}(t, n, m) = \frac{\text{sim}(L_{1...n}, t)}{\text{sim}(L_{(N-m)...N}, t)} \quad (2)$$

Smaller values of $\text{drift}(t, n, m)$ correspond to the current term moving further away from the first terms. A $\text{drift}(t, n, m)$ of 0.2 corresponds to a 20% difference in average similarity between $L_{1...n}$ and $L_{(N-m)...N}$ for term t.

Drift can be used as a post-processing step to filter terms that are a possible consequence of drift. However, our main proposal is to incorporate the drift measure directly within the WMEB bootstrapping algorithm, to detect and then prevent drift occuring. In each iteration, the set of candidate terms to be added to the lexicon are scored and ranked for their suitability. We now additionally determine the drift of each candidate term before it is added to the lexicon. If the term's drift is below a specified threshold, it is discarded from the extraction process. If the term has zero similarity with the last m terms, but is similar to at least one of the first n terms, the term is selected. Preventing the drifted term from entering the lexicon during the bootstrapping process, has a flow on effect as it will not be able to extract additional divergent patterns which would lead to accelerated drift.

For calculating drift we use the distributional similarity approach described in Curran (2004). We extracted window-based features from the filtered 5-grams to form context vectors for each term. We used the standard t-test weight and weighted Jaccard measure functions (Curran, 2004). This system produces a distributional score for each pair of terms presented by the bootstrapping system.

7.1 Drift detection results

To evaluate our semantic drift detection we incorporate our process in WMEB. Candidate terms are still weighted in WMEB using the χ^2 statistic as described in (McIntosh and Curran, 2008). Many of the MEDLINE categories suffer from semantic drift in WMEB in the later stages. Figure 2 shows the distribution of correct and incorrect terms appearing in the CELL lexicon extracted using S_{hand} with the term's ranks plotted against their drift scores. Firstly, it is evident that incorrect terms begin to dominate in later iterations. Encouragingly, there is a trend where low values of drift correspond to incorrect terms being added. Drift also occurs in ANTI and MUTN, with an average precision at 801-1000 terms of 41.5% and 33.0% respectively.

We utilise drift in two ways with WMEB; as a post-processing filter (WMEB+POST) and internally during the term selection phase (WMEB+DIST). Table 6 shows the performance

	1-200	401-600	801-1000	1000
WMEB	90.3	82.3	62.0	78.6
WMEB+POST				
n:20 m:5	90.3	82.3	62.1	78.6
n:20 m:20	90.3	81.5	62.0	76.9
n:100 m:5	90.2	82.3	62.1	78.6
n:100 m:20	90.3	82.1	62.1	78.1
WMEB+DIST				
n:20 m:5	90.8	79.7	72.1	80.2
n:20 m:20	90.6	80.1	76.3	81.4
n:100 m:5	90.5	82.0	79.3	82.8
n:100 m:20	90.5	81.5	77.5	81.9

Table 6: Semantic drift detection results

of drift detection with WMEB, using S_{hand}. We use a drift threshold of 0.2 which was selected empirically. A higher value substantially reduced the lexicons' size, while a lower value resulted in little improvements. We experimented with various sizes of initial terms $L_{1...n}$ (n=20, n=100) and $L_{(N-m)...N}$ (m=5, m=20).

There is little performance variation observed in the various WMEB+POST experiments. Overall, WMEB+POST was outperformed slightly by WMEB. The post-filtering removed many incorrect terms, but did not address the underlying drift problem. This only allowed additional incorrect terms to enter the top 1000, resulting in no appreciable difference.

Slight variations in precision are obtained using WMEB+DIST in the first 600 terms, but noticeable gains are achieved in the 801-1000 range. This is not surprising as drift in many categories does not start until later (cf. Figure 2).

With respect to the drift parameters n and m, we found values of n below 20 to be inadequate. We experimented initially with n=5 terms, but this is equivalent to comparing the new candidate terms to the initial seeds. Setting m to 5 was also less useful than a larger sample, unless n was also large. The best performance gain of 4.2% overall for 1000 terms and 17.3% at 801-1000 terms was obtained using n=100 and m=5. In different phases of WMEB+DIST we reduce semantic drift significantly. In particular, at 801-1000, ANTI increase by 46% to 87.5% and MUTN by 59% to 92.0%.

For our final experiments, we report the performance of our best performing WMEB+DIST system (n=100 m=5) using the 10 random GOLD seed sets from section 4.1, in Table 7. On average WMEB+DIST performs above WMEB, especially in the later iterations where the difference is 6.3%.

	S_{hand}	Avg.	Min.	Max.	S.D.
1-200					
WMEB	90.3	82.2	73.3	91.5	6.43
WMEB+DIST	90.7	84.8	78.0	91.0	4.61
401-600					
WMEB	82.3	66.8	61.4	74.5	4.67
WMEB+DIST	82.0	73.1	65.2	79.3	4.52

Table 7: Final accuracy with drift detection

8 Conclusion

In this paper, we have proposed unsupervised bagging and integrated distributional similarity to minimise the problem of semantic drift in iterative bootstrapping algorithms, particularly when extracting large semantic lexicons.

There are a number of avenues that require further examination. Firstly, we would like to take our two-round unsupervised bagging further by performing another iteration of sampling and then bootstrapping, to see if we can get a further improvement. Secondly, we also intend to experiment with machine learning methods for identifying the correct cutoff for the drift score. Finally, we intend to combine the bagging and distributional approaches to further improve the lexicons.

Our initial analysis demonstrated that the output and accuracy of bootstrapping systems can be very sensitive to the choice of seed terms and therefore robust evaluation requires results averaged across randomised seed sets. We exploited this variability to create both supervised and unsupervised bagging algorithms. The latter requires no more seeds than the original algorithm but performs significantly better and more reliably in later iterations. Finally, we incorporated distributional similarity measurements directly into WMEB which detect and censor terms which could lead to semantic drift. This approach significantly outperformed standard WMEB, with a 17.3% improvement over the last 200 terms extracted (801-1000). The result is an efficient, reliable and accurate system for extracting large-scale semantic lexicons.

Acknowledgments

We would like to thank Dr Cassie Thornley, our second evaluator who also helped with the evaluation guidelines; and the anonymous reviewers for their helpful feedback. This work was supported by the CSIRO ICT Centre and the Australian Research Council under Discovery project DP0665973.

References

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 26(2):123–140.

James R. Curran, Tara Murphy, and Bernhard Scholz. 2007. Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, pages 172–180, Melbourne, Australia.

James R. Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh.

Jason Eisner and Damianos Karakos. 2005. Bootstrapping without the boot. In *Proceedings of the Conference on Human Language Technology and Conference on Empirical Methods in Natural Language Processing*, pages 395–402, Vancouver, British Columbia, Canada.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, USA.

Claire Grover, Michael Matthews, and Richard Tobin. 2006. Tools to address the interdependence between tokenisation and standoff annotation. In *Proceedings of the Multi-dimensional Markup in Natural Language Processing Workshop*, Trento, Italy.

Zellig Harris. 1954. Distributional structure. *Word*, 10(2/3):146–162.

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

William Hersh, Aaron M. Cohen, Lynn Ruslen, and Phoebe M. Roberts. 2007. TREC 2007 Genomics Track Overview. In *Proceedings of the 16th Text REtrieval Conference*, Gaithersburg, MD, USA.

Mamoru Komachi, Taku Kudo, Masashi Shimbo, and Yuji Matsumoto. 2008. Graph-based analysis of semantic drift in Espresso-like bootstrapping algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1011–1020, Honolulu, USA.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement in categorical data. *Biometrics*, 33(1):159–174.

Tara McIntosh and James R. Curran. 2008. Weighted mutual exclusion bootstrapping for domain independent lexicon and template acquisition. In *Proceedings of the Australasian Language Technology Association Workshop*, pages 97–105, Hobart, Australia.

Edgar Meij and Sophia Katrenko. 2007. Bootstrapping language associated with biomedical entities. The AID group at TREC Genomics 2007. In *Proceedings of The 16th Text REtrieval Conference*, Gaithersburg, MD, USA.

Shachar Mirkin, Ido Dagan, and Maayan Geffet. 2006. Integrating pattern-based and distributional similarity methods for lexical entailment acquistion. In *Proceedings of the 21st International Conference on Computational Linguisitics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 579–586, Sydney, Australia.

Vincent Ng and Claire Cardie. 2003. Weakly supervised natural language learning without redundant views. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 94–101, Edmonton, USA.

Marius Paşca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. 2006. Names and similarities on the web: Fact extraction in the fast lane. In *Proceedings of the 21st International Conference on Computational Linguisitics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 809–816, Sydney, Australia.

Patrick Pantel and Deepak Ravichandran. 2004. Automatically labelling semantic classes. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 321–328, Boston, MA, USA.

Marco Pennacchiotti and Patrick Pantel. 2006. A bootstrapping algorithm for automatically harvesting semantic relations. In *Proceedings of Inference in Computational Semantics (ICoS-06)*, pages 87–96, Buxton, England.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference*, pages 474–479, Orlando, FL, USA.

Ellen Riloff, Janyce Wiebe, and Theresa Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL-2003)*, pages 25–32.

Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 214–221, Philadelphia, USA.

Xiaofeng Yang and Jian Su. 2007. Coreference resolution using semantic relatedness information from automatically discovered patterns. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 528–535, Prague, Czech Republic.

Roman Yangarber. 2003. Counter-training in discovery of semantic patterns. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 343–350, Sapporo, Japan.

Hong Yu and Eugene Agichtein. 2003. Extracting synonymous gene and protein terms from biological literature. *Bioinformatics*, 19(1):i340–i349.

Jointly Identifying Temporal Relations with Markov Logic

Katsumasa Yoshikawa NAIST, Japan katsumasa-y@is.naist.jp
Sebastian Riedel University of Tokyo, Japan sebastian.riedel@gmail.com
Masayuki Asahara NAIST, Japan masayu-a@is.naist.jp
Yuji Matsumoto NAIST, Japan matsu@is.naist.jp

Abstract

Recent work on temporal relation identification has focused on three types of relations between events: temporal relations between an event and a time expression, between a pair of events and between an event and the document creation time. These types of relations have mostly been identified in isolation by event pairwise comparison. However, this approach neglects logical constraints between temporal relations of different types that we believe to be helpful. We therefore propose a Markov Logic model that jointly identifies relations of all three relation types simultaneously. By evaluating our model on the TempEval data we show that this approach leads to about 2% higher accuracy for all three types of relations —and to the best results for the task when compared to those of other machine learning based systems.

1 Introduction

Temporal relation identification (or temporal ordering) involves the prediction of temporal order between events and/or time expressions mentioned in text, as well as the relation between events in a document and the time at which the document was created.

With the introduction of the TimeBank corpus (Pustejovsky et al., 2003), a set of documents annotated with temporal information, it became possible to apply machine learning to temporal ordering (Boguraev and Ando, 2005; Mani et al., 2006). These tasks have been regarded as essential for complete document understanding and are useful for a wide range of NLP applications such as question answering and machine translation.

Most of these approaches follow a simple schema: they learn classifiers that predict the temporal order of a given event pair based on a set of the pair's of features. This approach is *local* in the sense that only a single temporal relation is considered at a time.

Learning to predict temporal relations in this isolated manner has at least two advantages over any approach that considers several temporal relations jointly. First, it allows us to use off-the-shelf machine learning software that, up until now, has been mostly focused on the case of local classifiers. Second, it is computationally very efficient both in terms of training and testing.

However, the local approach has a inherent drawback: it can lead to solutions that violate logical constraints we know to hold for any sets of temporal relations. For example, by classifying temporal relations in isolation we may predict that event A happened before, and event B after, the time of document creation, but also that event A happened after event B—a clear contradiction in terms of temporal logic.

In order to repair the contradictions that the local classifier predicts, Chambers and Jurafsky (2008) proposed a global framework based on Integer Linear Programming (ILP). They showed that large improvements can be achieved by explicitly incorporating temporal constraints.

The approach we propose in this paper is similar in spirit to that of Chambers and Jurafsky: we seek to improve the accuracy of temporal relation identification by predicting relations in a more global manner. However, while they focused only on the temporal relations between events mentioned in a document, we also jointly predict the temporal order between events and time expressions, and between events and the document creation time.

Our work also differs in another important aspect from the approach of Chambers and Jurafsky. Instead of combining the output of a set of local classifiers using ILP, we approach the problem of joint temporal relation identification using Markov Logic (Richardson and Domingos, 2006). In this

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 405–413,
Suntec, Singapore, 2-7 August 2009.

framework global correlations can be readily captured through the addition of weighted first order logic formulae.

Using Markov Logic instead of an ILP-based approach has at least two advantages. First, it allows us to easily capture non-deterministic (soft) rules that tend to hold between temporal relations but do not have to. [1] For example, if event A happens before B, and B overlaps with C, then there is a good chance that A also happens before C, but this is not guaranteed.

Second, the amount of engineering required to build our system is similar to the efforts required for using an off-the-shelf classifier: we only need to define features (in terms of formulae) and provide input data in the correct format. [2] In particular, we do not need to manually construct ILPs for each document we encounter. Moreover, we can exploit and compare advanced methods of global inference and learning, as long as they are implemented in our Markov Logic interpreter of choice. Hence, in our future work we can focus entirely on temporal relations, as opposed to inference or learning techniques for machine learning.

We evaluate our approach using the data of the "TempEval" challenge held at the SemEval 2007 Workshop (Verhagen et al., 2007). This challenge involved three tasks corresponding to three types of temporal relations: between events and time expressions in a sentence (Task A), between events of a document and the document creation time (Task B), and between events in two consecutive sentences (Task C).

Our findings show that by incorporating global constraints that hold between temporal relations predicted in Tasks A, B and C, the accuracy for all three tasks can be improved significantly. In comparison to other participants of the "TempEval" challenge our approach is very competitive: for two out of the three tasks we achieve the best results reported so far, by a margin of at least 2%. [3] Only for Task B we were unable to reach the performance of a rule-based entry to the challenge. However, we do perform better than all pure machine learning-based entries.

The remainder of this paper is organized as follows: Section 2 describes temporal relation identification including TempEval; Section 3 introduces Markov Logic; Section 4 explains our proposed Markov Logic Network; Section 5 presents the setup of our experiments; Section 6 shows and discusses the results of our experiments; and in Section 7 we conclude and present ideas for future research.

2 Temporal Relation Identification

Temporal relation identification aims to predict the temporal order of events and/or time expressions in documents, as well as their relations to the document creation time (DCT). For example, consider the following (slightly simplified) sentence of Section 1 in this paper.

> With the introduction of the TimeBank corpus (Pustejovsky et al., 2003), machine learning approaches to temporal ordering became possible.

Here we have to predict that the "Machine learning becoming possible" event happened *AFTER* the "introduction of the TimeBank corpus" event, and that it has a temporal *OVERLAP* with the year 2003. Moreover, we need to determine that both events happened *BEFORE* the time this paper was created.

Most previous work on temporal relation identification (Boguraev and Ando, 2005; Mani et al., 2006; Chambers and Jurafsky, 2008) is based on the TimeBank corpus. The temporal relations in the Timebank corpus are divided into 11 classes; 10 of them are defined by the following 5 relations and their inverse: *BEFORE*, *IBEFORE (immediately before)*, *BEGINS*, *ENDS*, *INCLUDES*; the remaining one is *SIMULTANEOUS*.

In order to drive forward research on temporal relation identification, the SemEval 2007 shared task (Verhagen et al., 2007) (TempEval) included the following three tasks.

TASK A Temporal relations between events and time expressions that occur within the same sentence.

TASK B Temporal relations between the Document Creation Time (DCT) and events.

TASK C Temporal relations between the main events of adjacent sentences.[4]

[1] It is clearly possible to incorporate weighted constraints into ILPs, but how to learn the corresponding weights is not obvious.

[2] This is not to say that picking the right formulae in Markov Logic, or features for local classification, is always easy.

[3] To be slightly more precise: for Task C we achieve this margin only for "strict" scoring—see sections 5 and 6 for more details.

[4] The main event of a sentence is expressed by its syntactically dominant verb.

To simplify matters, in the TempEval data, the classes of temporal relations were reduced from the original 11 to 6: *BEFORE, OVERLAP, AFTER, BEFORE-OR-OVERLAP, OVERLAP-OR-AFTER*, and *VAGUE*.

In this work we are focusing on the three tasks of TempEval, and our running hypothesis is that they should be tackled *jointly*. That is, instead of learning separate probabilistic models for each task, we want to learn a single one for all three tasks. This allows us to incorporate rules of temporal consistency that should hold across tasks. For example, if an event X happens *before* DCT, and another event Y *after* DCT, then surely X should have happened *before* Y. We illustrate this type of transition rule in Figure 1.

Note that the correct temporal ordering of events and time expressions can be controversial. For instance, consider the example sentence again. Here one could argue that "the introduction of the TimeBank" may *OVERLAP* with "Machine learning becoming possible" because "introduction" can be understood as a process that is not finished with the release of the data but also includes later advertisements and announcements. This is reflected by the low inter-annotator agreement score of 72% on Tasks A and B, and 68% on Task C.

3 Markov Logic

It has long been clear that local classification alone cannot adequately solve all prediction problems we encounter in practice.[5] This observation motivated a field within machine learning, often referred to as Statistical Relational Learning (SRL), which focuses on the incorporation of global correlations that hold between statistical variables (Getoor and Taskar, 2007).

One particular SRL framework that has recently gained momentum as a platform for global learning and inference in AI is Markov Logic (Richardson and Domingos, 2006), a combination of first-order logic and Markov Networks. It can be understood as a formalism that extends first-order logic to allow formulae that can be violated with some penalty. From an alternative point of view, it is an expressive template language that uses first order logic formulae to instantiate Markov Networks of repetitive structure.

From a wide range of SRL languages we chose Markov Logic because it supports discriminative

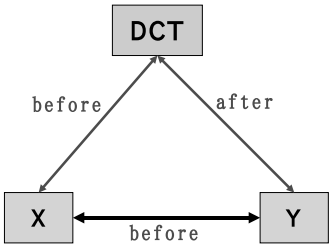


Figure 1: Example of Transition Rule 1

training (as opposed to generative SRL languages such as PRM (Koller, 1999)). Moreover, several Markov Logic software libraries exist and are freely available (as opposed to other discriminative frameworks such as Relational Markov Networks (Taskar et al., 2002)).

In the following we will explain Markov Logic by example. One usually starts out with a set of predicates that model the decisions we need to make. For simplicity, let us assume that we only predict two types of decisions: whether an event e happens before the document creation time (DCT), and whether, for a pair of events e_1 and e_2, e_1 happens before e_2. Here the first type of decision can be modeled through a unary predicate beforeDCT(e), while the latter type can be represented by a binary predicate before(e_1, e_2). Both predicates will be referred to as *hidden* because we do not know their extensions at test time. We also introduce a set of *observed* predicates, representing information that is available at test time. For example, in our case we could introduce a predicate futureTense(e) which indicates that e is an event described in the future tense.

With our predicates defined, we can now go on to incorporate our intuition about the task using weighted first-order logic formulae. For example, it seems reasonable to assume that

$$\text{futureTense}\,(e) \Rightarrow \neg\text{beforeDCT}\,(e) \qquad (1)$$

often, but not always, holds. Our remaining uncertainty with regard to this formula is captured by a weight w we associate with it. Generally we can say that the larger this weight is, the more likely/often the formula holds in the solutions described by our model. Note, however, that we do not need to manually pick these weights; instead they are learned from the given training corpus.

The intuition behind the previous formula can also be captured using a local classifier.[6] However,

[5] It can, however, solve a large number of problems surprisingly well.

[6] Consider a log-linear binary classifier with a "past-tense"

Markov Logic also allows us to say more:

$$\text{beforeDCT}(e_1) \wedge \neg\text{beforeDCT}(e_2) \Rightarrow \text{before}(e_1, e_2) \quad (2)$$

In this case, we made a statement about more global properties of a temporal ordering that cannot be captured with local classifiers. This formula is also an example of the transition rules as seen in Figure 2. This type of rule forms the core idea of our joint approach.

A *Markov Logic Network* (MLN) M is a set of pairs (ϕ, w) where ϕ is a first order formula and w is a real number (the formula's weight). It defines a probability distribution over sets of ground atoms, or so-called *possible worlds*, as follows:

$$p(\mathbf{y}) = \frac{1}{Z} \exp\left(\sum_{(\phi,w)\in M} w \sum_{\mathbf{c}\in C^\phi} f_{\mathbf{c}}^{\phi}(\mathbf{y})\right) \quad (3)$$

Here each $\mathbf{c}$ is a binding of free variables in ϕ to constants in our domain. Each $f_{\mathbf{c}}^{\phi}$ is a binary feature function that returns 1 if in the possible world $\mathbf{y}$ the *ground formula* we get by replacing the free variables in ϕ with the constants in $\mathbf{c}$ is true, and 0 otherwise. C^ϕ is the set of all bindings for the free variables in ϕ. Z is a normalisation constant. Note that this distribution corresponds to a Markov Network (the so-called *Ground Markov Network*) where nodes represent ground atoms and factors represent ground formulae.

Designing formulae is only one part of the game. In practice, we also need to choose a training regime (in order to learn the weights of the formulae we added to the MLN) and a search/inference method that picks the most likely set of ground atoms (temporal relations in our case) given our trained MLN and a set of observations. However, implementations of these methods are often already provided in existing Markov Logic interpreters such as *Alchemy* [7] and *Markov thebeast*. [8]

4 Proposed Markov Logic Network

As stated before, our aim is to jointly tackle Tasks A, B and C of the TempEval challenge. In this section we introduce the Markov Logic Network we designed for this goal.

We have three hidden predicates, corresponding to Tasks A, B, and C: relE2T(e, t, r) represents the temporal relation of class r between an event e and a time expression t; relDCT(e, r) denotes the temporal relation r between an event e and DCT; relE2E$(e1, e2, r)$ represents the relation r between two events of the adjacent sentences, $e1$ and $e2$.

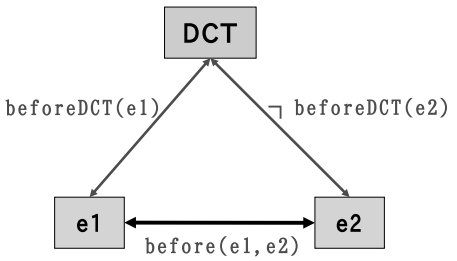


Figure 2: Example of Transition Rule 2

Our observed predicates reflect information we were given (such as the words of a sentence), and additional information we extracted from the corpus (such as POS tags and parse trees). Note that the TempEval data also contained temporal relations that were not supposed to be predicted. These relations are represented using two observed predicates: relT2T$(t1, t2, r)$ for the relation r between two time expressions $t1$ and $t2$; dctOrder(t, r) for the relation r between a time expression t and a fixed DCT.

An illustration of all "temporal" predicates, both hidden and observed, can be seen in Figure 3.

4.1 Local Formula

Our MLN is composed of several weighted formulae that we divide into two classes. The first class contains *local* formulae for the Tasks A, B and C. We say that a formula is local if it only considers the hidden temporal relation of a single event-event, event-time or event-DCT pair. The formulae in the second class are *global*: they involve two or more temporal relations at the same time, and consider Tasks A, B and C simultaneously.

The local formulae are based on features employed in previous work (Cheng et al., 2007; Bethard and Martin, 2007) and are listed in Table 1. What follows is a simple example in order to illustrate how we implement each feature as a formula (or set of formulae).

Consider the tense-feature for Task C. For this feature we first introduce a predicate tense(e, t) that denotes the tense t for an event e. Then we

feature: here for every event e the decision "e happens before DCT" becomes more likely with a higher weight for this feature.

[7] http://alchemy.cs.washington.edu/

[8] http://code.google.com/p/thebeast/

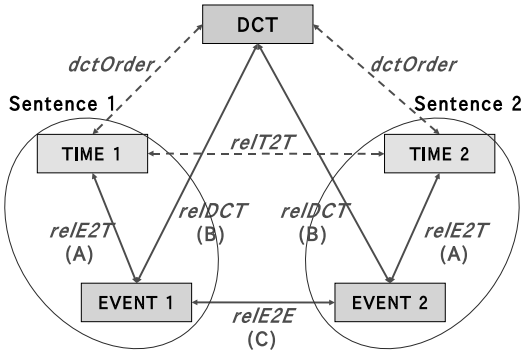


Figure 3: Predicates for Joint Formulae; observed predicates are indicated with dashed lines.

Table 1: Local Features

Feature	A	B	C
EVENT-word	X		X
EVENT-POS	X		X
EVENT-stem	X		X
EVENT-aspect	X	X	X
EVENT-tense	X	X	X
EVENT-class	X	X	X
EVENT-polarity	X		X
TIMEX3-word	X		
TIMEX3-POS	X		
TIMEX3-value	X		
TIMEX3-type	X		
TIMEX3-DCT order	X	X	
positional order	X		
in/outside	X		
unigram(word)	X		X
unigram(POS)	X		X
bigram(POS)	X		
trigram(POS)	X		X
Dependency-Word	X	X	X
Dependency-POS	X	X	

add a set of formulae such as

$$\text{tense}(e1, \text{past}) \wedge \text{tense}(e2, \text{future}) \Rightarrow \text{relE2E}(e1, e2, \text{before}) \quad (4)$$

for all possible combinations of tenses and temporal relations.[9]

4.2 Global Formula

Our global formulae are designed to enforce consistency between the three hidden predicates (and the two observed temporal predicates we mentioned earlier). They are based on the transition rules we mentioned in Section 3.

Table 2 shows the set of formula templates we use to generate the global formulae. Here each template produces several instantiations, one for each assignment of temporal relation classes to the variables R1, R2, etc. One example of a template instantiation is the following formula.

$$\text{dctOrder}(t1, \text{before}) \wedge \text{relDCT}(e1, \text{after}) \Rightarrow \text{relE2T}(e1, t1, \text{after}) \quad (5a)$$

This formula is an expansion of the formula template in the second row of Table 2. Note that it utilizes the results of Task B to solve Task A.

Formula 5a should always hold,[10] and hence we could easily implement it as a hard constraint in an ILP-based framework. However, some transition rules are less determinstic and should rather be taken as "rules of thumb". For example, formula 5b is a rule which we expect to hold often, but not always.

$$\text{dctOrder}(t1, \text{before}) \wedge \text{relDCT}(e1, \text{overlap}) \Rightarrow \text{relE2T}(e1, t1, \text{after}) \quad (5b)$$

Fortunately, this type of soft rule poses no problem for Markov Logic: after training, Formula 5b will simply have a lower weight than Formula 5a. By contrast, in a "Local Classifier + ILP"-based approach as followed by Chambers and Jurafsky (2008) it is less clear how to proceed in the case of soft rules. Surely it is possible to incorporate weighted constraints into ILPs, but how to learn the corresponding weights is not obvious.

5 Experimental Setup

With our experiments we want to answer two questions: (1) does jointly tackling Tasks A, B, and C help to increase overall accuracy of temporal relation identification? (2) How does our approach compare to state-of-the-art results? In the following we will present the experimental set-up we chose to answer these questions.

In our experiments we use the test and training sets provided by the TempEval shared task. We further split the original training data into a training and a development set, used for optimizing parameters and formulae. For brevity we will refer to the training, development and test set as TRAIN, DEV and TEST, respectively. The numbers of temporal relations in TRAIN, DEV, and TEST are summarized in Table 3.

[9] This type of "template-based" formulae generation can be performed automatically by the Markov Logic Engine.

[10] However, due to inconsistent annotations one will find violations of this rule in the TempEval data.

Table 2: Joint Formulae for Global Model

Task	Formula
$A \rightarrow B$	$\text{dctOrder}(t, R1) \wedge \text{relE2T}(e, t, R2) \Rightarrow \text{relDCT}(e, R3)$
$B \rightarrow A$	$\text{dctOrder}(t, R1) \wedge \text{relDCT}(e, R2) \Rightarrow \text{relE2T}(e, t, R3)$
$B \rightarrow C$	$\text{relDCT}(e1, R1) \wedge \text{relDCT}(e2, R2) \Rightarrow \text{relE2E}(e1, e2, R3)$
$C \rightarrow B$	$\text{relDCT}(e1, R1) \wedge \text{relE2E}(e1, e2, R2) \Rightarrow \text{relDCT}(e2, R3)$
$A \rightarrow C$	$\text{relE2T}(e1, t1, R1) \wedge \text{relT2T}(t1, t2, R2) \wedge \text{relE2T}(e2, t2, R3) \Rightarrow \text{relE2E}(e1, e2, R4)$
$C \rightarrow A$	$\text{relE2T}(e2, t2, R1) \wedge \text{relT2T}(t1, t2, R2) \wedge \text{relE2E}(e1, e2, R3) \Rightarrow \text{relE2T}(e1, t1, R4)$

Table 3: Numbers of Labeled Relations for All Tasks

	TRAIN	DEV	TEST	TOTAL
Task A	1359	131	169	1659
Task B	2330	227	331	2888
Task C	1597	147	258	2002

For feature generation we use the following tools. [11] POS tagging is performed with TnT ver2.2;[12] for our dependency-based features we use MaltParser 1.0.0.[13] For inference in our models we use Cutting Plane Inference (Riedel, 2008) with ILP as a base solver. This type of inference is exact and often very fast because it avoids instantiation of the complete Markov Network. For learning we apply one-best MIRA (Crammer and Singer, 2003) with Cutting Plane Inference to find the current model guess. Both training and inference algorithms are provided by *Markov thebeast*, a Markov Logic interpreter tailored for NLP applications.

Note that there are several ways to manually optimize the set of formulae to use. One way is to pick a task and then choose formulae that increase the accuracy for this task on DEV. However, our primary goal is to improve the performance of all the tasks together. Hence we choose formulae with respect to the total score over all three tasks. We will refer to this type of optimization as "averaged optimization". The total scores of the all three tasks are defined as follows:

$$\frac{C_a + C_b + C_c}{G_a + G_b + G_c}$$

where C_a, C_b, and C_c are the number of the correctly identified labels in each task, and G_a, G_b, and G_c are the numbers of gold labels of each task. Our system necessarily outputs one label to one relational link to identify. Therefore, for all our results, precision, recall, and F-measure are the exact same value.

For evaluation, TempEval proposed the two scoring schemes: "strict" and "relaxed". For strict scoring we give full credit if the relations match, and no credit if they do not match. On the other hand, relaxed scoring gives credit for a relation according to Table 4. For example, if a system picks the relation "AFTER" that should have been "BEFORE" according to the gold label, it gets neither "strict" nor "relaxed" credit. But if the system assigns "B-O (BEFORE-OR-OVERLAP)" to the relation, it gets a 0.5 "relaxed" score (and still no "strict" score).

6 Results

In the following we will first present our comparison of the local and global model. We will then go on to put our results into context and compare them to the state-of-the-art.

6.1 Impact of Global Formulae

First, let us show the results on TEST in Table 5. You will find two columns, "Global" and "Local", showing scores achieved with and without joint formulae, respectively. Clearly, the global models scores are higher than the local scores for all three tasks. This is also reflected by the last row of Table 5. Here we see that we have improved the averaged performance across the three tasks by approximately 2.5% ($\rho < 0.01$, McNemar's test 2-tailed). Note that with 3.5% the improvements are particularly large for Task C.

The TempEval test set is relatively small (see Table 3). Hence it is not clear how well our results would generalize in practice. To overcome this issue, we also evaluated the local and global model using 10-fold cross validation on the training data (TRAIN + DEV). The corresponding results can be seen in Table 6. Note that the general picture remains: performance for all tasks is improved, and the averaged score is improved only slightly less than for the TEST results. However, this time the score increase for Task B is lower than before. We

[11] Since the TempEval trial has no restriction on preprocessing such as syntactic parsing, most participants used some sort of parsers.

[12] http://www.coli.uni-saarland.de/~thorsten/tnt/

[13] http://w3.msi.vxu.se/~nivre/research/MaltParser.html

Table 4: Evaluation Weights for Relaxed Scoring

	B	O	A	B-O	O-A	V
B	1	0	0	0.5	0	0.33
O	0	1	0	0.5	0.5	0.33
A	0	0	1	0	0.5	0.33
B-O	0.5	0.5	0	1	0.5	0.67
O-A	0	0.5	0.5	0.5	1	0.67
V	0.33	0.33	0.33	0.67	0.67	1

B: BEFORE O: OVERLAP
A: AFTER B-O: BEFORE-OR-OVERLAP
O-A: OVERLAP-OR-AFTER V: VAGUE

Table 5: Results on TEST Set

	Local		Global	
task	strict	relaxed	strict	relaxed
Task A	0.621	0.669	0.645	0.687
Task B	0.737	0.753	0.758	0.777
Task C	0.531	0.599	0.566	0.632
All	0.641	0.682	0.668	0.708

Table 6: Results with 10-fold Cross Validation

	Local		Global	
task	strict	relaxed	strict	relaxed
Task A	0.613	0.645	0.662	0.691
Task B	0.789	0.810	0.799	0.819
Task C	0.533	0.608	0.552	0.623
All	0.667	0.707	0.689	0.727

see that this is compensated by much higher scores for Task A and C. Again, the improvements for all three tasks are statistically significant ($\rho < 10^{-8}$, McNemar's test, 2-tailed).

To summarize, we have shown that by tightly connecting tasks A, B and C, we can improve temporal relation identification significantly. But are we just improving a weak baseline, or can joint modelling help to reach or improve the state-of-the-art results? We will try to answer this question in the next section.

6.2 Comparison to the State-of-the-art

In order to put our results into context, Table 7 shows them along those of other TempEval participants. In the first row, TempEval Best gives the best scores of TempEval for each task. Note that all but the strict scores of Task C are achieved by WVALI (Puscasu, 2007), a hybrid system that combines machine learning and hand-coded rules. In the second row we see the TempEval average scores of all six participants in TempEval. The third row shows the results of CU-TMP (Bethard and Martin, 2007), an SVM-based system that achieved the second highest scores in TempEval for all three tasks. CU-TMP is of interest because it is the best pure Machine-Learning-based approach so far.

The scores of our local and global model come in the fourth and fifth row, respectively. The last row in the table shows task-adjusted scores. Here we essentially designed and applied three global MLNs, each one tailored and optimized for a different task. Note that the task-adjusted scores are always about 1% higher than those of the single global model.

Let us discuss the results of Table 7 in detail. We see that for task A, our global model improves an already strong local model to reach the best results both for strict scores (with a 3% points margin) and relaxed scores (with a 5% points margin).

For Task C we see a similar picture: here adding global constraints helped to reach the best strict scores, again by a wide margin. We also achieve competitive relaxed scores which are in close range to the TempEval best results.

Only for task B our results cannot reach the best TempEval scores. While we perform slightly better than the second-best system (CU-TMP), and hence report the best scores among all pure Machine-Learning based approaches, we cannot quite compete with WVALI.

6.3 Discussion

Let us discuss some further characteristics and advantages of our approach. First, notice that global formulae not only improve strict but also relaxed scores for all tasks. This suggests that we produce more ambiguous labels (such as BEFORE-OR-OVERLAP) in cases where the local model has been overconfident (and wrongly chose BEFORE or OVERLAP), and hence make less "fatal errors". Intuitively this makes sense: global consistency is easier to achieve if our labels remain ambiguous. For example, a solution that labels every relation as VAGUE is globally consistent (but not very informative).

Secondly, one could argue that our solution to joint temporal relation identification is too complicated. Instead of performing global inference, one could simply arrange local classifiers for the tasks into a pipeline. In fact, this has been done by Bethard and Martin (2007): they first solve task B and then use this information as features for Tasks A and C. While they do report improvements (0.7%

Table 7: Comparison with Other Systems

	Task A		Task B		Task C	
	strict	relaxed	strict	relaxed	strict	relaxed
TempEval Best	0.62	0.64	**0.80**	**0.81**	0.55	**0.64**
TempEval Average	0.56	0.59	0.74	0.75	0.51	0.58
CU-TMP	0.61	0.63	0.75	0.76	0.54	0.58
Local Model	0.62	0.67	0.74	0.75	0.53	0.60
Global Model	**0.65**	**0.69**	0.76	0.78	**0.57**	0.63
Global Model (Task-Adjusted)	(0.66)	(0.70)	(0.76)	(0.79)	(0.58)	(0.64)

on Task A, and about 0.5% on Task C), generally these improvements do not seem as significant as ours. What is more, by design their approach can not improve the first stage (Task B) of the pipeline.

On the same note, we also argue that our approach does not require more implementation efforts than a pipeline. Essentially we only have to provide features (in the form of formulae) to the Markov Logic Engine, just as we have to provide for a SVM or MaxEnt classifier.

Finally, it became more clear to us that there are problems inherent to this task and dataset that we cannot (or only partially) solve using global methods. First, there are inconsistencies in the training data (as reflected by the low inter-annotator agreement) that often mislead the learner—this problem applies to learning of local and global formulae/features alike. Second, the training data is relatively small. Obviously, this makes learning of reliable parameters more difficult, particularly when data is as noisy as in our case. Third, the temporal relations in the TempEval dataset only directly connect a small subset of events. This makes global formulae less effective.[14]

7 Conclusion

In this paper we presented a novel approach to temporal relation identification. Instead of using local classifiers to predict temporal order in a pairwise fashion, our approach uses Markov Logic to incorporate both local features and global transition rules between temporal relations.

We have focused on transition rules between temporal relations of the three TempEval subtasks: temporal ordering of events, of events and time expressions, and of events and the document creation time. Our results have shown that global transition rules lead to significantly higher accuracy for all three tasks. Moreover, our global Markov Logic model achieves the highest scores reported so far for two of three tasks, and very competitive results for the remaining one.

While temporal transition rules can also be captured with an Integer Linear Programming approach (Chambers and Jurafsky, 2008), Markov Logic has at least two advantages. First, handling of "rules of thumb" between less specific temporal relations (such as OVERLAP or VAGUE) is straightforward—we simply let the Markov Logic Engine learn weights for these rules. Second, there is less engineering overhead for us to perform, because we do not need to generate ILPs for each document.

However, potential for further improvements through global approaches seems to be limited by the sparseness and inconsistency of the data. To overcome this problem, we are planning to use external or untagged data along with methods for unsupervised learning in Markov Logic (Poon and Domingos, 2008).

Furthermore, TempEval-2 [15] is planned for 2010 and it has challenging temporal ordering tasks in five languages. So, we would like to investigate the utility of global formulae for multilingual temporal ordering. Here we expect that while lexical and syntax-based features may be quite language dependent, global transition rules should hold across languages.

Acknowledgements

This work is partly supported by the Integrated Database Project, Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

Steven Bethard and James H. Martin. 2007. Cu-tmp: Temporal relation classification using syntactic and semantic features. In *Proceedings of the 4th International Workshop on SemEval-2007.*, pages 129–132.

[14] See (Chambers and Jurafsky, 2008) for a detailed discussion of this problem, and a possible solution for it.

[15] http://www.timeml.org/tempeval2/

Branimir Boguraev and Rie Kubota Ando. 2005. Timeml-compliant text analysis for temporal reasoning. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 997–1003.

Nathanael Chambers and Daniel Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 698–706, Honolulu, Hawaii, October. Association for Computational Linguistics.

Yuchang Cheng, Masayuki Asahara, and Yuji Matsumoto. 2007. Naist.japan: Temporal relation identification using dependency parsed tree. In *Proceedings of the 4th International Workshop on SemEval-2007.*, pages 245–248.

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

Lise Getoor and Ben Taskar. 2007. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Daphne Koller, 1999. *Probabilistic Relational Models*, pages 3–13. Springer, Berlin/Heidelberg, Germany.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 753–760, Morristown, NJ, USA. Association for Computational Linguistics.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with Markov Logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 650–659, Honolulu, Hawaii, October. Association for Computational Linguistics.

Georgiana Puscasu. 2007. Wvali: Temporal relation identification by syntactico-semantic analysis. In *Proceedings of the 4th International Workshop on SemEval-2007.*, pages 484–487.

James Pustejovsky, Jose Castano, Robert Ingria, Reser Sauri, Robert Gaizauskas, Andrea Setzer, and Graham Katz. 2003. The timebank corpus. In *Proceedings of Corpus Linguistics 2003*, pages 647–656.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. In *Machine Learning*.

Sebastian Riedel. 2008. Improving the accuracy and efficiency of map inference for markov logic. In *Proceedings of UAI 2008*.

Ben Taskar, Abbeel Pieter, and Daphne Koller. 2002. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 485–492, San Francisco, CA. Morgan Kaufmann.

Marc Verhagen, Robert Gaizaukas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. Semeval-2007 task 15: Tempeval temporal relation identification. In *Proceedings of the 4th International Workshop on SemEval-2007.*, pages 75–80.

Profile Based Cross-Document Coreference Using Kernelized Fuzzy Relational Clustering

Jian Huang[†] **Sarah M. Taylor**[‡] **Jonathan L. Smith**[‡] **Konstantinos A. Fotiadis**[‡] **C. Lee Giles**[†]
[†]College of Information Sciences and Technology
Pennsylvania State University, University Park, PA 16802, USA
{jhuang, giles}@ist.psu.edu
[‡]Advanced Technology Office, Lockheed Martin IS&GS, Arlington, VA 22203, USA
{sarah.m.taylor, jonathan.l.smith, konstantinos.a.fotiadis}@lmco.com

Abstract

Coreferencing entities across documents in a large corpus enables advanced document understanding tasks such as question answering. This paper presents a novel cross document coreference approach that leverages the profiles of entities which are constructed by using information extraction tools and reconciled by using a within-document coreference module. We propose to match the profiles by using a learned ensemble distance function comprised of a suite of similarity specialists. We develop a kernelized soft relational clustering algorithm that makes use of the learned distance function to partition the entities into fuzzy sets of identities. We compare the kernelized clustering method with a popular fuzzy relation clustering algorithm (FRC) and show 5% improvement in coreference performance. Evaluation of our proposed methods on a large benchmark disambiguation collection shows that they compare favorably with the top runs in the SemEval evaluation.

1 Introduction

A named entity that represents a person, an organization or a geo-location may appear within and across documents in different forms. Cross document coreference (CDC) is the task of consolidating named entities that appear in multiple documents according to their real referents. CDC is a stepping stone for achieving intelligent information access to vast and heterogeneous text corpora, which includes advanced NLP techniques such as document summarization and question answering. A related and well studied task is within document coreference (WDC), which limits the scope of disambiguation to within the boundary of a document. When namesakes appear in an article, the author can explicitly help to disambiguate, using titles and suffixes (as in the example, "George Bush *Sr.* ... the *younger* Bush") besides other means. Cross document coreference, on the other hand, is a more challenging task because these linguistics cues and sentence structures no longer apply, given the wide variety of context and styles in different documents.

Cross document coreference research has recently become more popular due to the increasing interests in the web person search task (Artiles et al., 2007). Here, a search query for a person name is entered into a search engine and the desired outputs are documents clustered according to the identities of the entities in question. In our work, we propose to drill down to the sub-document mention level and construct an entity profile with the support of information extraction tools and reconciled with WDC methods. Hence our IE based approach has access to accurate information such as a person's mentions and geo-locations for disambiguation. Simple IR based CDC approaches (e.g. (Gooi and Allan, 2004)), on the other hand, may simply use all the terms and this can be detrimental to accuracy. For example, a biography of John F. Kennedy is likely to mention members of his family with related positions, besides references to other political figures. Even with careful word selection, these textual features can still confuse the disambiguation system about the true identity of the person.

We propose to handle the CDC task using a novel kernelized fuzzy relational clustering algorithm, which allows probabilistic cluster membership assignment. This not only addresses the intrinsic uncertainty nature of the CDC problem, but also yields additional performance improvement. We propose to use a specialist ensemble

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 414–422,
Suntec, Singapore, 2-7 August 2009.

learning approach to aggregate the diverse set of similarities in comparing attributes and relationships in entity profiles. Our approach is first fully described in Section 2. The effectiveness of the proposed method is demonstrated using real world benchmark test sets in Section 3. We review related work in cross document coreference and conclude in Section 5.

2 Methods

2.1 Document Level and Profile Based CDC

We make distinctions between document level and profile based cross document coreference. *Document level* CDC makes a simplifying assumption that a named entity (and its variants) in a document has one underlying real identity. The assumption is generally acceptable but may be violated when a document refers to namesakes at the same time (e.g. George W. Bush and George H. W. Bush referred to as George or President Bush). Furthermore, the context surrounding the person NE President Clinton can be counterproductive for disambiguating the NE Senator Clinton, with both entities likely to appear in a document at the same time. The simplified document level CDC has nevertheless been used in the WePS evaluation (Artiles et al., 2007), called the web people task.

In this work, we advocate *profile based* disambiguation that aims to leverage the advances in NLP techniques. Rather than treating a document as simply a bag of words, an information extraction tool first extracts NE's and their relationships. For the NE's of interest (i.e. persons in this work), a within-document coreference (WDC) module then links the entities deemed as referring to the same underlying identity into a WDC chain. This process includes both *anaphora resolution* (resolving 'He' and its antecedent 'President Clinton') and *entity tracking* (resolving 'Bill' and 'President Clinton'). Let $\mathcal{E} = \{\mathbf{e}_1, ..., \mathbf{e}_N\}$ denote the set of N chained entities (each corresponding to a WDC chain), provided as input to the CDC system. We intentionally do not distinguish which document each $\mathbf{e}_j$ belongs to, as profile based CDC can potentially rectify WDC errors by leveraging information across document boundaries. Each $\mathbf{e}_i$ is represented as a profile which contains the NE, its attributes and associated relationships, i.e. $\mathbf{e}_j =< e_{j,1}, ..., e_{j,L} >$ ($e_{j,l}$ can be a textual attribute or a pointer to another entity). The profile based CDC method generates a partition of $\mathcal{E}$, represented by a partition matrix U (where u_{ij} denotes the membership of an entity $\mathbf{e}_j$ to the i-th identity cluster). Therefore, the chained entities placed in a name cluster are deemed as coreferent.

Profile based CDC addresses a finer grained coreference problem in the mention level, enabled by the recent advances in IE and WDC techniques. In addition, profile based CDC facilitates user information consumption with structured information and short summary passages. Next, we focus on the relational clustering algorithm that lies at the core of the profile based CDC system. We then turn our attention to the specialist learning algorithm for the distance function used in clustering, capable of leveraging the available training data.

2.2 CDC Using Fuzzy Relational Clustering

2.2.1 Preliminaries

Traditionally, hard clustering algorithms (where $u_{ij} \in \{0, 1\}$) such as complete linkage hierarchical agglomerative clustering (Mann and Yarowsky, 2003) have been applied to the disambiguation problem. In this work, we propose to use fuzzy clustering methods (relaxing the membership condition to $u_{ij} \in [0, 1]$) as a better way of handling uncertainty in cross document coreference. First, consider the following motivating example,

Example. The named entity *President Bush* is extracted from the sentence "President Bush addressed the nation from the Oval Office Monday."

- Without additional cues, a hard clustering algorithm has to arbitrarily assign the mention "President Bush" to either the NE "George W. Bush" or "George H. W. Bush".
- A soft clustering algorithm, on the other hand, can assign equal probability to the two identities, indicating low entropy or high uncertainty in the solution. Additionally, the soft clustering algorithm can assign lower probability to the identity "Governor Jeb Bush", reflecting a less likely (though not impossible) coreference decision.

We first formalize the cross document coreference problem as a soft clustering problem, which minimizes the following objective function:

$$J_C(\mathcal{E}) = \sum_{i=1}^{C} \sum_{j=1}^{N} u_{ij}^m d^2(\mathbf{e}_j, \mathbf{v}_i) \qquad (1)$$

$$s.t. \quad \sum_{i=1}^{C} u_{ij} = 1 \text{ and } \sum_{j=1}^{N} u_{ij} > 0, u_{ij} \in [0, 1]$$

where $\mathbf{v}_i$ is a virtual (implicit) prototype of the i-th cluster ($\mathbf{e}_j, \mathbf{v}_i \in \mathcal{D}$) and m controls the fuzziness of the solution ($m > 1$; the solution approaches hard clustering as m approaches 1). We will further explain the generic distance function $d : \mathcal{D} \times \mathcal{D} \rightarrow \mathsf{R}$ in the next subsection. The goal of the optimization is to minimize the sum of deviations of patterns to the cluster prototypes. The clustering solution is a fuzzy partition $\mathcal{P}_\theta = \{\mathcal{C}_i\}$, where $\mathbf{e}_j \in \mathcal{C}_i$ if and only if $u_{ij} > \theta$.

We note from the outset that the optimization functional has the same form as the classical Fuzzy C-Means (FCM) algorithm (Bezdek, 1981), but major differences exist. FCM, as most object clustering algorithms, deals with *object data* represented in a vectorial form. In our case, the data is purely relational and only the mutual relationships between entities can be determined. To be exact, we can define the similarity/dissimilarity between a pair of attributes or relationships of the same type l between entities $\mathbf{e}_j$ and $\mathbf{e}_k$ as $s^{(l)}(\mathbf{e}_j, \mathbf{e}_k)$. For instance, the similarity between the occupations 'President' and 'Commander in Chief' can be computed using the JC semantic distance (Jiang and Conrath, 1997) with WordNet; the similarity of co-occurrence with other people can be measured by the Jaccard coefficient. In the next section, we propose to compute the relation strength $r(\cdot, \cdot)$ from the component similarities using aggregation weights learned from training data. Hence the N chained entities to be clustered can be represented as relational data using an $n \times n$ matrix $\mathbf{R}$, where $r_{j,k} = r(\mathbf{e}_j, \mathbf{e}_k)$. The Any Relation Clustering Algorithm (ARCA) (Corsini et al., 2005; Cimino et al., 2006) represents *relational data* as object data using their mutual relation strength and uses FCM for clustering. We adopt this approach to transform (*objectify*) a relational pattern $\mathbf{e}_j$ into an N dimensional vector $\mathbf{r}_j$ (i.e. the j-th row in the matrix $\mathbf{R}$) using a mapping $\Theta : \mathcal{D} \rightarrow \mathsf{R}^N$. In other words, each chained entity is represented as a vector of its relation strengths with all the entities. Fuzzy clusters can then be obtained by grouping closely related patterns using object clustering algorithm.

Furthermore, it is well known that FCM is a spherical clustering algorithm and thus is not generally applicable to relational data which may yield relational clusters of arbitrary and complicated shapes. Also, the distance in the transformed space may be non-Euclidean, rendering many clustering algorithms ineffective (many FCM extensions theoretically require the underlying distance to satisfy certain metric properties). In this work, we propose kernelized ARCA (called KARC) which uses a kernel-induced metric to handle the *objectified* relational data, as we introduce next.

2.2.2 Kernelized Fuzzy Clustering

Kernelization (Schölkopf and Smola, 2002) is a machine learning technique to transform patterns in the data space to a high-dimensional feature space so that the structure of the data can be more easily and adequately discovered. Specifically, a nonlinear transformation Φ maps data in R^N to H of possibly infinite dimensions (Hilbert space). The key idea is the *kernel trick* – without explicitly specifying Φ and H, the inner product in H can be computed by evaluating a kernel function K in the data space, i.e. $< \Phi(\mathbf{r}_i), \Phi(\mathbf{r}_j) >= K(\mathbf{r}_i, \mathbf{r}_j)$ (one of the most frequently used kernel functions is the Gaussian RBF kernel: $K(\mathbf{r}_j, \mathbf{r}_k) = \exp(-\lambda \|\mathbf{r}_j - \mathbf{r}_k\|^2)$). This technique has been successfully applied to SVMs to classify nonlinearly separable data (Vapnik, 1995). Kernelization preserves the simplicity in the formalism of the underlying clustering algorithm, meanwhile it yields highly nonlinear boundaries so that spherical clustering algorithms can apply (e.g. (Zhang and Chen, 2003) developed a kernelized object clustering algorithm based on FCM).

Let $\mathbf{w}_i$ denote the objectified virtual cluster $\mathbf{v}_i$, i.e. $\mathbf{w}_i = \Theta(\mathbf{v}_i)$. Using the kernel trick, the squared distance between $\Phi(\mathbf{r}_j)$ and $\Phi(\mathbf{w}_i)$ in the feature space H can be computed as:

$$\begin{aligned} & \|\Phi(\mathbf{r}_j) - \Phi(\mathbf{w}_i)\|_H^2 \qquad (2) \\ = \; & < \Phi(\mathbf{r}_j) - \Phi(\mathbf{w}_i), \Phi(\mathbf{r}_j) - \Phi(\mathbf{w}_i) > \\ = \; & < \Phi(\mathbf{r}_j), \Phi(\mathbf{r}_j) > -2 < \Phi(\mathbf{r}_j), \Phi(\mathbf{w}_i) > \\ & + < \Phi(\mathbf{w}_i), \Phi(\mathbf{w}_i) > \\ = \; & 2 - 2K(\mathbf{r}_j, \mathbf{w}_i) \qquad (3) \end{aligned}$$

assuming $K(\mathbf{r}, \mathbf{r}) = 1$. The KARC algorithm defines the generic distance d as $d^2(\mathbf{e}_j, \mathbf{v}_i) \stackrel{def}{=} \|\Phi(\mathbf{r}_j) - \Phi(\mathbf{w}_i)\|_H^2 = \|\Phi(\Theta(\mathbf{e}_j)) - \Phi(\Theta(\mathbf{v}_i))\|_H^2$ (we also use d_{ji}^2 as a notational shorthand).

Using Lagrange Multiplier as in FCM, the optimal solution for Equation (1) is:

$$u_{ij} = \begin{cases} \left[\sum_{h=1}^{C} \left(\frac{d_{ji}^2}{d_{jh}^2} \right)^{1/(m-1)} \right]^{-1} & , (d_{ji}^2 \neq 0) \\ 1 & , (d_{ji}^2 = 0) \end{cases} \qquad (4)$$

$$\Phi(\mathbf{w}_i) = \frac{\sum_{k=1}^{N} u_{ik}^m \Phi(\mathbf{r}_k)}{\sum_{k=1}^{N} u_{ik}^m} \quad (5)$$

Since Φ is an implicit mapping, Eq. (5) can not be explicitly evaluated. On the other hand, plugging Eq. (5) into Eq. (3), d_{ji}^2 can be explicitly represented by using the kernel matrix,

$$d_{ji}^2 = 2 - 2 \cdot \frac{\sum_{k=1}^{N} u_{ik}^m K(\mathbf{r}_j, \mathbf{r}_k)}{\sum_{k=1}^{N} u_{ik}^m} \quad (6)$$

With the derivation, the kernelized fuzzy clustering algorithm KARC works as follows. The chained entities $\mathcal{E}$ are first objectified into the relation strength matrix $\mathbf{R}$ using SEG, the details of which are described in the following section. The Gram matrix $\mathbf{K}$ is then computed based on the relation strength vectors using the kernel function. For a given number of clusters C, the initialization step is done by randomly picking C patterns as cluster centers, equivalently, C indices $\{n_1, .., n_C\}$ are randomly picked from $\{1, .., N\}$. D^0 is initialized by setting $d_{ji}^2 = 2 - 2K(\mathbf{r}_j, \mathbf{r}_{n_i})$. KARC alternately updates the membership matrix U and the kernel distance matrix D until convergence or running more than $maxIter$ iterations (Algorithm 1). Finally, the soft partition is generated based on the membership matrix U, which is the desired cross document coreference result.

Algorithm 1 KARC Alternating Optimization

Input: Gram matrix $\mathbf{K}$; #Clusters C; threshold θ
  initialize D^0
  $t \leftarrow 0$
  repeat
    $t \leftarrow t + 1$
    // 1– Update membership matrix U^t:
    $u_{ij} = \frac{(d_{ji}^2)^{-\frac{1}{m-1}}}{\sum_{h=1}^{C} (d_{jh}^2)^{-\frac{1}{m-1}}}$
    // 2– Update kernel distance matrix D^t:
    $d_{ji}^2 = 2 - 2 \cdot \frac{\sum_{k=1}^{N} u_{ik}^m K_{jk}}{\sum_{k=1}^{N} u_{ik}^m}$
  until $(t > maxIter)$ or
    $(t > 1$ and $|U^t - U^{t-1}| < \epsilon)$
  $\mathcal{P}_\theta \leftarrow$ `Generate_soft_partition`(U^t, θ)
Output: Fuzzy partition $\mathcal{P}_\theta$

2.2.3 Cluster Validation

In the CDC setting, the number of true underlying identities may vary depending on the entities' level of ambiguity (e.g. name frequency). Selecting the optimal number of clusters is in general a hard research question in clustering[1]. We adopt the Xie-Beni Index (XBI) (Xie and Beni, 1991) as in ARCA, which is one of the most popular cluster validities for fuzzy clustering algorithms. Xie-Beni Index (XBI) measures the goodness of clustering using the ratio of the intra-cluster variation and the inter-cluster separation. We measure the kernelized XBI (KXBI) in the feature space as,

$$\text{KXBI} = \frac{\sum_{i=1}^{C} \sum_{j=1}^{N} u_{ij}^m \|\Phi(\mathbf{r}_j) - \Phi(\mathbf{w}_i)\|_H^2}{N \cdot \min_{1 \le i < j \le C} \|\Phi(\mathbf{w}_i) - \Phi(\mathbf{w}_j)\|_H^2}$$

where the nominator is readily computed using D and the inter-cluster separation in the denominator can be evaluated using the similar kernel trick above (details omitted). Note that KXBI is only defined for $C > 1$. Thus we pick the C that corresponds to the first minimum of KXBI, and then compare its objective function value J_C with the cluster variance (J_1 for $C = 1$). The optimal C is chosen from the minimum of the two[2].

2.3 Specialist Ensemble Learning of Relation Strengths between Entities

One remaining element in the overall CDC approach is how the relation strength $r_{j,k}$ between two entities is computed. In (Cohen et al., 2003), a binary SVM model is trained and its confidence in predicting the non-coreferent class is used as the distance metric. In our case of using information extraction results for disambiguation, however, only some of the similarity features are present based on the available relationships in two profiles. In this work, we propose to treat each similarity function as a *specialist* that specializes in computing the similarity of a particular type of relationship. Indeed, the similarity function between a pair of attributes or relationships may in itself be a sophisticated component algorithm. We utilize the specialist ensemble learning framework (Freund et al., 1997) to combine these component

[1] In particular, clustering algorithms that regularize the optimization with cluster size are not applicable in our case.

[2] In practice, the entities to be disambiguated tend to be dominated by several major identities. Hence performance generally does not vary much in the range of large C values.

similarities into the relation strength for clustering. Here, a specialist is *awakened* for prediction only when the same type of relationships are present in both chained entities. A specialist can choose not to make a prediction if it is not confident enough for an instance. These aspects contrast with the traditional *insomniac* ensemble learning methods, where each component learner is always available for prediction (Freund et al., 1997). Also, specialists have different weights (in addition to their prediction) on the final relation strength, e.g. a match in a family relationship is considered more important than in a co-occurrence relationship.

Algorithm 2 SEG (Freund et al., 1997)

Input: Initial weight distribution $\mathbf{p^1}$;
learning rate $\eta > 0$; training set $\{< \mathbf{s}^t, y^t >\}$

1: **for** t=1 to T **do**
2: Predict using:

$$\tilde{y}^t = \frac{\sum_{i \in E^t} p_i^t s_i^t}{\sum_{i \in E^t} p_i^t} \quad (7)$$

3: Observe the true label y^t and incur square loss $L(\tilde{y}^t, y^t) = (\tilde{y}^t - y^t)^2$
4: Update weight distribution: for $i \in E_t$

$$p_i^{t+1} = \frac{p_i^t e^{-2\eta x_i^t (\tilde{y}^t - y^t)}}{\sum_{j \in E^t} p_j^t e^{-2\eta x_i^t (\tilde{y}^t - y^t)}} \cdot \sum_{j \in E_t} p_j^t \quad (8)$$

Otherwise: $p_i^{t+1} = p_i^t$
5: **end for**

Output: Model $\mathbf{p}$

The ensemble relation strength model is learned as follows. Given training data, the set of chained entities $\mathcal{E}_{train}$ is extracted as described earlier. For a pair of entities $\mathbf{e}_j$ and $\mathbf{e}_k$, a similarity vector $\mathbf{s}$ is computed using the component similarity functions for the respective attributes and relationships, and the true label is defined as $y = I\{\mathbf{e}_j$ and $\mathbf{e}_k$ are coreferent$\}$. The instances are subsampled to yield a balanced pairwise training set $\{< \mathbf{s}^t, y^t >\}$. We adopt the Specialist Exponentiated Gradient (SEG) (Freund et al., 1997) algorithm to learn the mixing weights of the specialists' prediction (Algorithm 2) in an online manner. In each training iteration, an instance $< \mathbf{s}^t, y^t >$ is presented to the learner (with E^t denoting the set of indices of awake specialists in $\mathbf{s}^t$). The SEG algorithm first predicts the value $\tilde{y}^t$ based on the awake specialists' decisions. The true value y^t is then revealed and the learner incurs a square loss between the predicted and the true values. The current weight distribution $\mathbf{p}$ is updated to minimize square loss: awake specialists are promoted or demoted in their weights according to the difference between the predicted and the true value. The learning iterations can run a few passes till convergence, and the model is learned in linear time with respect to T and is thus very efficient. In prediction time, let $E^{(jk)}$ denote the set of active specialists for the pair of entities $\mathbf{e}_j$ and $\mathbf{e}_k$, and $\mathbf{s}^{(jk)}$ denote the computed similarity vector. The predicted relation strength $r_{j,k}$ is,

$$r_{j,k} = \frac{\sum_{i \in E^{(jk)}} p_i s_i^{(jk)}}{\sum_{i \in E^{(jk)}} p_i} \quad (9)$$

2.4 Remarks

Before we conclude this section, we make several comments on using fuzzy clustering for cross document coreference. First, instead of conducting CDC for all entities concurrently (which can be computationally intensive with a large corpus), chained entities are first distributed into non-overlapping blocks. Clustering is performed for each block which is a drastically smaller problem space, while entities from different blocks are unlikely to be coreferent. Our CDC system uses phonetic blocking on the full name, so that name variations arising from translation, transliteration and abbreviation can be accommodated. Additional link constraints checking is also implemented to improve scalability though these are not the main focus of the paper.

There are several additional benefits in using a fuzzy clustering method besides the capability of probabilistic membership assignments in the CDC solution. In the clustered web search context, splitting a true identity into two clusters is perceived as a more severe error than putting irrelevant records in a cluster, as it is more difficult for the user to collect records in different clusters (to reconstruct the real underlying identity) than to prune away noisy records. While there is no universal way to handle this with hard clustering, soft clustering algorithms can more easily avoid the false negatives by allowing records to probabilistically appear in different clusters (subject to the sum of 1) using a more lenient threshold. Also, while there is no real prototypical elements in relational clustering, soft relational clustering

methods can naturally rank the profiles within a cluster according to their membership levels, which is an additional advantage for enhancing user consumption of the disambiguation results.

3 Experiments

In this section, we first formally define the evaluation metrics, followed by the introduction to the benchmark test sets and the system's performance.

3.1 Evaluation Metrics

We benchmarked our method using the standard purity and inverse purity clustering metrics as in the WePS evaluation. Let a set of clusters $\mathcal{P} = \{\mathcal{C}_i\}$ denote the system's partition as aforementioned and a set of categories $\mathcal{Q} = \{\mathcal{D}_j\}$ be the gold standard. The precision of a cluster $\mathcal{C}_i$ with respect to a category $\mathcal{D}_j$ is defined as,

$$\text{Precision}(\mathcal{C}_i, \mathcal{D}_j) = \frac{|\mathcal{C}_i \cap \mathcal{D}_j|}{|\mathcal{C}_i|}$$

Purity is in turn defined as the weighted average of the maximum precision achieved by the clusters on one of the categories,

$$\text{Purity}(\mathcal{P}, \mathcal{Q}) = \sum_{i=1}^{C} \frac{|\mathcal{C}_i|}{n} \max_j \text{Precision}(\mathcal{C}_i, \mathcal{D}_j)$$

where $n = \sum |\mathcal{C}_i|$. Hence purity penalizes putting noise chained entities in a cluster. Trivially, the maximum purity (i.e. 1) can be achieved by making one cluster per chained entity (referred to as the one-in-one baseline). Reversing the role of clusters and categories, $\text{Inverse\_purity}(\mathcal{P}, \mathcal{Q}) \stackrel{def}{=} \text{Purity}(\mathcal{Q}, \mathcal{P})$. Inverse Purity penalizes splitting chained entities belonging to the same category into different clusters. The maximum inverse purity can be similarly achieved by putting all entities into one cluster (all-in-one baseline).

Purity and inverse purity are similar to the precision and recall measures commonly used in IR. The F score, $F = 1/(\alpha\frac{1}{Purity} + (1-\alpha)\frac{1}{InversePurity})$, is used in performance evaluation. $\alpha = 0.2$ is used to give more weight to inverse purity, with the justification for the web person search mentioned earlier.

3.2 Dataset

We evaluate our methods using the benchmark test collection from the ACL SemEval-2007 web person search task (WePS) (Artiles et al., 2007). The test collection consists of three sets of 10 different names, sampled from ambiguous names from English Wikipedia (famous people), participants of the ACL 2006 conference (computer scientists) and common names from the US Census data, respectively. For each name, the top 100 documents retrieved from the Yahoo! Search API were annotated, yielding on average 45 real world identities per set and about 3k documents in total.

As we note in the beginning of Section 2, the human markup for the entities corresponding to the search queries is on the document level. The profile-based CDC approach, however, is to merge the mention-level entities. In our evaluation, we adopt the document label (and the person search query) to annotate the entity profiles that corresponds to the person name search query. Despite the difference, the results of the one-in-one and all-in-one baselines are almost identical to those reported in the WePS evaluation ($F = 0.52, 0.58$ respectively). Hence the performance reported here is comparable to the official evaluation results (Artiles et al., 2007).

3.3 Information Extraction and Similarities

We use an information extraction tool AeroText (Taylor, 2004) to construct the entity profiles. AeroText extracts two types of information for an entity. First, the attribute information about the person named entity includes first/middle/last names, gender, mention, etc. In addition, AeroText extracts relationship information between named entities, such as Family, List, Employment, Ownership, Citizen-Resident-Religion-Ethnicity and so on, as specified in the ACE evaluation. AeroText resolves the references of entities within a document and produces the entity profiles, used as input to the CDC system. Note that alternative IE or WDC tools, as well as additional attributes or relationships, can be readily used in the CDC methods we proposed.

A suite of similarity functions is designed to determine if the attributes relationships in a pair of entity profiles match or not:

Text similarity. To decide whether two names in the co-occurrence or family relationship match, we use the SoftTFIDF measure (Cohen et al., 2003), which is a hybrid matching scheme that combines the token-based TFIDF with the Jaro-Winkler string distance metric. This permits inexact matching of named entities due to name

variations, typos, etc.

Semantic similarity. Text or syntactic similarity is not always sufficient for matching relationships. WordNet and the information theoretic semantic distance (Jiang and Conrath, 1997) are used to measure the semantic similarity between concepts in relationships such as mention, employment, ownership, etc.

Other rule-based similarity. Several other cases require special treatment. For example, the employment relationships of *Senator* and *D-N.Y.* should match based on domain knowledge. Also, we design dictionary-based similarity functions to handle nicknames (Bill and William), acronyms (COLING for International Conference on Computational Linguistics), and geo-locations.

3.4 Evaluation Results

From the WePS training data, we generated a training set of around 32k pairwise instances as previously stated in Section 2.3. We then used the SEG algorithm to learn the weight distribution model. We tuned the parameters in the KARC algorithm using the training set with discrete grid search and chose $m = 1.6$ and $\theta = 0.3$. The RBF kernel (Gaussian) is used with $\gamma = 0.015$.

Table 1: Cross document coreference performance (I. Purity denotes inverse purity).

Method	Purity	I. Purity	F
KARC-S	0.657	0.795	0.740
KARC-H	0.662	0.762	0.710
FRC	0.484	0.840	0.697
One-in-one	1.000	0.482	0.524
All-in-one	0.279	1.000	0.571

The macro-averaged cross document coreference on the WePS test sets are reported in Table 1. The F score of our CDC system (KARC-S) is 0.740, comparable to the test results of the first tier systems in the official evaluation. The two baselines are also included. Since different feature sets, NLP tools, etc are used in different benchmarked systems, we are also interested in comparing the proposed algorithm with different soft relational clustering variants. First, we 'harden' the fuzzy partition produced by KARC by allowing an entity to appear in the cluster with highest membership value (KARC-H). Purity improves because of the removal of noise entities, though at the sacrifice of inverse purity and the F score deteriorates. We also implement a popular fuzzy relational clustering algorithm called FRC (Dave and Sen, 2002), whose optimization functional directly minimizes with respect to the relation matrix. With the same feature sets and distance function, KARC-S outperforms FRC in F score by about 5%. Because the test set is very ambiguous (on average only two documents per real world entity), the baselines have relatively high F score as observed in the WePS evaluation (Artiles et al., 2007). Table 2 further analyzes KARC-S's result on the three subsets Wikipedia, ACL06 and US Census. The F score is higher in the less ambiguous (the average number of identities) dataset and lower in the more ambiguous one, with a spread of 6%.

Table 2: Cross document coreference performance on subsets (I. Purity denotes inverse purity).

Test set	Identity	Purity	I. Purity	F
Wikipedia	56.5	0.666	0.752	0.717
ACL-06	31.0	0.783	0.771	0.773
US Census	50.3	0.554	0.889	0.754

We study how the cross document coreference performance changes as we vary the fuzziness in the solution (controlled by m). In Figure 1, as m increases from 1.4 to 1.9, purity improves by 10% to 0.67, which indicates that more correct coreference decisions (true positives) can be made in a softer configuration. The complimentary is true for inverse purity, though to a lesser extent. In this case, more false negatives, corresponding to the entities of different coreferents incorrectly

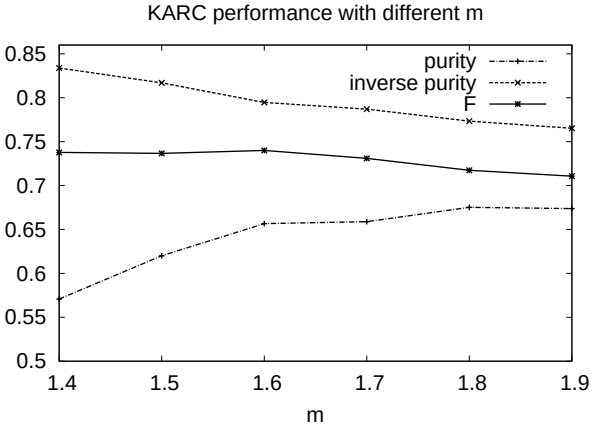


Figure 1: Purity, inverse purity and F score with different fuzzifiers m.

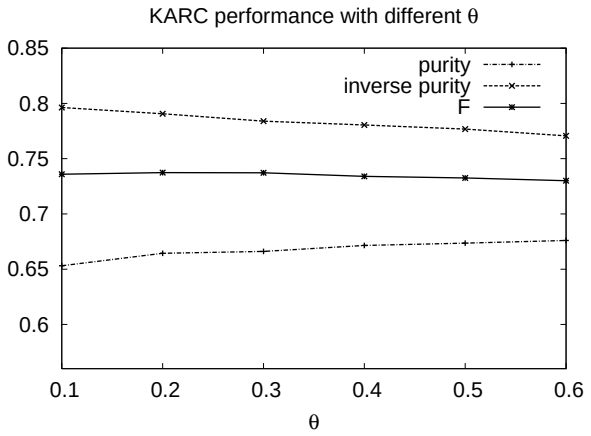


Figure 2: CDC performance with different θ.

linked, are made in a softer partition. The F score peaks at 0.74 ($m = 1.6$) and then slightly decreases, as the gain in purity is outweighed by the loss in inverse purity.

Figure 2 evaluates the impact of the different settings of θ (the threshold of including a chained entity in the fuzzy cluster) on the coreference performance. We observe that as we increase θ, purity improves indicating less 'noise' entities are included in the solution. On the other hand, inverse purity decreases meaning more coreferent entities are not linked due to the stricter threshold. Overall, the changes in the two metrics offset each other and the F score is relatively stable across a broad range of θ settings.

4 Related Work

The original work in (Bagga and Baldwin, 1998) proposed a CDC system by first performing WDC and then disambiguating based on the summary sentences of the chains. This is similar to ours in that mentions rather than documents are clustered, leveraging the advances in state-of-the-art WDC methods developed in NLP, e.g. (Ng and Cardie, 2001; Yang et al., 2008). On the other hand, our work goes beyond the simple bag-of-word features and vector space model in (Bagga and Baldwin, 1998; Gooi and Allan, 2004) with IE results. (Wan et al., 2005) describes a person resolution system WebHawk that clusters web pages using some extracted personal information including person name, title, organization, email and phone number, besides lexical features. (Mann and Yarowsky, 2003) extracts biographical information, which is relatively scarce in web data, for disambiguation. With the support of state-of-the-art information extraction tools, the profiles of entities in this work covers a broader range of relational information. (Niu et al., 2004) also leveraged IE support, but their approach was evaluated on a small artificial corpus. Also, the pairwise distance model is *insomniac* (i.e. all similarity specialists are *awake* for prediction) and our work extends this with a specialist learning framework.

Prior work has largely relied on using hierarchical clustering methods for CDC, with the threshold for stopping the merging set using the training data, e.g. (Mann and Yarowsky, 2003; Chen and Martin, 2007; Baron and Freedman, 2008). The fuzzy relational clustering method proposed in this paper we believe better addresses the uncertainty aspect of the CDC problem.

There are also orthogonal research directions for the CDC problem. (Li et al., 2004) solved the CDC problem by adopting a probabilistic view on how documents are generated and how names are sprinkled into them. (Bunescu and Pasca, 2006) showed that external information from Wikipedia can improve the disambiguation performance.

5 Conclusions

We have presented a profile-based Cross Document Coreference (CDC) approach based on a novel fuzzy relational clustering algorithm KARC. In contrast to traditional hard clustering methods, KARC produces fuzzy sets of identities which better reflect the intrinsic uncertainty of the CDC problem. Kernelization, as used in KARC, enables the optimization of clustering that is spherical in nature to apply to relational data that tend to have complicated shapes. KARC partitions named entities based on their profiles constructed by an information extraction tool. To match the profiles, a specialist ensemble algorithm predicts the pairwise distance by aggregating the similarities of the attributes and relationships in the profiles. We evaluated the proposed methods with experiments on a large benchmark collection and demonstrate that the proposed methods compare favorably with the top runs in the SemEval evaluation.

The focus of this work is on the novel learning and clustering methods for coreference. Future research directions include developing rich feature sets and using corpus level or external information. We believe that such efforts can further improve cross document coreference performance.

References

Javier Artiles, Julio Gonzalo, and Satoshi Sekine. 2007. The SemEval-2007 WePS evaluation: Establishing a benchmark for the web people search task. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, pages 64–69.

Amit Bagga and Breck Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of 36th International Conference On Computational Linguistics (ACL) and 17th international conference on Computational linguistics (COLING)*, pages 79–85.

Alex Baron and Marjorie Freedman. 2008. Who is who and what is what: Experiments in cross-document co-reference. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 274–283.

J. C. Bezdek. 1981. *Pattern Recognition with Fuzzy Objective Function Algoritms*. Plenum Press, NY.

Razvan Bunescu and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 9–16.

Ying Chen and James Martin. 2007. Towards robust unsupervised personal name disambiguation. In *Proc. of 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Mario G. C. A. Cimino, Beatrice Lazzerini, and Francesco Marcelloni. 2006. A novel approach to fuzzy clustering based on a dissimilarity relation extracted from data using a TS system. *Pattern Recognition*, 39(11):2077–2091.

William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. 2003. A comparison of string distance metrics for name-matching tasks. In *Proceedings of IJCAI Workshop on Information Integration on the Web*.

Paolo Corsini, Beatrice Lazzerini, and Francesco Marcelloni. 2005. A new fuzzy relational clustering algorithm based on the fuzzy c-means algorithm. *Soft Computing*, 9(6):439 – 447.

Rajesh N. Dave and Sumit Sen. 2002. Robust fuzzy clustering of relational data. *IEEE Transactions on Fuzzy Systems*, 10(6):713–727.

Yoav Freund, Robert E. Schapire, Yoram Singer, and Manfred K. Warmuth. 1997. Using and combining predictors that specialize. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing (STOC)*, pages 334–343.

Chung H. Gooi and James Allan. 2004. Cross-document coreference on a large scale corpus. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 9–16.

Jay J. Jiang and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference Research on Computational Linguistics*.

Xin Li, Paul Morie, and Dan Roth. 2004. Robust reading: Identification and tracing of ambiguous names. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 17–24.

Gideon S. Mann and David Yarowsky. 2003. Unsupervised personal name disambiguation. In *Conference on Computational Natural Language Learning (CoNLL)*, pages 33–40.

Vincent Ng and Claire Cardie. 2001. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 104–111.

Cheng Niu, Wei Li, and Rohini K. Srihari. 2004. Weakly supervised learning for cross-document person name disambiguation supported by information extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL)*, pages 597–604.

Bernhard Schölkopf and Alex Smola. 2002. *Learning with Kernels*. MIT Press, Cambridge, MA.

Sarah M. Taylor. 2004. Information extraction tools: Deciphering human language. *IT Professional*, 6(6):28 – 34.

Vladimir Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag New York.

Xiaojun Wan, Jianfeng Gao, Mu Li, and Binggong Ding. 2005. Person resolution in person search results: WebHawk. In *Proceedings of the 14th ACM international conference on Information and knowledge management (CIKM)*, pages 163–170.

Xuanli Lisa Xie and Gerardo Beni. 1991. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):841 – 847.

Xiaofeng Yang, Jian Su, Jun Lang, Chew L. Tan, Ting Liu, and Sheng Li. 2008. An entity-mention model for coreference resolution with inductive logic programming. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 843–851.

Dao-Qiang Zhang and Song-Can Chen. 2003. Clustering incomplete data using kernel-based fuzzy c-means algorithm. *Neural Processing Letters*, 18(3):155 – 162.

Who, What, When, Where, Why? Comparing Multiple Approaches to the Cross-Lingual 5W Task

Kristen Parton*, Kathleen R. McKeown*, Bob Coyne*, Mona T. Diab*, Ralph Grishman†, Dilek Hakkani-Tür‡, Mary Harper§, Heng Ji•, Wei Yun Ma*, Adam Meyers†, Sara Stolbach*, Ang Sun†, Gokhan Tur˚, Wei Xu† and Sibel Yaman‡

*Columbia University
New York, NY, USA
{kristen, kathy, coyne, mdiab, ma, sara}@cs.columbia.edu

‡International Computer Science Institute
Berkeley, CA, USA
{dilek, sibel}@icsi.berkeley.edu

•City University of New York
New York, NY, USA
hengji@cs.qc.cuny.edu

†New York University
New York, NY, USA
{grishman, meyers, asun, xuwei}@cs.nyu.edu

§Human Lang. Tech. Ctr. of Excellence, Johns Hopkins and U. of Maryland, College Park
mharper@umd.edu

˚SRI International
Palo Alto, CA, USA
gokhan@speech.sri.com

Abstract

Cross-lingual tasks are especially difficult due to the compounding effect of errors in language processing and errors in machine translation (MT). In this paper, we present an error analysis of a new cross-lingual task: the 5W task, a sentence-level understanding task which seeks to return the English 5W's (Who, What, When, Where and Why) corresponding to a Chinese sentence. We analyze systems that we developed, identifying specific problems in language processing and MT that cause errors. The best cross-lingual 5W system was still 19% worse than the best monolingual 5W system, which shows that MT significantly degrades sentence-level understanding. Neither source-language nor target-language analysis was able to circumvent problems in MT, although each approach had advantages relative to the other. A detailed error analysis across multiple systems suggests directions for future research on the problem.

1 Introduction

In our increasingly global world, it is ever more likely for a mono-lingual speaker to require information that is only available in a foreign language document. Cross-lingual applications address this need by presenting information in the speaker's language even when it originally appeared in some other language, using machine translation (MT) in the process. In this paper, we present an evaluation and error analysis of a cross-lingual application that we developed for a government-sponsored evaluation, the *5W task*.

The *5W task* seeks to summarize the information in a natural language sentence by distilling it into the answers to the 5W questions: Who, What, When, Where and Why. To solve this problem, a number of different problems in NLP must be addressed: predicate identification, argument extraction, attachment disambiguation, location and time expression recognition, and (partial) semantic role labeling. In this paper, we address the *cross-lingual 5W task*: given a source-language sentence, return the 5W's translated (comprehensibly) into the target language. Success in this task requires a synergy of successful MT and answer selection.

The questions we address in this paper are:

- How much does machine translation (MT) degrade the performance of cross-lingual 5W systems, as compared to monolingual performance?
- Is it better to do source-language analysis and then translate, or do target-language analysis on MT?
- Which specific problems in language processing and/or MT cause errors in 5W answers?

In this evaluation, we compare several different approaches to the cross-lingual 5W task, two that work on the target language (English) and one that works in the source language (Chinese).

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 423–431,
Suntec, Singapore, 2-7 August 2009.

A central question for many cross-lingual applications is whether to process in the source language and then translate the result, or translate documents first and then process the translation. Depending on how errorful the translation is, results may be more accurate if models are developed for the source language. However, if there are more resources in the target language, then the translate-then-process approach may be more appropriate. We present a detailed analysis, both quantitative and qualitative, of how the approaches differ in performance.

We also compare system performance on human translation (which we term reference translations) and MT of the same data in order to determine how much MT degrades system performance. Finally, we do an in-depth analysis of the errors in our 5W approaches, both on the NLP side and the MT side. Our results provide explanations for why different approaches succeed, along with indications of where future effort should be spent.

2 Prior Work

The cross-lingual 5W task is closely related to cross-lingual information retrieval and cross-lingual question answering (Wang and Oard 2006; Mitamura et al. 2008). In these tasks, a system is presented a query or question in the target language and asked to return documents or answers from a corpus in the source language. Although MT may be used in solving this task, it is only used by the algorithms – the final evaluation is done in the source language. However, in many real-life situations, such as global business, international tourism, or intelligence work, users may not be able to read the source language. In these cases, users must rely on MT to understand the system response. (Parton et al. 2008) examine the case of "translingual" information retrieval, where evaluation is done on translated results in the target language. In cross-lingual information extraction (Sudo et al. 2004) the evaluation is also done on MT, but the goal is to learn knowledge from a large corpus, rather than analyzing individual sentences.

The 5W task is also closely related to Semantic Role Labeling (SRL), which aims to efficiently and effectively derive semantic information from text. SRL identifies predicates and their arguments in a sentence, and assigns roles to each argument. For example, in the sentence "I baked a cake yesterday.", the predicate "baked" has three arguments. "I" is the subject of the predicate, "a cake" is the object and "yesterday" is a temporal argument.

Since the release of large data resources annotated with relevant levels of semantic information, such as the FrameNet (Baker et al., 1998) and PropBank corpora (Kingsbury and Palmer, 2003), efficient approaches to SRL have been developed (Carreras and Marquez, 2005). Most approaches to the problem of SRL follow the Gildea and Jurafsky (2002) model. First, for a given predicate, the SRL system identifies its arguments' boundaries. Second, the Argument types are classified depending on an adopted lexical resource such as PropBank or FrameNet. Both steps are based on supervised learning over labeled gold standard data. A final step uses heuristics to resolve inconsistencies when applying both steps simultaneously to the test data.

Since many of the SRL resources are English, most of the SRL systems to date have been for English. There has been work in other languages such as German and Chinese (Erk 2006; Sun 2004; Xue and Palmer 2005). The systems for the other languages follow the successful models devised for English, e.g. (Gildea and Palmer, 2002; Chen and Rambow, 2003; Moschitti, 2004; Xue and Palmer, 2004; Haghighi et al., 2005).

3 The Chinese-English 5W Task

3.1 5W Task Description

We participated in the 5W task as part of the DARPA GALE (Global Autonomous Language Exploitation) project. The goal is to identify the 5W's (Who, What, When, Where and Why) for a complete sentence. The motivation for the 5W task is that, as their origin in journalism suggests, the 5W's cover the key information nuggets in a sentence. If a system can isolate these pieces of information successfully, then it can produce a précis of the basic meaning of the sentence. Note that this task differs from QA tasks, where "Who" and "What" usually refer to definition type questions. In this task, the 5W's refer to semantic roles within a sentence, as defined in Table 1.

In order to get all 5W's for a sentence correct, a system must identify a top-level predicate, extract the correct arguments, and resolve attachment ambiguity. In the case of multiple top-level predicates, any of the top-level predicates may be chosen. In the case of passive verbs, the Who is the agent (often expressed as a "by clause", or not stated), and the What should include the syntactic subject.

Answers are judged Correct[1] if they identify a correct null argument or correctly extract an argument that is present in the sentence. Answers are not penalized for including extra text, such as prepositional phrases or subordinate clauses, unless the extra text includes text from another answer or text from another top-level predicate. In sentence 2a in Table 2, returning "bought and cooked" for the What would be Incorrect. Similarly, returning "bought the fish at the market" for the What would also be Incorrect, since it contains the Where. Answers may also be judged Partial, meaning that only part of the answer was returned. For example, if the What contains the predicate but not the logical object, it is Partial.

Since each sentence may have multiple correct sets of 5W's, it is not straightforward to produce a gold-standard corpus for automatic evaluation. One would have to specify answers for each possible top-level predicate, as well as which parts of the sentence are optional and which are not allowed. This also makes creating training data for system development problematic. For example, in Table 2, the sentence in 2a and 2b is the same, but there are two possible sets of correct answers. Since we could not rely on a gold-standard corpus, we used manual annotation to judge our 5W system, described in section 5.

3.2 The Cross-Lingual 5W Task

In the cross-lingual 5W task, a system is given a sentence in the source language and asked to produce the 5W's in the target language. In this task, both machine translation (MT) and 5W extraction must succeed in order to produce correct answers. One motivation behind the cross-lingual 5W task is MT evaluation. Unlike word- or phrase-overlap measures such as BLEU, the 5W evaluation takes into account "concept" or "nugget" translation. Of course, only the top-level predicate and arguments are evaluated, so it is not a complete evaluation. But it seeks to get at the understandability of the MT output, rather than just n-gram overlap.

Translation exacerbates the problem of automatically evaluating 5W systems. Since translation introduces paraphrase, rewording and sentence restructuring, the 5W's may change from one translation of a sentence to another translation of the same sentence. In some cases, roles may swap. For example, in Table 2, sentences 1a and 1b could be valid translations of the same Chinese sentence. They contain the same information, but the 5W answers are different. Also, translations may produce answers that are textually similar to correct answers, but actually differ in meaning. These differences complicate processing in the source followed by translation.

Example: On Tuesday, President Obama met with French President Sarkozy in Paris to discuss the economic crisis.

W	Definition	Example answer
WHO	Logical subject of the top-level predicate in WHAT, or null.	President Obama
WHAT	One of the top-level predicates in the sentence, and the predicate's logical object.	met with French President Sarkozy
WHEN	ARGM-TMP of the top-level predicate in WHAT, or null.	On Tuesday
WHERE	ARGM-LOC of the top-level predicate in WHAT, or null.	in Paris
WHY	ARGM-CAU of the top-level predicate in WHAT, or null.	to discuss the economic crisis

Table 1. Definition of the 5W task, and 5W answers from the example sentence above.

4 5W System

We developed a 5W combination system that was based on five other 5W systems. We selected four of these different systems for evaluation: the final combined system (which was our submission for the official evaluation), two systems that did analysis in the target-language (English), and one system that did analysis in the source language (Chinese). In this section, we describe the individual systems that we evaluated, the combination strategy, the parsers that we tuned for the task, and the MT systems.

	Sentence	WHO	WHAT
1a	Mary bought a cake from Peter.	Mary	bought a cake
1b	Peter sold Mary a cake.	Peter	sold Mary
2a	I bought the fish at the market yesterday and cooked it today.	I	bought the fish [WHEN: yesterday]
2b	I bought the fish at the market yesterday and cooked it today.	I	cooked it [WHEN: today]

Table 2. Example 5W answers.

[1] The specific guidelines for determining correctness were formulated by BAE.

4.1 Latent Annotation Parser

For this work, we have re-implemented and enhanced the Berkeley parser (Petrov and Klein 2007) in several ways: (1) developed a new method to handle rare words in English and Chinese; (2) developed a new model of unknown Chinese words based on characters in the word; (3) increased robustness by adding adaptive modification of pruning thresholds and smoothing of word emission probabilities. While the enhancements to the parser are important for robustness and accuracy, it is even more important to train grammars matched to the conditions of use. For example, parsing a Chinese sentence containing full-width punctuation with a parser trained on half-width punctuation reduces accuracy by over 9% absolute F. In English, parsing accuracy is seriously compromised by training a grammar with punctuation and case to process sentences without them.

We developed grammars for English and Chinese trained specifically for each genre by subsampling from available treebanks (for English, WSJ, BN, Brown, Fisher, and Switchboard; for Chinese, CTB5) and transforming them for a particular genre (e.g., for informal speech, we replaced symbolic expressions with verbal forms and remove punctuation and case) and by utilizing a large amount of genre-matched self-labeled training parses. Given these genre-specific parses, we extracted chunks and POS tags by script. We also trained grammars with a subset of function tags annotated in the treebank that indicate case role information (e.g., SBJ, OBJ, LOC, MNR) in order to produce function tags.

4.2 Individual 5W Systems

The English systems were developed for the monolingual 5W task and not modified to handle MT. They used hand-crafted rules on the output of the latent annotation parser to extract the 5Ws.

English-function used the function tags from the parser to map parser constituents to the 5Ws. First the Who, When, Where and Why were extracted, and then the remaining pieces of the sentence were returned as the What. The goal was to make sure to return a complete What answer and avoid missing the object.

English-LF, on the other hand, used a system developed over a period of eight years (Meyers et al. 2001) to map from the parser's syntactic constituents into logical grammatical relations (GLARF), and then extracted the 5Ws from the logical form. As a back-up, it also extracted GLARF relations from another English-treebank trained parser, the Charniak parser (Charniak 2001). After the parses were both converted to the 5Ws, they were then merged, favoring the system that: recognized the passive, filled more 5W slots or produced shorter 5W slots (providing that the WHAT slot consisted of more than just the verb). A third back-up method extracted 5Ws from part-of-speech tag patterns. Unlike *English-function, English-LF* explicitly tried to extract the shortest What possible, provided there was a verb and a possible object, in order to avoid multiple predicates or other 5W answers.

Chinese-align uses the latent annotation parser (trained for Chinese) to parse the Chinese sentences. A dependency tree converter (Johansson and Nuges 2007) was applied to the constituent-based parse trees to obtain the dependency relations and determine top-level predicates. A set of hand-crafted dependency rules based on observation of Chinese OntoNotes were used to map from the Chinese function tags into Chinese 5Ws. Finally, *Chinese-align* used the alignments of three separate MT systems to translate the 5Ws: a phrase-based system, a hierarchical phrase-based system, and a syntax augmented hierarchical phrase-based system. *Chinese-align* faced a number of problems in using the alignments, including the fact that the best MT did not always have the best alignment. Since the predicate is essential, it tried to detect when verbs were deleted in MT, and back-off to a different MT system. It also used strategies for finding and correcting noisy alignments, and for filtering When/Where answers from Who and What.

4.3 Hybrid System

A merging algorithm was learned based on a development test set. The algorithm selected all 5W's from a single system, rather than trying to merge W's from different systems, since the predicates may vary across systems. For each document genre (described in section 5.4), we ranked the systems by performance on the development data. We also experimented with a variety of features (for instance, does "What" include a verb). The best-performing features were used in combination with the ranked list of priority systems to create a rule-based merger.

4.4 MT Systems

The MT Combination system used by both of the English 5W systems combined up to nine separate MT systems. System weights for combination were optimized together with the language

model score and word penalty for a combination of BLEU and TER (2*(1-BLEU) + TER). Rescoring was applied after system combination using large language models and lexical trigger models. Of the nine systems, six were phrased-based systems (one of these used chunk-level reordering of the Chinese, one used word sense disambiguation, and one used unsupervised Chinese word segmentation), two were hierarchical phrase-based systems, one was a string-to-dependency system, one was syntax-augmented, and one was a combination of two other systems. Bleu scores on the government supplied test set in December 2008 were 35.2 for formal text, 29.2 for informal text, 33.2 for formal speech, and 27.6 for informal speech. More details may be found in (Matusov et al. 2009).

5 Methods

5.1 5W Systems

For the purposes of this evaluation[2], we compared the output of 4 systems: *English-Function*, *English-LF*, *Chinese-align*, and the combined system. Each English system was also run on reference translations of the Chinese sentence. So for each sentence in the evaluation corpus, there were 6 systems that each provided 5Ws.

5.2 5W Answer Annotation

For each 5W output, annotators were presented with the reference translation, the MT version, and the 5W answers. The 5W system names were hidden from the annotators. Annotators had to select "Correct", "Partial" or "Incorrect" for each W. For answers that were Partial or Incorrect, annotators had to further specify the source of the error based on several categories (described in section 6). All three annotators were native English speakers who were not system developers for any of the 5W systems that were being evaluated (to avoid biased grading, or assigning more blame to the MT system). None of the annotators knew Chinese, so all of the judgments were based on the reference translations.

After one round of annotation, we measured inter-annotator agreement on the Correct, Partial, or Incorrect judgment only. The kappa value was 0.42, which was lower than we expected. Another surprise was that the agreement was lower for When, Where and Why (κ=0.31) than for Who or What (κ=0.48). We found that, in cases where a system would get both Who and What wrong, it was often ambiguous how the remaining W's should be graded. Consider the sentence: "He went to the store yesterday and cooked lasagna today." A system might return erroneous Who and What answers, and return Where as "to the store" and When as "today." Since Where and When apply to different predicates, they cannot both be correct. In order to be consistent, if a system returned erroneous Who and What answers, we decided to mark the When, Where and Why answers Incorrect by default. We added clarifications to the guidelines and discussed areas of confusion, and then the annotators reviewed and updated their judgments.

After this round of annotating, κ=0.83 on the Correct, Partial, Incorrect judgments. The remaining disagreements were genuinely ambiguous cases, where a sentence could be interpreted multiple ways, or the MT could be understood in various ways. There was higher agreement on 5W's answers from the reference text compared to MT text, since MT is inherently harder to judge and some annotators were more flexible than others in grading garbled MT.

5.3 5W Error Annotation

In addition to judging the system answers by the task guidelines, annotators were asked to provide reason(s) an answer was wrong by selecting from a list of predefined errors. Annotators were asked to use their best judgment to "assign blame" to the 5W system, the MT, or both. There were six types of system errors and four types of MT errors, and the annotator could select any number of errors. (Errors are described further in section 6.) For instance, if the translation was correct, but the 5W system still failed, the blame would be assigned to the system. If the 5W system picked an incorrectly translated argument (e.g., "baked a moon" instead of "baked a cake"), then the error would be assigned to the MT system. Annotators could also assign blame to both systems, to indicate that they both made mistakes.

Since this annotation task was a 10-way selection, with multiple selections possible, there were some disagreements. However, if categorized broadly into 5W System errors only, MT errors only, and both 5W System and MT errors, then the annotators had a substantial level of agreement (κ=0.75 for error type, on sentences where both annotators indicated an error).

[2] Note that an official evaluation was also performed by DARPA and BAE. This evaluation provides more fine-grained detail on error types and gives results for the different approaches.

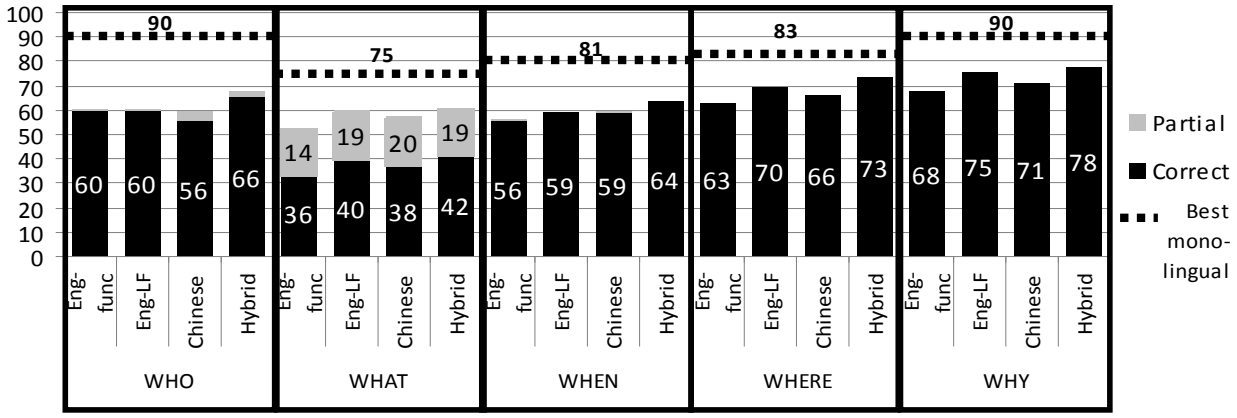


Figure 1. System performance on each 5W. "Partial" indicates that part of the answer was missing. Dashed lines show the performance of the best monolingual system (% Correct on human translations). For the last 3W's, the percent of answers that were Incorrect "by default" were: 30%, 24%, 27% and 22%, respectively, and 8% for the best monolingual system

5.4 5 W Corpus

The full evaluation corpus is 350 documents, roughly evenly divided between four genres: formal text (newswire), informal text (blogs and newsgroups), formal speech (broadcast news) and informal speech (broadcast conversation). For this analysis, we randomly sampled documents to judge from each of the genres. There were 50 documents (249 sentences) that were judged by a single annotator. A subset of that set, with 22 documents and 103 sentences, was judged by two annotators. In comparing the results from one annotator to the results from both annotators, we found substantial agreement. Therefore, we present results from the single annotator so we can do a more in-depth analysis. Since each sentence had 5W's, and there were 6 systems that were compared, there were 7,500 single-annotator judgments over 249 sentences.

6 Results

Figure 1 shows the cross-lingual performance (on MT) of all the systems for each 5W. The best monolingual performance (on human translations) is shown as a dashed line (% Correct only). If a system returned Incorrect answers for Who and What, then the other answers were marked Incorrect (as explained in section 5.2). For the last 3W's, the majority of errors were due to this (details in Figure 1), so our error analysis focuses on the Who and What questions.

6.1 Monolingual 5W Performance

To establish a monolingual baseline, the English 5W system was run on reference (human) translations of the Chinese text. For each partial or incorrect answer, annotators could select one or more of these reasons:

- Wrong predicate or multiple predicates.
- Answer contained another 5W answer.
- Passive handled wrong (WHO/WHAT).
- Answer missed.
- Argument attached to wrong predicate.

Figure 1 shows the performance of the best monolingual system for each 5W as a dashed line. The What question was the hardest, since it requires two pieces of information (the predicate and object). The When, Where and Why questions were easier, since they were null most of the time. (In English OntoNotes 2.0, 38% of sentences have a When, 15% of sentences have a Where, and only 2.6% of sentences have a Why.) The most common monolingual system error on these three questions was a missed answer, accounting for all of the Where errors, all but one Why error and 71% of the When errors. The remaining When errors usually occurred when the system assumed the wrong sense for adverbs (such as "then" or "just").

	Missing	Other 5W	Wrong/Multiple Predicates	Wrong
REF-func	37	29	22	7
REF-LF	54	20	17	13
MT-func	18	18	18	8
MT-LF	26	19	10	11
Chinese	23	17	14	8
Hybrid	13	17	15	12

Table 3. Percentages of Who/What errors attributed to each system error type.

The top half of Table 3 shows the reasons attributed to the Who/What errors for the reference corpus. Since *English-LF* preferred shorter answers, it frequently missed answers or parts of

answers. *English-LF* also had more Partial answers on the What question: 66% Correct and 12% Partial, versus 75% Correct and 1% Partial for *English-function*. On the other hand, *English-function* was more likely to return answers that contained incorrect extra information, such as another 5W or a second predicate.

6.2 Effect of MT on 5W Performance

The cross-lingual 5W task requires that systems return intelligible responses that are semantically equivalent to the source sentence (or, in the case of this evaluation, equivalent to the reference).

As can be seen in Figure 1, MT degrades the performance of the 5W systems significantly, for all question types, and for all systems. Averaged over all questions, the best monolingual system does 19% better than the best cross-lingual system. Surprisingly, even though *English-function* outperformed *English-LF* on the reference data, *English-LF* does consistently better on MT. This is likely due to its use of multiple back-off methods when the parser failed.

6.3 Source-Language vs. Target-Language

The Chinese system did slightly worse than either English system overall, but in the formal text genre, it outperformed both English systems.

Although the accuracies for the Chinese and English systems are similar, the answers vary a lot. Nearly half (48%) of the answers can be answered correctly by both the English system and the Chinese system. But 22% of the time, the English system returned the correct answer when the Chinese system did not. Conversely, 10% of the answers were returned correctly by the Chinese system and not the English systems. The hybrid system described in section 4.2 attempts to exploit these complementary advantages.

After running the hybrid system, 61% of the answers were from *English-LF*, 25% from *English-function*, 7% from *Chinese-align*, and the remaining 7% were from the other Chinese methods (not evaluated here). The hybrid did better than its parent systems on all 5Ws, and the numbers above indicate that further improvement is possible with a better combination strategy.

6.4 Cross-Lingual 5W Error Analysis

For each Partial or Incorrect answer, annotators were asked to select system errors, translation errors, or both. (Further analysis is necessary to distinguish between ASR errors and MT errors.) The translation errors considered were:

- Word/phrase deleted.
- Word/phrase mistranslated.
- Word order mixed up.
- MT unreadable.

Table 4 shows the translation reasons attributed to the Who/What errors. For all systems, the errors were almost evenly divided between system-only, MT-only and both, although the Chinese system had a higher percentage of system-only errors. The hybrid system was able to overcome many system errors (for example, in Table 2, only 13% of the errors are due to missing answers), but still suffered from MT errors.

	Mistranslation	Deletion	Word Order	Unreadable
MT-func	34	18	24	18
MT-LF	29	22	21	14
Chinese	32	17	9	13
Hybrid	35	19	27	18

Table 4. Percentages of Who/What errors by each system attributed to each translation error type.

Mistranslation was the biggest translation problem for all the systems. Consider the first example in Figure 3. Both English systems correctly extracted the Who and the When, but for

MT: After several rounds of reminded, I was a little bit
Ref: After several hints, it began to come back to me.
经过**几番提醒**,**我回**忆**起来了一点点**。

MT: The Guizhou province, within a certain bank robber, under the watchful eyes of a weak woman, and, with a knife stabbed the woman.
Ref: I saw that in a bank in Guizhou Province, robbers seized a vulnerable young woman in front of a group of onlookers and stabbed the woman with a knife.
看到贵**州省某**银**行内**,**劫匪在众目睽睽之下**,抢夺**一个弱女子**,**并且**,**用刀刺**伤该**女子**。

MT: Woke up after it was discovered that the property is not more than eleven people do not even said that the memory of the receipt of the country into the country.
Ref: Well, after waking up, he found everything was completely changed. Apart from having additional eleven grandchildren, even the motherland as he recalled has changed from a socialist country to a capitalist country.
那么**醒来之后却**发现**物是人非**,**多了十一个**孙子**不**说,连**祖国也从**记忆**当中的社会主**义**国家**变成**了**资本主义国家

Figure 3 Example sentences that presented problems for the 5W systems.

What they returned "was a little bit." This is the correct predicate for the sentence, but it does not match the meaning of the reference. The Chinese 5W system was able to select a better translation, and instead returned "remember a little bit."

Garbled word order was chosen for 21-24% of the target-language system Who/What errors, but only 9% of the source-language system Who/What errors. The source-language word order problems tended to be local, within-phrase errors (e.g., "the dispute over frozen funds" was translated as "the freezing of disputes"). The target-language system word order problems were often long-distance problems. For example, the second sentence in Figure 3 has many phrases in common with the reference translation, but the overall sentence makes no sense. The watchful eyes actually belong to a "group of onlookers" (deleted). Ideally, the robber would have "stabbed the woman" "with a knife," rather than vice versa. Long-distance phrase movement is a common problem in Chinese-English MT, and many MT systems try to handle it (e.g., Wang et al. 2007). By doing analysis in the source language, the Chinese 5W system is often able to avoid this problem – for example, it successfully returned "robbers" "grabbed a weak woman" for the Who/What of this sentence.

Although we expected that the Chinese system would have fewer problems with MT deletion, since it could choose from three different MT versions, MT deletion was a problem for all systems. In looking more closely at the deletions, we noticed that over half of deletions were verbs that were completely missing from the translated sentence. Since MT systems are tuned for word-based overlap measures (such as BLEU), verb deletion is penalized equally as, for example, determiner deletion. Intuitively, a verb deletion destroys the central meaning of a sentence, while a determiner is rarely necessary for comprehension. Other kinds of deletions included noun phrases, pronouns, named entities, negations and longer connecting phrases.

Deletion also affected When and Where. Deleting particles such as "in" and "when" that indicate a location or temporal argument caused the English systems to miss the argument. Word order problems in MT also caused attachment ambiguity in When and Where.

The "unreadable" category was an option of last resort for very difficult MT sentences. The third sentence in Figure 3 is an example where ASR and MT errors compounded to create an unparseable sentence.

7 Conclusions

In our evaluation of various 5W systems, we discovered several characteristics of the task. The What answer was the hardest for all systems, since it is difficult to include enough information to cover the top-level predicate and object, without getting penalized for including too much. The challenge in the When, Where and Why questions is due to sparsity – these responses occur in much fewer sentences than Who and What, so systems most often missed these answers. Since this was a new task, this first evaluation showed clear issues on the language analysis side that can be improved in the future.

The best cross-lingual 5W system was still 19% worse than the best monolingual 5W system, which shows that MT significantly degrades sentence-level understanding. A serious problem in MT for systems was deletion. Chinese constituents that were never translated caused serious problems, even when individual systems had strategies to recover. When the verb was deleted, no top level predicate could be found and then all 5Ws were wrong.

One of our main research questions was whether to extract or translate first. We hypothesized that doing source-language analysis would be more accurate, given the noise in Chinese MT, but the systems performed about the same. This is probably because the English tools (logical form extraction and parser) were more mature and accurate than the Chinese tools.

Although neither source-language nor target-language analysis was able to circumvent problems in MT, each approach had advantages relative to the other, since they did well on different sets of sentences. For example, *Chinese-align* had fewer problems with word order, and most of those were due to local word-order problems.

Since the source-language and target-language systems made different kinds of mistakes, we were able to build a hybrid system that used the relative advantages of each system to outperform all systems. The different types of mistakes made by each system suggest features that can be used to improve the combination system in the future.

Acknowledgments

This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under contract number HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsors.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In COLING-ACL '98: Proceedings of the Conference, held at the University of Montréal, pages 86–90.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005), pages 152–164.

Eugene Charniak. 2001. Immediate-head parsing for language models. In Proceedings of the 39th Annual Meeting on Association For Computational Linguistics (Toulouse, France, July 06 - 11, 2001).

John Chen and Owen Rambow. 2003. Use of deep linguistic features for the recognition and labeling of semantic arguments. In Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, Sapporo, Japan.

Katrin Erk and Sebastian Pado. 2006. Shalmaneser – a toolchain for shallow semantic parsing. Proceedings of LREC.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. Computational Linguistics, 28(3):245–288.

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02), Philadelphia, PA, USA.

Mary Harper and Zhongqiang Huang. 2009. Chinese Statistical Parsing, chapter to appear.

Aria Haghighi, Kristina Toutanova, and Christopher Manning. 2005. A joint model for semantic role labeling. In Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005), pages 173–176.

Paul Kingsbury and Martha Palmer. 2003. Propbank: the next level of treebank. In Proceedings of Treebanks and Lexical Theories.

Evgeny Matusov, Gregor Leusch, & Hermann Ney: Learning to combine machine translation systems. In: Cyril Goutte, Nicola Cancedda, Marc Dymetman, & George Foster (eds.) Learning machine translation. (Cambridge, Mass.: The MIT Press, 2009); pp.257-276.

Adam Meyers, Ralph Grishman, Michiko Kosaka and Shubin Zhao. 2001. Covering Treebanks with GLARF. In Proceedings of the ACL 2001 Workshop on Sharing Tools and Resources. Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, 51-58.

Teruko Mitamura, Eric Nyberg, Hideki Shima, Tsuneaki Kato, Tatsunori Mori, Chin-Yew Lin, Ruihua Song, Chuan-Jie Lin, Tetsuya Sakai, Donghong Ji, and Noriko Kando. 2008. Overview of the NTCIR-7 ACLIA Tasks: Advanced Cross-Lingual Information Access. In Proceedings of the Seventh NTCIR Workshop Meeting.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question answer classification. In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 776–783.

Kristen Parton, Kathleen R. McKeown, James Allan, and Enrique Henestroza. Simultaneous multilingual search for translingual information retrieval. In Proceedings of ACM 17th Conference on Information and Knowledge Management (CIKM), Napa Valley, CA, 2008.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2007).

Sudo, K., Sekine, S., and Grishman, R. 2004. Cross-lingual information extraction system evaluation. In Proceedings of the 20th international Conference on Computational Linguistics.

Honglin Sun and Daniel Jurafsky. 2004. Shallow Semantic Parsing of Chinese. In Proceedings of NAACL-HLT.

Cynthia A. Thompson, Roger Levy, and Christopher Manning. 2003. A generative model for semantic role labeling. In 14th European Conference on Machine Learning.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In Dekang Lin and Dekai Wu, editors, Proceedings of EMNLP 2004, pages 88–94, Barcelona, Spain, July. Association for Computational Linguistics.

Xue, Nianwen and Martha Palmer. 2005. Automatic semantic role labeling for Chinese verbs. InProceedings of the Nineteenth International Joint Conference on Artificial Intelligence, pages 1160-1165.

Chao Wang, Michael Collins, and Philipp Koehn. 2007. Chinese Syntactic Reordering for Statistical Machine Translation. Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), 737-745.

Jianqiang Wang and Douglas W. Oard, 2006. "Combining Bidirectional Translation and Synonymy for Cross-Language Information Retrieval," in 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 202-209.

Bilingual Co-Training for Monolingual Hyponymy-Relation Acquisition

Jong-Hoon Oh, Kiyotaka Uchimoto, and Kentaro Torisawa
Language Infrastructure Group, MASTAR Project,
National Institute of Information and Communications Technology (NICT)
3-5 Hikaridai Seika-cho, Soraku-gun, Kyoto 619-0289 Japan
{rovellia,uchimoto,torisawa}@nict.go.jp

Abstract

This paper proposes a novel framework called *bilingual co-training* for a large-scale, accurate acquisition method for *monolingual* semantic knowledge. In this framework, we combine the independent processes of monolingual semantic-knowledge acquisition for two languages using bilingual resources to boost performance. We apply this framework to large-scale hyponymy-relation acquisition from Wikipedia. Experimental results show that our approach improved the F-measure by 3.6–10.3%. We also show that bilingual co-training enables us to build classifiers for two languages in tandem with the same combined amount of data as required for training a single classifier in isolation while achieving superior performance.

1 Motivation

Acquiring and accumulating semantic knowledge are crucial steps for developing high-level NLP applications such as question answering, although it remains difficult to acquire a large amount of highly accurate semantic knowledge. This paper proposes a novel framework for a large-scale, accurate acquisition method for monolingual semantic knowledge, especially for semantic relations between nominals such as hyponymy and meronymy. We call the framework *bilingual co-training*.

The acquisition of semantic relations between nominals can be seen as a classification task of semantic relations – to determine whether two nominals hold a particular semantic relation (Girju et al., 2007). Supervised learning methods, which have often been applied to this classification task, have shown promising results. In those methods, however, a large amount of training data is usually required to obtain high performance, and the high costs of preparing training data have always been a bottleneck.

Our research on bilingual co-training sprang from a very simple idea: perhaps training data in a language can be enlarged without much cost if we translate training data in another language and add the translation to the training data in the original language. We also noticed that it may be possible to further enlarge the training data by translating the reliable part of the classification results in another language. Since the learning settings (feature sets, feature values, training data, corpora, and so on) are usually different in two languages, the reliable part in one language may be overlapped by an unreliable part in another language. Adding the translated part of the classification results to the training data will improve the classification results in the unreliable part. This process can also be repeated by swapping the languages, as illustrated in Figure 1. Actually, this is nothing other than a bilingual version of co-training (Blum and Mitchell, 1998).

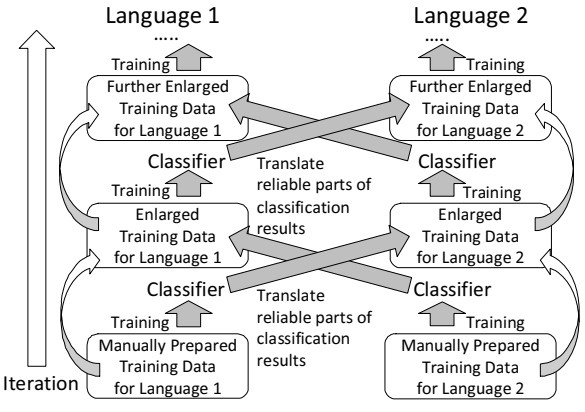


Figure 1: Concept of *bilingual co-training*

Let us show an example in our current task: hyponymy-relation acquisition from Wikipedia. Our original approach for this task was super-

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 432–440,
Suntec, Singapore, 2-7 August 2009.

vised learning based on the approach proposed by Sumida et al. (2008), which was only applied for Japanese and achieved around 80% in F-measure. In their approach, a common substring in a hypernym and a hyponym is assumed to be one strong clue for recognizing that the two words constitute a hyponymy relation. For example, recognizing a proper hyponymy relation between two Japanese words, 酵素 (***kouso*** meaning *enzyme*) and 加水分解酵素 (*kasuibunkai****kouso*** meaning *hydrolase*), is relatively easy because they share a common suffix: **kouso**. On the other hand, judging whether their English translations (*enzyme* and *hydrolase*) have a hyponymy relation is probably more difficult since they do not share any substrings. A classifier for Japanese will regard the hyponymy relation as valid with high confidence, while a classifier for English may not be so positive. In this case, we can compensate for the weak part of the English classifier by adding the English translation of the Japanese hyponymy relation, which was recognized with high confidence, to the English training data.

In addition, if we repeat this process by swapping English and Japanese, further improvement may be possible. Furthermore, the reliable parts that are automatically produced by a classifier can be larger than manually tailored training data. If this is the case, the effect of adding the translation to the training data can be quite large, and the same level of effect may not be achievable by a reasonable amount of labor for preparing the training data. This is the whole idea.

Through a series of experiments, this paper shows that the above idea is valid at least for one task: large-scale monolingual hyponymy-relation acquisition from English and Japanese Wikipedia. Experimental results showed that our method based on bilingual co-training improved the performance of monolingual hyponymy-relation acquisition about 3.6–10.3% in the F-measure. Bilingual co-training also enables us to build classifiers for two languages in tandem with the same combined amount of data as would be required for training a single classifier in isolation while achieving superior performance.

People probably expect that a key factor in the success of this bilingual co-training is how to translate the training data. We actually did translation by a simple look-up procedure in the existing translation dictionaries without any machine translation systems or disambiguation processes. Despite this simple approach, we obtained consistent improvement in our task using various translation dictionaries.

This paper is organized as follows. Section 2 presents bilingual co-training, and Section 3 precisely describes our system. Section 4 describes our experiments and presents results. Section 5 discusses related work. Conclusions are drawn and future work is mentioned in Section 6.

2 Bilingual Co-Training

Let S and T be two different languages, and let CL be a set of class labels to be obtained as a result of learning/classification. To simplify the discussion, we assume that a class label is binary; i.e., the classification results are "yes" or "no." Thus, $CL = \{\textit{yes}, \textit{no}\}$. Also, we denote the set of all nonnegative real numbers by R^+.

Assume $X = X_S \cup X_T$ is a set of instances in languages S and T to be classified. In the context of a hyponymy-relation acquisition task, the instances are pairs of nominals. Then we assume that classifier c assigns class label cl in CL and confidence value r for assigning the label, i.e., $c(x) = (x, cl, r)$, where $x \in X$, $cl \in CL$, and $r \in R^+$. Note that we used support vector machines (SVMs) in our experiments and (the absolute value of) the distance between a sample and the hyperplane determined by the SVMs was used as confidence value r. The training data are denoted by $L \subset X \times CL$, and we denote the learning by function $LEARN$; if classifier c is trained by training data L, then $c = LEARN(L)$. Particularly, we denote the training sets for S and T that are manually prepared by L_S and L_T, respectively. Also, bilingual instance dictionary D_{BI} is defined as the translation pairs of instances in X_S and X_T. Thus, $D_{BI} = \{(s,t)\} \subset X_S \times X_T$. In the case of hyponymy-relation acquisition in English and Japanese, $(s,t) \in D_{BI}$ could be (s=(*enzyme*, *hydrolase*), t=(酵素 (meaning *enzyme*), 加水分解酵素 (meaning *hydrolase*))).

Our bilingual co-training is given in Figure 2. In the initial stage, c_S^0 and c_T^0 are learned with manually labeled instances L_S and L_T (lines 2–5). Then c_S^i and c_T^i are applied to classify instances in X_S and X_T (lines 6–7). Denote CR_S^i as a set of the classification results of c_S^i on instances X_S that is not in L_S^i and is registered in D_{BI}. Lines 10–18 describe a way of selecting from CR_S^i newly la-

1: $i = 0$
2: $L_S^0 = L_S$; $L_T^0 = L_T$
3: **repeat**
4: $c_S^i := LEARN(L_S^i)$
5: $c_T^i := LEARN(L_T^i)$
6: $CR_S^i := \{c_S^i(x_S) | x_S \in X_S,$
$\forall cl\ (x_S, cl) \notin L_S^i, \exists x_T\ (x_S, x_T) \in D_{BI}\}$
7: $CR_T^i := \{c_T^i(x_T) | x_T \in X_T,$
$\forall cl\ (x_T, cl) \notin L_T^i, \exists x_S\ (x_S, x_T) \in D_{BI}\}$
8: $L_S^{(i+1)} := L_S^i$
9: $L_T^{(i+1)} := L_T^i$
10: **for** each $(x_S, cl_S, r_S) \in TopN(CR_S^i)$ **do**
11: **for** each x_T such that $(x_S, x_T) \in D_{BI}$ and $(x_T, cl_T, r_T) \in CR_T^i$ **do**
12: **if** $r_S > \theta$ **then**
13: **if** $r_T < \theta$ or $cl_S = cl_T$ **then**
14: $L_T^{(i+1)} := L_T^{(i+1)} \cup \{(x_T, cl_S)\}$
15: **end if**
16: **end if**
17: **end for**
18: **end for**
19: **for** each $(x_T, cl_T, r_T) \in TopN(CR_T^i)$ **do**
20: **for** each x_S such that $(x_S, x_T) \in D_{BI}$ and $(x_S, cl_S, r_S) \in CR_S^i$ **do**
21: **if** $r_T > \theta$ **then**
22: **if** $r_S < \theta$ or $cl_S = cl_T$ **then**
23: $L_S^{(i+1)} := L_S^{(i+1)} \cup \{(x_S, cl_T)\}$
24: **end if**
25: **end if**
26: **end for**
27: **end for**
28: $i = i + 1$
29: **until** a fixed number of iterations is reached

Figure 2: Pseudo-code of *bilingual co-training*

beled instances to be added to a new training set in T. $TopN(CR_S^i)$ is a set of $c_S^i(x)$, whose r_S is top-N highest in CR_S^i. (In our experiments, $N = 900$.) During the selection, c_S^i acts as a teacher and c_T^i as a student. The teacher instructs his student in the class label of x_T, which is actually a translation of x_S by bilingual instance dictionary D_{BI}, through cl_S only if he can do it with a certain level of confidence, say $r_S > \theta$, and if one of two other condition meets ($r_T < \theta$ or $cl_S = cl_T$). $cl_S = cl_T$ is a condition to avoid problems, especially when the student also has a certain level of confidence in his opinion on a class label but disagrees with the teacher: $r_T > \theta$ and $cl_S \neq cl_T$. In that case, the teacher does nothing and ignores the instance. Condition $r_T < \theta$ enables the teacher to instruct his student in the class label of x_T in spite of their disagreement in a class label. If every condition is satisfied, (x_T, cl_S) is added to existing labeled instances $L_T^{(i+1)}$. The roles are reversed in lines 19–27 so that c_T^i becomes a teacher and c_S^i a student.

Similar to co-training (Blum and Mitchell, 1998), one classifier seeks another's opinion to select new labeled instances. One main difference between co-training and bilingual co-training is the space of instances: co-training is based on different features of the same instances, and bilingual co-training is based on different spaces of instances divided by languages. Since some of the instances in different spaces are connected by a bilingual instance dictionary, they seem to be in the same space. Another big difference lies in the role of the two classifiers. The two classifiers in co-training work on the same task, but those in bilingual co-training do *the same type of* task rather than the same task.

3 Acquisition of Hyponymy Relations from Wikipedia

Our system, which acquires hyponymy relations from Wikipedia based on bilingual co-training, is described in Figure 3. The following three main parts are described in this section: candidate extraction, hyponymy-relation classification, and bilingual instance dictionary construction.

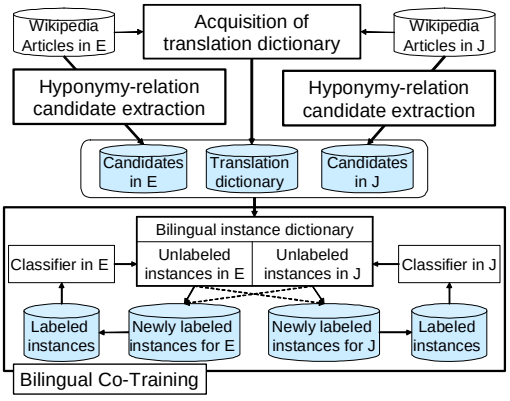
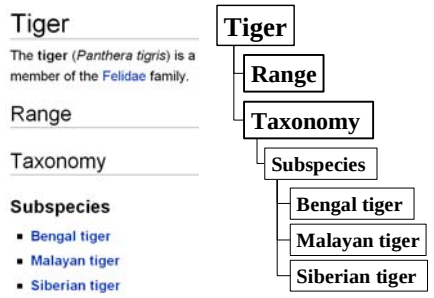


Figure 3: System architecture

3.1 Candidate Extraction

We follow Sumida et al. (2008) to extract hyponymy-relation candidates from English and Japanese Wikipedia. A layout structure is chosen



(a) Layout structure of article TIGER (b) Tree structure of Figure 4(a)

Figure 4: Wikipedia article and its layout structure

as a source of hyponymy relations because it can provide a huge amount of them (Sumida et al., 2008; Sumida and Torisawa, 2008)[1], and recognition of the layout structure is easy regardless of languages. Every English and Japanese Wikipedia article was transformed into a tree structure like Figure 4, where layout items *title*, *(sub)section headings*, and *list items* in an article were used as nodes in a tree structure. Sumida et al. (2008) found that some pairs consisting of a node and one of its descendants constituted a proper hyponymy relation (e.g., (TIGER, SIBERIAN TIGER)), and this could be a knowledge source of hyponymy relation acquisition. A hyponymy-relation candidate is then extracted from the tree structure by regarding a node as a hypernym candidate and all its subordinate nodes as hyponym candidates of the hypernym candidate (e.g., (TIGER, TAXONOMY) and (TIGER, SIBERIAN TIGER) from Figure 4). 39 M English hyponymy-relation candidates and 10 M Japanese ones were extracted from Wikipedia. These candidates are classified into proper hyponymy relations and others by using the classifiers described below.

3.2 Hyponymy-Relation Classification

We use SVMs (Vapnik, 1995) as classifiers for the classification of the hyponymy relations on the hyponymy-relation candidates. Let **hyper** be a hypernym candidate, **hypo** be a **hyper**'s hyponym candidate, and (**hyper, hypo**) be a hyponymy-relation candidate. The lexical, structure-based, and infobox-based features of (**hyper, hypo**) in Table 1 are used for building English and Japanese classifiers. Note that SF_3–SF_5 and IF were not used in Sumida et al. (2008) but LF_1–LF_5 and SF_1–SF_2 are the same as their feature set.

Let us provide an overview of the feature sets used in Sumida et al. (2008). See Sumida et al. (2008) for more details. Lexical features LF_1–LF_5 are used to recognize the lexical evidence encoded in **hyper** and **hypo** for hyponymy relations. For example, (**hyper,hypo**) is often a proper hyponymy relation if **hyper** and **hypo** share the same head morpheme or word. In LF_1 and LF_2, such information is provided along with the words/morphemes and the parts of speech of **hyper** and **hypo**, which can be multi-word/morpheme nouns. TagChunk (Daumé III et al., 2005) for English and MeCab (MeCab, 2008) for Japanese were used to provide the lexical features. Several simple lexical patterns[2] were also applied to hyponymy-relation candidates. For example, "List of artists" is converted into "artists" by lexical pattern "list of X." Hyponymy-relation candidates whose hypernym candidate matches such a lexical pattern are likely to be valid (e.g., (List of artists, Leonardo da Vinci)). We use LF_4 for dealing with these cases. If a typical or frequently used section heading in a Wikipedia article, such as "History" or "References," is used as a hyponym candidate in a hyponymy-relation candidate, the hyponymy-relation candidate is usually not a hyponymy relation. LF_5 is used to recognize these hyponymy-relation candidates.

Structure-based features are related to the tree structure of Wikipedia articles from which hyponymy-relation candidate (**hyper,hypo**) is extracted. SF_1 provides the distance between **hyper** and **hypo** in the tree structure. SF_2 represents the type of layout items from which **hyper** and **hypo** are originated. These are the feature sets used in Sumida et al. (2008).

We also added some new items to the above feature sets. SF_3 represents the types of tree nodes including root, leaf, and others. For example, (**hyper,hypo**) is seldom a hyponymy relation if **hyper** is from a root node (or title) and **hypo** is from a **hyper**'s child node (or section headings). SF_4 and SF_5 represent the structural contexts of **hyper** and **hypo** in a tree structure. They can provide evidence related to similar hyponymy-relation candidates in the structural contexts.

An infobox-based feature, IF, is based on a

[1] Sumida et al. (2008) reported that they obtained 171 K, 420 K, and 1.48 M hyponymy relations from a definition sentence, a category system, and a layout structure in Japanese Wikipedia, respectively.

[2] We used the same Japanese lexical patterns in Sumida et al. (2008) to build English lexical patterns with them.

Type	Description	Example
LF_1	Morphemes/words	hyper: tiger*, hypo: Siberian, hypo: tiger*
LF_2	POS of morphemes/words	hyper: NN*, hypo: NP, hypo: NN*
LF_3	**hyper** and **hypo**, themselves	hyper: Tiger, hypo: Siberian tiger
LF_4	Used lexical patterns	hyper: "List of X", hypo: "Notable X"
LF_5	Typical section headings	hyper: History, hypo: Reference
SF_1	Distance between **hyper** and **hypo**	3
SF_2	Type of layout items	hyper: title, hypo: bulleted list
SF_3	Type of tree nodes	hyper: root node, hypo: leaf node
SF_4	LF_1 and LF_3 of **hypo**'s parent node	LF_3:Subspecies
SF_5	LF_1 and LF_3 of **hyper**'s child node	LF_3: Taxonomy
IF	Semantic properties of **hyper** and **hypo**	hyper: (taxobox,species), hypo: (taxobox,name)

Table 1: Feature type and its value. $*$ in LF_1 and LF_2 represent the head morpheme/word and its POS. Except those in LF_4 and LF_5, examples are derived from (TIGER, SIBERIAN TIGER) in Figure 4.

Wikipedia infobox, a special kind of template, that describes a tabular summary of an article subject expressed by attribute-value pairs. An attribute type coupled with the infobox name to which it belongs provides the semantic properties of its value that enable us to easily understand what the attribute value means (Auer and Lehmann, 2007; Wu and Weld, 2007). For example, infobox template *City Japan* in Wikipedia article *Kyoto* contains several attribute-value pairs such as "Mayor=Daisaku Kadokawa" as *attribute=its value*. What *Daisaku Kadokawa*, the attribute value of *mayor* in the example, represents is hard to understand alone if we lack knowledge, but its attribute type, *mayor*, gives a clue–*Daisaku Kadokawa* is a *mayor* related to *Kyoto*. These semantic properties enable us to discover semantic evidence for hyponymy relations. We extract triples (*infobox name, attribute type, attribute value*) from the Wikipedia infoboxes and encode such information related to **hyper** and **hypo** in our feature set IF.[3]

3.3 Bilingual Instance Dictionary Construction

Multilingual versions of Wikipedia articles are connected by cross-language links and usually have titles that are bilinguals of each other (Erdmann et al., 2008). English and Japanese articles connected by a cross-language link are extracted from Wikipedia, and their titles are regarded as translation pairs[4]. The translation pairs between English and Japanese terms are used for building bilingual instance dictionary D_{BI} for hyponymy-relation acquisition, where D_{BI} is composed of translation pairs between English and Japanese hyponymy-relation candidates[5].

4 Experiments

We used the MAY 2008 version of English Wikipedia and the JUNE 2008 version of Japanese Wikipedia for our experiments. 24,000 hyponymy-relation candidates, randomly selected in both languages, were manually checked to build training, development, and test sets[6]. Around 8,000 hyponymy relations were found in the manually checked data for both languages[7]. 20,000 of the manually checked data were used as a training set for training the initial classifier. The rest were equally divided into development and test sets. The development set was used to select the optimal parameters in bilingual co-training and the test set was used to evaluate our system.

We used TinySVM (TinySVM, 2002) with a polynomial kernel of degree 2 as a classifier. The maximum iteration number in the bilingual co-training was set as 100. Two parameters, θ and $TopN$, were selected through experiments on the development set. $\theta = 1$ and $TopN$=900 showed

[3]We obtained 1.6 M object-attribute-value triples in Japanese and 5.9 M in English.

[4]197 K translation pairs were extracted.

[5]We also used redirection links in English and Japanese Wikipedia for recognizing the variations of terms when we built a bilingual instance dictionary with Wikipedia cross-language links.

[6]It took about two or three months to check them in each language.

[7]Regarding a hyponymy relation as a positive sample and the others as a negative sample for training SVMs, "positive sample:negative sample" was about 8,000:16,000=1:2

the best performance and were used as the optimal parameter in the following experiments.

We conducted three experiments to show effects of bilingual co-training, training data size, and bilingual instance dictionaries. In the first two experiments, we experimented with a bilingual instance dictionary derived from Wikipedia cross-language links. Comparison among systems based on three different bilingual instance dictionaries is shown in the third experiment.

Precision (P), recall (R), and F_1-measure (F_1), as in Eq (1), were used as the evaluation measures, where *Rel* represents a set of manually checked hyponymy relations and *HRbyS* represents a set of hyponymy-relation candidates classified as hyponymy relations by the system:

$$\begin{aligned} P &= |Rel \cap HRbyS|/|HRbyS| \quad (1) \\ R &= |Rel \cap HRbyS|/|Rel| \\ F_1 &= 2 \times (P \times R)/(P + R) \end{aligned}$$

4.1 Effect of Bilingual Co-Training

	ENGLISH			JAPANESE		
	P	R	F_1	P	R	F_1
SYT	78.5	63.8	70.4	75.0	77.4	76.1
INIT	77.9	67.4	72.2	74.5	78.5	76.6
TRAN	76.8	70.3	73.4	76.7	79.3	78.0
BICO	78.0	83.7	80.7	78.3	85.2	81.6

Table 2: Performance of different systems (%)

Table 2 shows the comparison results of the four systems. SYT represents the Sumida et al. (2008) system that we implemented and tested with the same data as ours. INIT is a system based on initial classifier c^0 in bilingual co-training. We translated training data in one language by using our bilingual instance dictionary and added the translation to the existing training data in the other language like bilingual co-training did. The size of the English and Japanese training data reached 20,729 and 20,486. We trained initial classifier c^0 with the new training data. TRAN is a system based on the classifier. BICO is a system based on bilingual co-training.

For Japanese, SYT showed worse performance than that reported in Sumida et al. (2008), probably due to the difference in training data size (ours is 20,000 and Sumida et al. (2008) was 29,900). The size of the test data was also different – ours is 2,000 and Sumida et al. (2008) was 1,000.

Comparison between INIT and SYT shows the effect of SF_3–SF_5 and IF, newly introduced feature types, in hyponymy-relation classification. INIT consistently outperformed SYT, although the difference was merely around 0.5–1.8% in F_1.

BICO showed significant performance improvement (around 3.6–10.3% in F_1) over SYT, INIT, and TRAN regardless of the language. Comparison between TRAN and BICO showed that bilingual co-training is useful for enlarging the training data and that the performance gain by bilingual co-training cannot be achieved by simply translating the existing training data.

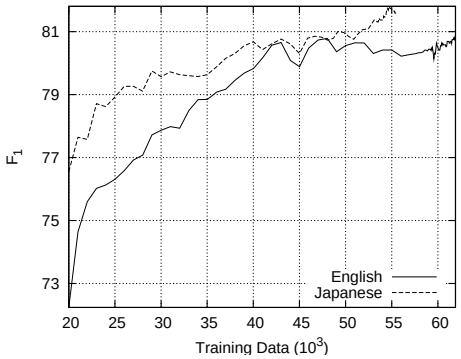


Figure 5: F_1 curves based on the increase of training data size during bilingual co-training

Figure 5 shows F_1 curves based on the size of the training data including those manually tailored and automatically obtained through bilingual co-training. The curve starts from 20,000 and ends around 55,000 in Japanese and 62,000 in English. As the training data size increases, the F_1 curves tend to go upward in both languages. This indicates that the two classifiers cooperate well to boost their performance through bilingual co-training.

We recognized 5.4 M English and 2.41 M Japanese hyponymy relations from the classification results of BICO on all hyponymy-relation candidates in both languages.

4.2 Effect of Training Data Size

We performed two tests to investigate the effect of the training data size on bilingual co-training. The first test posed the following question: "If we build $2n$ training samples by hand and the building cost is the same in both languages, which is better from the monolingual aspects: $2n$ monolingual training samples or n bilingual training samples?" Table 3 and Figure 6 show the results.

In INIT-E and INIT-J, a classifier in each language, which was trained with $2n$ monolingual training samples, did not learn through bilingual co-training. In BICO-E and BICO-J, bilingual co-training was applied to the initial classifiers trained with n training samples in both languages. As shown in Table 3, BICO, with half the size of the training samples used in INIT, always performed better than INIT in both languages. This indicates that bilingual co-training enables us to build classifiers for two languages in tandem with the same combined amount of data as required for training a single classifier in isolation while achieving superior performance.

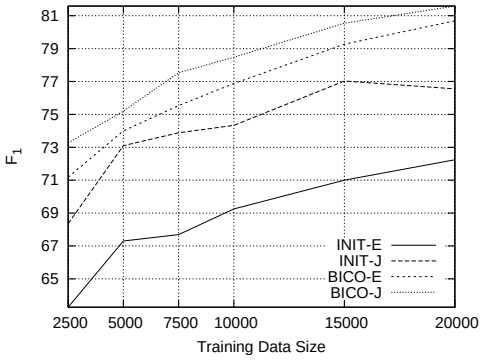


Figure 6: F_1 based on training data size: with/without bilingual co-training

n	*2n*		*n*	
	INIT-E	INIT-J	BICO-E	BICO-J
2500	67.3	72.3	70.5	73.0
5000	69.2	74.3	74.6	76.9
10000	72.2	76.6	76.9	78.6

Table 3: F_1 based on training data size: with/without bilingual co-training (%)

The second test asked: "Can we always improve performance through bilingual co-training with one strong and one weak classifier?" If the answer is yes, then we can apply our framework to acquisition of hyponymy-relations in other languages, i.e., German and French, without much effort for preparing a large amount of training data, because our strong classifier in English or Japanese can boost the performance of a weak classifier in other languages.

To answer the question, we tested the performance of classifiers by using all training data (20,000) for a strong classifier and by changing the training data size of the other from 1,000 to 15,000 ($\{$1,000, 5,000, 10,000, 15,000$\}$) for a weak classifier.

	INIT-E	BICO-E	INIT-J	BICO-J
1,000	72.2	79.6	64.0	72.7
5,000	72.2	79.6	73.1	75.3
10,000	72.2	79.8	74.3	79.0
15,000	72.2	80.4	77.0	80.1

Table 4: F_1 based on training data size: when English classifier is strong one

	INIT-E	BICO-E	INIT-J	BICO-J
1,000	60.3	69.7	76.6	79.3
5,000	67.3	74.6	76.6	79.6
10,000	69.2	77.7	76.6	80.1
15,000	71.0	79.3	76.6	80.6

Table 5: F_1 based on training data size: when Japanese classifier is strong one

Tables 4 and 5 show the results, where "INIT" represents a system based on the initial classifier in each language and "BICO" represents a system based on bilingual co-training. The results were encouraging because the classifiers showed better performance than their initial ones in every setting. In other words, a strong classifier always taught a weak classifier well, and the strong one also got help from the weak one, regardless of the size of the training data with which the weaker one learned. The test showed that bilingual co-training can work well if we have one strong classifier.

4.3 Effect of Bilingual Instance Dictionaries

We tested our method with different bilingual instance dictionaries to investigate their effect. We built bilingual instance dictionaries based on different translation dictionaries whose translation entries came from different domains (i.e., general domain, technical domain, and Wikipedia) and had a different degree of translation ambiguity. In Table 6, D1 and D2 correspond to systems based on a bilingual instance dictionary derived from two handcrafted translation dictionaries, EDICT (Breen, 2008) (a general-domain dictionary) and "The Japan Science and Technology Agency Dictionary," (a translation dictionary for technical terms) respectively. D3, which is the same as BICO in Table 2, is based on a bilingual

instance dictionary derived from Wikipedia. ENTRY represents the number of translation dictionary entries used for building a bilingual instance dictionary. E2J (or J2E) represents the average translation ambiguities of English (or Japanese) terms in the entries. To show the effect of these translation ambiguities, we used each dictionary under two different conditions, α=5 and ALL. α=5 represents the condition where only translation entries with less than five translation ambiguities are used; ALL represents no restriction on translation ambiguities.

DIC TYPE		F_1		DIC STATISTICS		
		E	J	ENTRY	E2J	J2E
D1	α=5	76.5	78.4	588K	1.80	1.77
D1	ALL	75.0	77.2	990K	7.17	2.52
D2	α=5	76.9	78.5	667K	1.89	1.55
D2	ALL	77.0	77.9	750K	3.05	1.71
D3	α=5	80.7	81.6	197K	1.03	1.02
D3	ALL	80.7	81.6	197K	1.03	1.02

Table 6: Effect of different bilingual instance dictionaries

The results showed that D3 was the best and that the performances of the others were similar to each other. The differences in the F_1 scores between α=5 and ALL were relatively small within the same system triggered by translation ambiguities. The performance gap between D3 and the other systems might explain the fact that both hyponymy-relation candidates and the translation dictionary used in D3 were extracted from the same dataset (i.e., Wikipedia), and thus the bilingual instance dictionary built with the translation dictionary in D3 had better coverage of the Wikipedia entries consisting of hyponymy-relation candidates than the other bilingual instance dictionaries. Although D1 and D2 showed lower performance than D3, the experimental results showed that bilingual co-training was always effective no matter which dictionary was used (Note that F_1 of INIT in Table 2 was 72.2 in English and 76.6 in Japanese.)

5 Related Work

Li and Li (2002) proposed bilingual bootstrapping for word translation disambiguation. Similar to bilingual co-training, classifiers for two languages cooperated in learning with bilingual resources in bilingual bootstrapping. However, the two classifiers in bilingual bootstrapping were for a bilingual task but did different tasks from the monolingual viewpoint. A classifier in each language is for word sense disambiguation, where a class label (or word sense) is different based on the languages. On the contrary, classifiers in bilingual co-training cooperate in doing the same type of tasks.

Bilingual resources have been used for monolingual tasks including verb classification and noun phrase semantic interpolation (Merlo et al., 2002; Girju, 2006). However, unlike ours, their focus was limited to bilingual features for one monolingual classifier based on supervised learning.

Recently, there has been increased interest in semantic relation acquisition from corpora. Some regarded Wikipedia as the corpora and applied hand-crafted or machine-learned rules to acquire semantic relations (Herbelot and Copestake, 2006; Kazama and Torisawa, 2007; Ruiz-casado et al., 2005; Nastase and Strube, 2008; Sumida et al., 2008; Suchanek et al., 2007). Several researchers who participated in SemEval-07 (Girju et al., 2007) proposed methods for the classification of semantic relations between simple nominals in English sentences. However, the previous work seldom considered the bilingual aspect of semantic relations in the acquisition of monolingual semantic relations.

6 Conclusion

We proposed a bilingual co-training approach and applied it to hyponymy-relation acquisition from Wikipedia. Experiments showed that bilingual co-training is effective for improving the performance of classifiers in both languages. We further showed that bilingual co-training enables us to build classifiers for two languages in tandem, outperforming classifiers trained individually for each language while requiring no more training data in total than a single classifier trained in isolation.

We showed that bilingual co-training is also helpful for boosting the performance of a weak classifier in one language with the help of a strong classifier in the other language without lowering the performance of either classifier. This indicates that the framework can reduce the cost of preparing training data in new languages with the help of our English and Japanese strong classifiers. Our future work focuses on this issue.

References

Sören Auer and Jens Lehmann. 2007. What have Innsbruck and Leipzig in common? Extracting semantics from wiki content. In *Proc. of the 4th European Semantic Web Conference (ESWC 2007)*, pages 503–517. Springer.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100.

Jim Breen. 2008. EDICT Japanese/English dictionary file, The Electronic Dictionary Research and Development Group, Monash University.

Hal Daumé III, John Langford, and Daniel Marcu. 2005. Search-based structured prediction as classification. In *Proc. of NIPS Workshop on Advances in Structured Learning for Text and Speech Processing*, Whistler, Canada.

Maike Erdmann, Kotaro Nakayama, Takahiro Hara, and Shojiro Nishio. 2008. A bilingual dictionary extracted from the Wikipedia link structure. In *Proc. of DASFAA*, pages 686–689.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. Semeval-2007 task 04: Classification of semantic relations between nominals. In *Proc. of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 13–18.

Roxana Girju. 2006. Out-of-context noun phrase semantic interpretation with cross-linguistic evidence. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 268–276.

Aurelie Herbelot and Ann Copestake. 2006. Acquiring ontological relationships from Wikipedia using RMRS. In *Proc. of the ISWC 2006 Workshop on Web Content Mining with Human Language Technologies*.

Jun'ichi Kazama and Kentaro Torisawa. 2007. Exploiting Wikipedia as external knowledge for named entity recognition. In *Proc. of Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 698–707.

Cong Li and Hang Li. 2002. Word translation disambiguation using bilingual bootstrapping. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 343–351.

MeCab. 2008. MeCab: Yet another part-of-speech and morphological analyzer. `http://mecab.sourceforge.net/`.

Paola Merlo, Suzanne Stevenson, Vivian Tsang, and Gianluca Allaria. 2002. A multilingual paradigm for automatic verb classification. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 207–214.

Vivi Nastase and Michael Strube. 2008. Decoding Wikipedia categories for knowledge acquisition. In *Proc. of AAAI 08*, pages 1219–1224.

Maria Ruiz-casado, Enrique Alfonseca, and Pablo Castells. 2005. Automatic extraction of semantic relationships for Wordnet by means of pattern learning from Wikipedia. In *Proc. of NLDB*, pages 67–79. Springer Verlag.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *Proc. of the 16th international conference on World Wide Web*, pages 697–706.

Asuka Sumida and Kentaro Torisawa. 2008. Hacking Wikipedia for hyponymy relation acquisition. In *Proc. of the Third International Joint Conference on Natural Language Processing (IJCNLP)*, pages 883–888, January.

Asuka Sumida, Naoki Yoshinaga, and Kentaro Torisawa. 2008. Boosting precision and recall of hyponymy relation acquisition from hierarchical layouts in Wikipedia. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*.

TinySVM. 2002. `http://chasen.org/~taku/software/TinySVM`.

Vladimir N. Vapnik. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Fei Wu and Daniel S. Weld. 2007. Autonomously semantifying Wikipedia. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 41–50.

Automatic Set Instance Extraction using the Web

Richard C. Wang
Language Technologies Institute
Carnegie Mellon University
rcwang@cs.cmu.edu

William W. Cohen
Machine Learning Department
Carnegie Mellon University
wcohen@cs.cmu.edu

Abstract

An important and well-studied problem is the production of semantic lexicons from a large corpus. In this paper, we present a system named ASIA (Automatic Set Instance Acquirer), which takes in the name of a semantic class as input (e.g., "car makers") and automatically outputs its instances (e.g., "ford", "nissan", "toyota"). ASIA is based on recent advances in web-based *set expansion* - the problem of finding all instances of a set given a small number of "seed" instances. This approach effectively exploits web resources and can be easily adapted to different languages. In brief, we use language-dependent hyponym patterns to find a noisy set of initial seeds, and then use a state-of-the-art language-independent set expansion system to expand these seeds. The proposed approach matches or outperforms prior systems on several English-language benchmarks. It also shows excellent performance on three dozen additional benchmark problems from English, Chinese and Japanese, thus demonstrating language-independence.

1 Introduction

An important and well-studied problem is the production of semantic lexicons for classes of interest; that is, the generation of all instances of a set (e.g., "apple", "orange", "banana") given a name of that set (e.g., "fruits"). This task is often addressed by linguistically analyzing very large collections of text (Hearst, 1992; Kozareva et al., 2008; Etzioni et al., 2005; Pantel and Ravichandran, 2004; Pasca, 2004), often using hand-constructed or machine-learned shallow linguistic patterns to detect *hyponym* instances. A hyponym is a word or phrase whose semantic range is included within that of another word. For example, x is a hyponym of y if x is a (kind of) y. The opposite of hyponym is *hypernym*.

	English	Chinese	Japanese
input seed	Amazing Race	豬血糕	ドラえもん
	Survivor	臭豆腐	ハローキティ
output	Big Brother	蘿蔔糕	アンパンマン
	The Mole	蚵仔煎	シナモロール
	The Apprentice	肉圓	ウルトラマン
	Project Runway	滷肉飯	スヌーピー
	The Bachelor	春捲	くまのプーさん

Figure 1: Examples of SEAL's input and output. English entities are reality TV shows, Chinese entities are popular Taiwanese foods, and Japanese entities are famous cartoon characters.

In this paper, we evaluate a novel approach to this problem, embodied in a system called ASIA[1] (Automatic Set Instance Acquirer). ASIA takes a semantic class name as input (e.g., "car makers") and automatically outputs instances (e.g., "ford", "nissan", "toyota"). Unlike prior methods, ASIA makes heavy use of tools for web-based *set expansion*. Set expansion is the task of finding all instances of a set given a small number of example (seed) instances. ASIA uses SEAL (Wang and Cohen, 2007), a language-independent web-based system that performed extremely well on a large number of benchmark sets – given three correct seeds, SEAL obtained average MAP scores in the high 90's for 36 benchmark problems, including a dozen test problems each for English, Chinese and Japanese. SEAL works well in part because it can efficiently find and process many *semi-structured* web documents containing instances of the set being expanded. Figure 1 shows some examples of SEAL's input and output.

SEAL has been recently extended to be robust to errors in its initial set of seeds (Wang et al.,

[1] http://rcwang.com/asia

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 441–449,
Suntec, Singapore, 2-7 August 2009.

2008), and to use bootstrapping to iteratively improve its performance (Wang and Cohen, 2008). These extensions allow ASIA to extract instances of sets from the Web, as follows. First, given a semantic class name (e.g., "fruits"), ASIA uses a small set of language-dependent *hyponym patterns* (e.g., "fruits such as ___") to find a large but noisy set of seed instances. Second, ASIA uses the extended version of SEAL to expand the noisy set of seeds.

ASIA's approach is motivated by the conjecture that for many natural classes, the amount of information available in semi-structured documents on the Web is much larger than the amount of information available in free-text documents; hence, it is natural to attempt to augment search for set instances in free-text with semi-structured document analysis. We show that ASIA performs extremely well experimentally. On the 36 benchmarks used in (Wang and Cohen, 2007), which are relatively small closed sets (e.g., countries, constellations, NBA teams), ASIA has excellent performance for both recall and precision. On four additional English-language benchmark problems (US states, countries, singers, and common fish), we compare to recent work by Kozareva, Riloff, and Hovy (Kozareva et al., 2008), and show comparable or better performance on each of these benchmarks; this is notable because ASIA requires less information than the work of Kozareva *et al* (their system requires a concept name and a seed). We also compare ASIA on twelve additional benchmarks to the extended Wordnet 2.1 produced by Snow *et al* (Snow et al., 2006), and show that for these twelve sets, ASIA produces more than five times as many set instances with much higher precision (98% versus 70%).

Another advantage of ASIA's approach is that it is nearly language-independent: since the underlying set-expansion tools are language-independent, all that is needed to support a new target language is a new set of hyponym patterns for that language. In this paper, we present experimental results for Chinese and Japanese, as well as English, to demonstrate this language-independence.

We present related work in Section 2, and explain our proposed approach for ASIA in Section 3. Section 4 presents the details of our experiments, as well as the experimental results. A comparison of results are illustrated in Section 5, and the paper concludes in Section 6.

2 Related Work

There has been a significant amount of research done in the area of semantic class learning (aka lexical acquisition, lexicon induction, hyponym extraction, or open-domain information extraction). However, to the best of our knowledge, there is not a system that can perform set instance extraction in multiple languages given only the *name* of the set.

Hearst (Hearst, 1992) presented an approach that utilizes hyponym patterns for extracting candidate instances given the name of a semantic set. The approach presented in Section 3.1 is based on this work, except that we extended it to two other languages: Chinese and Japanese.

Pantel *et al* (Pantel and Ravichandran, 2004) presented an algorithm for automatically inducing names for semantic classes and for finding their instances by using "concept signatures" (statistics on co-occuring instances). Pasca (Pasca, 2004) presented a method for acquiring named entities in arbitrary categories using lexico-syntactic extraction patterns. Etzioni *et al* (Etzioni et al., 2005) presented the KnowItAll system that also utilizes hyponym patterns to extract class instances from the Web. All the systems mentioned rely on either a English part-of-speech tagger, a parser, or both, and hence are language-dependent.

Kozareva *et al* (Kozareva et al., 2008) illustrated an approach that uses a single hyponym pattern combined with graph structures to learn semantic class from the Web. Section 5.1 shows that our approach is competitive experimentally; however, their system requires more information, as it uses the name of the semantic set and a seed instance.

Pasca (Paşca, 2007b; Paşca, 2007a) illustrated a set expansion approach that extracts instances from Web search queries given a set of input seed instances. This approach is similar in flavor to SEAL but, addresses a different task from that addressed here: for ASIA the user provides no seeds, but instead provides the *name* of the set being expanded. We compare to Pasca's system in Section 5.2.

Snow *et al* (Snow et al., 2006) use known hypernym/hyponym pairs to generate training data for a machine-learning system, which then learns many lexico-syntactic patterns. The patterns learned are based on English-language dependency parsing. We compare to Snow *et al*'s results in Section 5.3.

3 Proposed Approach

ASIA is composed of three main components: the Noisy Instance Provider, the Noisy Instance Expander, and the Bootstrapper. Given a semantic class name, the Provider extracts a initial set of noisy candidate instances using hand-coded patterns, and ranks the instances by using a simple ranking model. The Expander expands and ranks the instances using evidence from semi-structured web documents, such that irrelevant ones are ranked lower in the list. The Bootstrapper enhances the quality and completeness of the ranked list by using an unsupervised iterative technique. Note that the Expander and Bootstrapper rely on SEAL to accomplish their goals. In this section, we first describe the Noisy Instance Provider, then we briefly introduce SEAL, followed by the Noisy Instance Expander, and finally, the Bootstrapper.

3.1 Noisy Instance Provider

Noisy Instance Provider extracts candidate instances from free text (i.e., web snippets) using the methods presented in Hearst's early work (Hearst, 1992). Hearst exploited several patterns for identifying hyponymy relation (e.g., such author as Shakespeare) that many current state-of-the-art systems (Kozareva et al., 2008; Pantel and Ravichandran, 2004; Etzioni et al., 2005; Pasca, 2004) are using. However, unlike all of those systems, ASIA does not use any NLP tool (e.g., parts-of-speech tagger, parser) or rely on capitalization for extracting candidates (since we wanted ASIA to be as language-independent as possible). This leads to sets of instances that are noisy; however, we will show that set expansion and re-ranking can improve the initial sets dramatically. Below, we will refer to the initial set of noisy instances extracted by the Provider as the *initial* set.

In more detail, the Provider first constructs a few queries of hyponym phrase by using a semantic class name and a set of pre-defined hyponym patterns. For every query, the Provider retrieves a hundred snippets from Yahoo!, and splits each snippet into multiple excerpts (a snippet often contains multiple continuous excerpts from its web page). For each excerpt, the Provider extracts all chunks of characters that would then be used as candidate instances. Here, we define a *chunk* as a sequence of characters bounded by punctuation marks or the beginning and end of an excerpt.

English	Chinese	Japanese
<C> such as <I>	<C> 例如 <I>	<C> 例えば <I>
such <C> as <I>	<C> 比如 <I>	<C> たとえば <I>
<C> i.e. <I>	<C> 如 <I>	<C> と言えば <I>
<C> e.g. <I>	<C> 包含 <I>	<C> といえば <I>
<C> include <I>	<C> 包括 <I>	<I> 等の <C>
<C> including <I>	<I> 等 <C>	<I> などの <C>
<C> like <I>		<I> とかの <C>
<I> and other <C>		
<I> or other <C>		

Figure 2: Hyponym patterns in English, Chinese, and Japanese. In each pattern, <C> is a placeholder for the semantic class name and <I> is a placeholder for its instances.

Lastly, the Provider ranks each candidate instance x based on its weight assigned by the simple ranking model presented below:

$$weight(x) = \frac{sf(x, \mathbf{S})}{|\mathbf{S}|} \times \frac{ef(x, \mathbf{E})}{|\mathbf{E}|} \times \frac{wcf(x, \mathbf{E})}{|\mathbf{C}|}$$

where $\mathbf{S}$ is the set of snippets, $\mathbf{E}$ is the set of excerpts, and $\mathbf{C}$ is the set of chunks. $sf(x, \mathbf{S})$ is the snippet frequency of x (i.e., the number of snippets containing x) and $ef(x, \mathbf{E})$ is the excerpt frequency of x. Furthermore, $wcf(x, \mathbf{E})$ is the weighted chunk frequency of x, which is defined as follows:

$$wcf(x, \mathbf{E}) = \sum_{e \in \mathbf{E}} \sum_{x \in e} \frac{1}{dist(x, e) + 1}$$

where $dist(x, e)$ is the number of characters between x and the hyponym phrase in excerpt e. This model weights every occurrence of x based on the assumption that chunks closer to a hyponym phrase are usually more important than those further away. It also heavily rewards frequency, as our assumption is that the most common instances will be more useful as seeds for SEAL.

Figure 2 shows the hyponym patterns we use for English, Chinese, and Japanese. There are two types of hyponym patterns: The first type are the ones that require the class name C to precede its instance I (e.g., C such as I), and the second type are the opposite ones (e.g., I and other C). In order to reduce irrelevant chunks, when excerpts were extracted, the Provider drops all characters preceding the hyponym phrase in excerpts that contain the first type, and also drops all characters following the hyponym phrase in excerpts that contain the second type. For some semantic class names (e.g., "cmu buildings"), there are no web

documents containing any of the hyponym-phrase queries that were constructed using the name. In this case, the Provider turns to a *back-off* strategy which simply treats the semantic class name as the hyponym phrase and extracts/ranks all chunks co-occurring with the class name in the excerpts.

3.2 Set Expander - SEAL

In this paper, we rely on a set expansion system named SEAL (Wang and Cohen, 2007), which stands for Set Expander for Any Language. The system accepts as input a few *seeds* of some target set S (e.g., "fruits") and automatically finds other probable instances (e.g., "apple", "banana") of S in web documents. As its name implies, SEAL is independent of document languages: both the written (e.g., English) and the markup language (e.g., HTML). SEAL is a research system that has shown good performance in published results (Wang and Cohen, 2007; Wang et al., 2008; Wang and Cohen, 2008). Figure 1 shows some examples of SEAL's input and output.

In more detail, SEAL contains three major components: the Fetcher, Extractor, and Ranker. The *Fetcher* is responsible for fetching web documents, and the URLs of the documents come from top results retrieved from the search engine using the concatenation of all seeds as the query. This ensures that every fetched web page contains all seeds. The *Extractor* automatically constructs "wrappers" (i.e. page-specific extraction rules) for each page that contains the seeds. Every wrapper comprises two character strings that specify the left and right contexts necessary for extracting candidate instances. These contextual strings are maximally-long contexts that bracket at least one occurrence of every seed string on a page. All other candidate instances bracketed by these contextual strings derived from a particular page are extracted from the same page.

After the candidates are extracted, the *Ranker* constructs a graph that models all the relations between documents, wrappers, and candidate instances. Figure 3 shows an example graph where each node d_i represents a document, w_i a wrapper, and m_i a candidate instance. The Ranker performs Random Walk with Restart (Tong et al., 2006) on this graph (where the initial "restart" set is the set of seeds) until all node weights converge, and then ranks nodes by their final score; thus nodes are weighted higher if they are connected to many seed nodes by many short, low fan-out paths. The final expanded set contains all candidate instance nodes, ranked by their weights in the graph.

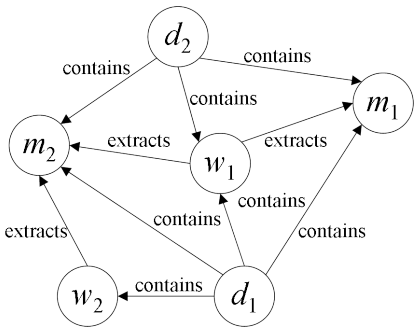


Figure 3: An example graph constructed by SEAL. Every edge from node x to y actually has an inverse relation edge from node y to x that is not shown here (e.g., m_1 *is extracted by* w_1).

3.3 Noisy Instance Expander

Wang (Wang et al., 2008) illustrated that it is feasible to perform set expansion on noisy input seeds. The paper showed that the noisy output of any Question Answering system for list questions can be improved by using a noise-resistant version of SEAL (An example of a list question is "Who were the husbands of Heddy Lamar?"). Since the initial set of candidate instances obtained using Hearst's method are noisy, the Expander expands them by performing multiple iterations of set expansion using the noise-resistant SEAL.

For every iteration, the Expander performs set expansion on a static collection of web pages. This collection is pre-fetched by querying Google and Yahoo! using the input class name and words such as "list", "names", "famous", and "common" for discovering web pages that might contain lists of the input class. In the first iteration, the Expander expands instances with scores of at least k in the initial set. In every upcoming iteration, it expands instances obtained in the last iteration that have scores of at least k and that also exist in the initial set. We have determined k to be 0.4 based on our development set[2]. This process repeats until the set of seeds for i^{th} iteration is identical to that of $(i-1)^{th}$ iteration.

There are several differences between the original SEAL and the noise-resistant SEAL. The most important difference is the Extractor. In the origi-

[2]A collection of closed-set lists such as planets, Nobel prizes, and continents in English, Chinese and Japanese

nal SEAL, the Extractor requires the longest common contexts to bracket at least one instance of *every* seed per web page. However, when seeds are noisy, such common contexts usually do not exist. The Extractor in noise-resistant SEAL solves this problem by requiring the contexts to bracket at least one instance of *a minimum of two* seeds, rather than *every* seed. This is implemented using a trie-based method described briefly in the original SEAL paper (Wang and Cohen, 2007). In this paper, the Expander utilizes a slightly-modified version of the Extractor, which requires the contexts to bracket as many seed instances as possible. This idea is based on the assumption that irrelevant instances usually do not have common contexts; whereas relevant ones do.

3.4 Bootstrapper

Bootstrapping (Etzioni et al., 2005; Kozareva, 2006; Nadeau et al., 2006) is an unsupervised iterative process in which a system continuously consumes its own outputs to improve its own performance. Wang (Wang and Cohen, 2008) showed that it is feasible to bootstrap the results of set expansion to improve the quality of a list. The paper introduces an iterative version of SEAL called iSEAL, which expands a list in multiple iterations. In each iteration, iSEAL expands a few candidates extracted in previous iterations and aggregates statistics. The Bootstrapper utilizes iSEAL to further improve the quality of the list returned by the Expander.

In every iteration, the Bootstrapper retrieves 25 web pages by using the concatenation of three seeds as query to each of Google and Yahoo!. In the first iteration, the Bootstrapper expands randomly-selected instances returned by the Expander that exist in the initial set. In every upcoming iteration, the Bootstrapper expands randomly-selected unsupervised instances obtained in the last iteration that also exist in the initial set. This process terminates when all possible seed combinations have been consumed or five iterations[3] have been reached, whichever comes first. Notice that from iteration to iteration, statistics are aggregated by growing the graph described in Section 3.2. We perform Random Walk with Restart (Tong et al., 2006) on this graph to determine the final ranking of the extracted instances.

[3]To keep the overall runtime minimal.

4 Experiments

4.1 Datasets

We evaluated our approach using the evaluation set presented in (Wang and Cohen, 2007), which contains 36 manually constructed lists across three different languages: English, Chinese, and Japanese (12 lists per language). Each list contains all instances of a particular semantic class in a certain language, and each instance contains a set of synonyms (e.g., USA, America). There are a total of 2515 instances, with an average of 70 instances per semantic class. Figure 4 shows the datasets and their corresponding semantic class names that we use in our experiments.

4.2 Evaluation Metric

Since the output of ASIA is a ranked list of extracted instances, we choose mean average precision (MAP) as our evaluation metric. MAP is commonly used in the field of Information Retrieval for evaluating ranked lists because it is sensitive to the entire ranking and it contains both recall and precision-oriented aspects. The MAP for multiple ranked lists is simply the mean value of average precisions calculated separately for each ranked list. We define the average precision of a single ranked list as:

$$AvgPrec(L) = \frac{\sum_{r=1}^{|L|} \text{Prec}(r) \times \text{isFresh}(r)}{\text{Total \# of Correct Instances}}$$

where L is a ranked list of extracted instances, r is the rank ranging from 1 to $|L|$, Prec(r) is the precision at rank r. isFresh(r) is a binary function for ensuring that, if a list contains multiple synonyms of the same instance, we do not evaluate that instance more than once. More specifically, the function returns 1 if a) the synonym at r is correct, and b) it is the highest-ranked synonym of its instance in the list; it returns 0 otherwise.

4.3 Experimental Results

For each semantic class in our dataset, the Provider first produces a noisy list of candidate instances, using its corresponding class name shown in Figure 4. This list is then expanded by the Expander and further improved by the Bootstrapper.

We present our experimental results in Table 1. As illustrated, although the Provider performs badly, the Expander substantially improves the

English Dataset	Class Name	Chinese Dataset	Class Name	Japanese Dataset	Class Name
1. classic disney movies	classic disney movies	**13.** classic disney movies	迪士尼動畫	**25.** classic disney movies	ディズニー映画
2. constellations	constellations	**14.** constellations	星座	**26.** constellations	星座
3. countries	countries	**15.** countries	國家	**27.** countries	国
4. MLB teams	MLB teams	**16.** MLB teams	MLB 球隊	**28.** MLB teams	MLB チーム
5. NBA teams	NBA teams	**17.** NBA teams	NBA 球隊	**29.** NBA teams	NBA チーム
6. NFL teams	NFL teams	**18.** NFL teams	NFL 球隊	**30.** NFL teams	NFL チーム
7. popular car makers	car makers	**19.** popular car makers	車廠	**31.** popular car makers	自動車メーカー
8. US presidents	US presidents	**20.** US presidents	美國總統	**32.** US presidents	アメリカ大統領
9. US states	US states	**21.** US states	州	**33.** US states	アメリカの州
10. CMU buildings	CMU buildings	**22.** China dynasties	朝代	**34.** Japan emperors	天皇
11. common diseases	common diseases	**23.** China provinces	省	**35.** Japan prime ministers	総理大臣
12. periodic comets	comets	**24.** Taiwan cities	縣市	**36.** Japan provinces	県

Figure 4: The 36 datasets and their semantic class names used as inputs to ASIA in our experiments.

English Dataset				NP	Chinese Dataset				NP	Japanese Dataset				NP
#	NP	NP +BS	NP +NE	+NE +BS	#	NP	NP +BS	NP +NE	+NE +BS	#	NP	NP +BS	NP +NE	+NE +BS
1.	0.22	0.83	0.82	0.87	**13.**	0.09	0.75	0.80	0.80	**25.**	0.20	0.63	0.71	0.76
2.	0.31	1.00	1.00	1.00	**14.**	0.08	0.99	0.80	0.89	**26.**	0.20	0.40	0.90	0.96
3.	0.54	0.99	0.99	0.98	**15.**	0.29	0.66	0.84	0.91	**27.**	0.16	0.96	0.97	0.96
4.	0.48	1.00	1.00	1.00	***16.**	0.09	0.00	0.93	0.93	***28.**	0.01	0.00	0.80	0.87
5.	0.54	1.00	1.00	1.00	**17.**	0.21	0.00	1.00	1.00	**29.**	0.09	0.00	0.95	0.95
6.	0.64	0.98	1.00	1.00	***18.**	0.00	0.00	0.19	0.23	***30.**	0.02	0.00	0.73	0.73
7.	0.32	0.82	0.98	0.97	**19.**	0.11	0.90	0.68	0.89	**31.**	0.20	0.49	0.83	0.89
8.	0.41	1.00	1.00	1.00	**20.**	0.18	0.00	0.94	0.97	**32.**	0.09	0.00	0.88	0.88
9.	0.81	1.00	1.00	1.00	**21.**	0.64	1.00	1.00	1.00	**33.**	0.07	0.00	0.95	1.00
***10.**	0.00	0.00	0.00	0.00	**22.**	0.08	0.00	0.67	0.80	**34.**	0.04	0.32	0.98	0.97
11.	0.11	0.62	0.51	0.76	**23.**	0.47	1.00	1.00	1.00	**35.**	0.15	1.00	1.00	1.00
12.	0.01	0.00	0.30	0.30	**24.**	0.60	1.00	1.00	1.00	**36.**	0.20	0.90	1.00	1.00
Avg.	0.37	0.77	0.80	0.82	**Avg.**	0.24	0.52	0.82	0.87	**Avg.**	0.12	0.39	0.89	0.91

Table 1: Performance of set instance extraction for each dataset measured in MAP. NP is the Noisy Instance Provider, NE is the Noisy Instance Expander, and BS is the Bootstrapper.

quality of the initial list, and the Bootstrapper then enhances it further more. On average, the Expander improves the performance of the Provider from 37% to 80% for English, 24% to 82% for Chinese, and 12% to 89% for Japanese. The Bootstrapper then further improves the performance of the Expander to 82%, 87% and 91% respectively. In addition, the results illustrate that the Bootstrapper is also effective even without the Expander; it directly improves the performance of the Provider from 37% to 77% for English, 24% to 52% for Chinese, and 12% to 39% for Japanese.

The simple *back-off* strategy seems to be effective as well. There are five datasets (marked with * in Table 1) of which their hyponym phrases return zero web documents. For those datasets, ASIA automatically uses the *back-off* strategy described in Section 3.1. Considering only those five datasets, the Expander, on average, improves the performance of the Provider from 2% to 53% and the Bootstrapper then improves it to 55%.

5 Comparison to Prior Work

We compare ASIA's performance to the results of three previously published work. We use the best-configured ASIA (NP+NE+BS) for all comparisons, and we present the comparison results in this section.

5.1 (Kozareva et al., 2008)

Table 2 shows a comparison of our extraction performance to that of Kozareva (Kozareva et al., 2008). They report results on four tasks: US states, countries, singers, and common fish. We evaluated our results manually. The results indicate that ASIA outperforms theirs for all four datasets that they reported. Note that the input to their system is a semantic class name plus one seed instance; whereas, the input to ASIA is only the class name. In terms of system runtime, for each semantic class, Kozareva *et al* reported that their extraction process usually finished overnight; however, ASIA usually finished within a minute.

N	Kozareva	ASIA	N	Kozareva	ASIA
US States			**Countries**		
25	1.00	1.00	50	1.00	1.00
50	1.00	1.00	100	1.00	1.00
64	0.78	0.78	150	1.00	1.00
			200	0.90	0.93
			300	0.61	0.67
			323	0.57	0.62
Singers			**Common Fish**		
10	1.00	1.00	10	1.00	1.00
25	1.00	1.00	25	1.00	1.00
50	0.97	1.00	50	1.00	1.00
75	0.96	1.00	75	0.93	1.00
100	0.96	1.00	100	0.84	1.00
150	0.95	0.97	116	0.80	1.00
180	0.91	0.96			

Table 2: Set instance extraction performance compared to Kozareva *et al*. We report our precision for all semantic classes and at the same ranks reported in their work.

5.2 (Paşca, 2007b)

We compare ASIA to Pasca (Paşca, 2007b) and present comparison results in Table 3. There are ten semantic classes in his evaluation dataset, and the input to his system for each class is a set of seed entities rather than a class name. We evaluate every instance manually for each class. The results show that, on average, ASIA performs better.

However, we should emphasize that for the three classes: movie, person, and video game, ASIA did not initially converge to the correct instance list given the most natural concept name. Given "movies", ASIA returns as instances strings like "comedy", "action", "drama", and other kinds of movies. Given "video games", it returns "PSP", "Xbox", "Wii", etc. Given "people", it returns "musicians", "artists", "politicians", etc. We addressed this problem by simply re-running ASIA with a more specific class name (i.e., the first one returned); however, the result suggests that future work is needed to support automatic construction of hypernym hierarchy using semi-structured web documents.

5.3 (Snow et al., 2006)

Snow (Snow et al., 2006) has extended the WordNet 2.1 by adding thousands of entries (synsets) at a relatively high precision. They have made several versions of extended WordNet available[4]. For comparison purposes, we selected the version (+30K) that achieved the best F-score in their experiments.

[4] `http://ai.stanford.edu/~rion/swn/`

		Precision @				
Target Class	**System**	**25**	**50**	**100**	**150**	**250**
Cities	Pasca	1.00	0.96	0.88	0.84	0.75
	ASIA	1.00	1.00	0.97	0.98	0.96
Countries	Pasca	1.00	0.98	0.95	0.82	0.60
	ASIA	1.00	1.00	1.00	1.00	0.79
Drugs	Pasca	1.00	1.00	0.96	0.92	0.75
	ASIA	1.00	1.00	1.00	1.00	0.98
Food	Pasca	0.88	0.86	0.82	0.78	0.62
	ASIA	1.00	1.00	0.93	0.95	0.90
Locations	Pasca	1.00	1.00	1.00	1.00	1.00
	ASIA	1.00	1.00	1.00	1.00	1.00
Newspapers	Pasca	0.96	0.98	0.93	0.86	0.54
	ASIA	1.00	1.00	0.98	0.99	0.85
Universities	Pasca	1.00	1.00	1.00	1.00	0.99
	ASIA	1.00	1.00	1.00	1.00	1.00
Movies	Pasca	0.92	0.90	0.88	0.84	0.79
Comedy Movies	ASIA	1.00	1.00	1.00	1.00	1.00
People	Pasca	1.00	1.00	1.00	1.00	1.00
Jazz Musicians	ASIA	1.00	1.00	1.00	0.94	0.88
Video Games	Pasca	1.00	1.00	0.99	0.98	0.98
PSP Games	ASIA	1.00	1.00	1.00	0.99	0.97
	Pasca	0.98	0.97	0.94	0.90	0.80
Average	ASIA	1.00	1.00	0.99	0.98	0.93

Table 3: Set instance extraction performance compared to Pasca. We report our precision for all semantic classes and at the same ranks reported in his work.

For the experimental comparison, we focused on leaf semantic classes from the extended WordNet that have many hypernyms, so that a meaningful comparison could be made: specifically, we selected nouns that have at least three hypernyms, such that the hypernyms are the leaf nodes in the hypernym hierarchy of WordNet. Of these, 210 were extended by Snow. Preliminary experiments showed that (as in the experiments with Pasca's classes above) ASIA did not always converge to the intended meaning; to avoid this problem, we instituted a second filter, and discarded ASIA's results if the intersection of hypernyms from ASIA and WordNet constituted less than 50% of those in WordNet. About 50 of the 210 nouns passed this filter. Finally, we manually evaluated precision and recall of a randomly selected set of twelve of these 50 nouns.

We present the results in Table 4. We used a fixed cut-off score[5] of 0.3 to truncate the ranked list produced by ASIA, so that we can compute precision. Since only a few of these twelve nouns are closed sets, we cannot generally compute recall; instead, we define *relative recall* to be the ratio of correct instances to the union of correct instances from both systems. As shown in the results, ASIA has much higher precision, and much higher relative recall. When we evaluated Snow's extended WordNet, we assumed all instances that

[5] Determined from our development set.

Class Name	Snow's Wordnet (+30k) # Right	# Wrong	Prec.	Relative Recall	ASIA # Right	# Wrong	Prec.	Relative Recall
Film Directors	4	4	0.50	0.01	457	0	1.00	1.00
Manias	11	0	1.00	0.09	120	0	1.00	1.00
Canadian Provinces	10	82	0.11	1.00	10	3	0.77	1.00
Signs of the Zodiac	12	10	0.55	1.00	12	0	1.00	1.00
Roman Emperors	44	4	0.92	0.47	90	0	1.00	0.96
Academic Departments	20	0	1.00	0.67	27	0	1.00	0.90
Choreographers	23	10	0.70	0.14	156	0	1.00	0.94
Elected Officials	5	102	0.05	0.31	12	0	1.00	0.75
Double Stars	11	1	0.92	0.46	20	0	1.00	0.83
South American Countries	12	1	0.92	1.00	12	0	1.00	1.00
Prizefighters	16	4	0.80	0.23	63	1	0.98	0.89
Newspapers	20	0	1.00	0.23	71	0	1.00	0.81
Average	15.7	18.2	0.70	0.47	87.5	0.3	0.98	0.92

Table 4: Set instance extraction performance compared to Snow *et al*.

	English	Chinese	Japanese
input class	Time Travel Movies	節日	ドラマ
output	The Time Machine	母親節	プロポーズ大作戦
	Back to the Future	元旦	ホテリアー
	Time Bandits	中秋節	ガリレオ
	Somewhere in Time	端午節	山田太郎ものがたり
	Peggy Sue Got Married	聖誕節	ホタルノヒカリ
	Time After Time	清明節	ファースト・キス
	Millennium	勞動節	働きマン
	Frequency	萬聖節	山おんな壁おんな
	The Final Countdown	兒童節	ハケンの品格
	Timeline	重陽節	ハタチの恋人

Figure 5: Examples of ASIA's input and output. Input class for Chinese is "holidays" and for Japanese is "dramas".

were in the original WordNet are correct. The three incorrect instances of Canadian provinces from ASIA are actually the three Canadian territories.

6 Conclusions

In this paper, we have shown that ASIA, a SEAL-based system, extracts set instances with high precision and recall in multiple languages given only the set name. It obtains a high MAP score (87%) averaged over 36 benchmark problems in three languages (Chinese, Japanese, and English). Figure 5 shows some real examples of ASIA's input and output in those three languages. ASIA's approach is based on web-based set expansion using semi-structured documents, and is motivated by the conjecture that for many natural classes, the amount of information available in semi-structured documents on the Web is much larger than the amount of information available in free-text documents. This conjecture is given some support by our experiments: for instance, ASIA finds 457 instances of the set "film director" with perfect precision, whereas Snow *et al*'s state-of-the-art methods for extraction from free text extract only four correct instances, with only 50% precision.

ASIA's approach is also quite language-independent. By adding a few simple hyponym patterns, we can easily extend the system to support other languages. We have also shown that Hearst's method works not only for English, but also for other languages such as Chinese and Japanese. We note that the ability to construct semantic lexicons in diverse languages has obvious applications in machine translation. We have also illustrated that ASIA outperforms three other English systems (Kozareva et al., 2008; Paşca, 2007b; Snow et al., 2006), even though many of these use more input than just a semantic class name. In addition, ASIA is also quite efficient, requiring only a few minutes of computation and couple hundreds of web pages per problem.

In the future, we plan to investigate the possibility of constructing hypernym hierarchy automatically using semi-structured documents. We also plan to explore whether lexicons can be constructed using only the *back-off* method for hyponym extraction, to make ASIA completely language independent. We also wish to explore whether performance can be improved by simultaneously finding class instances in multiple languages (e.g., Chinese and English) while learning translations between the extracted instances.

7 Acknowledgments

This work was supported by the Google Research Awards program.

References

Oren Etzioni, Michael J. Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artif. Intell.*, 165(1):91–134.

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

Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of ACL-08: HLT*, pages 1048–1056, Columbus, Ohio, June. Association for Computational Linguistics.

Zornitsa Kozareva. 2006. Bootstrapping named entity recognition with automatically generated gazetteer lists. In *EACL*. The Association for Computer Linguistics.

David Nadeau, Peter D. Turney, and Stan Matwin. 2006. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In Luc Lamontagne and Mario Marchand, editors, *Canadian Conference on AI*, volume 4013 of *Lecture Notes in Computer Science*, pages 266–277. Springer.

Marius Paşca. 2007a. Organizing and searching the world wide web of facts – step two: harnessing the wisdom of the crowds. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 101–110, New York, NY, USA. ACM.

Marius Paşca. 2007b. Weakly-supervised discovery of named entities using web search queries. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 683–690, New York, NY, USA. ACM.

Patrick Pantel and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 321–328, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Marius Pasca. 2004. Acquisition of categorized named entities for web search. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 137–145, New York, NY, USA. ACM.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *ACL '06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 801–808, Morristown, NJ, USA. Association for Computational Linguistics.

Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. In *ICDM*, pages 613–622. IEEE Computer Society.

Richard C. Wang and William W. Cohen. 2007. Language-independent set expansion of named entities using the web. In *ICDM*, pages 342–350. IEEE Computer Society.

Richard C. Wang and William W. Cohen. 2008. Iterative set expansion of named entities using the web. In *ICDM*, pages 1091–1096. IEEE Computer Society.

Richard C. Wang, Nico Schlaefer, William W. Cohen, and Eric Nyberg. 2008. Automatic set expansion for list question answering. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 947–954, Honolulu, Hawaii, October. Association for Computational Linguistics.

Extracting Lexical Reference Rules from Wikipedia

Eyal Shnarch
Computer Science Department
Bar-Ilan University
Ramat-Gan 52900, Israel
shey@cs.biu.ac.il

Libby Barak
Dept. of Computer Science
University of Toronto
Toronto, Canada M5S 1A4
libbyb@cs.toronto.edu

Ido Dagan
Computer Science Department
Bar-Ilan University
Ramat-Gan 52900, Israel
dagan@cs.biu.ac.il

Abstract

This paper describes the extraction from Wikipedia of *lexical reference* rules, identifying references to term meanings triggered by other terms. We present extraction methods geared to cover the broad range of the lexical reference relation and analyze them extensively. Most extraction methods yield high precision levels, and our rule-base is shown to perform better than other automatically constructed baselines in a couple of lexical expansion and matching tasks. Our rule-base yields comparable performance to WordNet while providing largely complementary information.

1 Introduction

A most common need in applied semantic inference is to infer the meaning of a target term from other terms in a text. For example, a Question Answering system may infer the answer to a question regarding *luxury cars* from a text mentioning *Bentley*, which provides a concrete reference to the sought meaning.

Aiming to capture such lexical inferences we followed (Glickman et al., 2006), which coined the term *lexical reference* (LR) to denote references in text to the specific meaning of a target term. They further analyzed the dataset of the First Recognizing Textual Entailment Challenge (Dagan et al., 2006), which includes examples drawn from seven different application scenarios. It was found that an entailing text indeed includes a concrete reference to practically every term in the entailed (inferred) sentence.

The lexical reference relation between two terms may be viewed as a lexical inference rule, denoted *LHS* $\Rightarrow$ *RHS*. Such rule indicates that the left-hand-side term would generate a reference, in some texts, to a possible meaning of the right hand side term, as the *Bentley* $\Rightarrow$ *luxury car* example.

In the above example the LHS is a hyponym of the RHS. Indeed, the commonly used hyponymy, synonymy and some cases of the meronymy relations are special cases of lexical reference. However, lexical reference is a broader relation. For instance, the LR rule *physician* $\Rightarrow$ *medicine* may be useful to infer the topic *medicine* in a text categorization setting, while an information extraction system may utilize the rule *Margaret Thatcher* $\Rightarrow$ *United Kingdom* to infer a UK announcement from the text "*Margaret Thatcher announced*".

To perform such inferences, systems need large scale knowledge bases of LR rules. A prominent available resource is WordNet (Fellbaum, 1998), from which classical relations such as synonyms, hyponyms and some cases of meronyms may be used as LR rules. An extension to WordNet was presented by (Snow et al., 2006). Yet, available resources do not cover the full scope of lexical reference.

This paper presents the extraction of a large-scale rule base from Wikipedia designed to cover a wide scope of the lexical reference relation. As a starting point we examine the potential of definition sentences as a source for LR rules (Ide and Jean, 1993; Chodorow et al., 1985; Moldovan and Rus, 2001). When writing a concept definition, one aims to formulate a concise text that includes the most characteristic aspects of the defined concept. Therefore, a definition is a promising source for LR relations between the defined concept and the definition terms.

In addition, we extract LR rules from Wikipedia redirect and hyperlink relations. As a guideline, we focused on developing simple extraction methods that may be applicable for other Web knowledge resources, rather than focusing on Wikipedia-specific attributes. Overall, our rule base contains about 8 million candidate lexical ref-

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 450–458,
Suntec, Singapore, 2-7 August 2009.

erence rules. [1]

Extensive analysis estimated that 66% of our rules are correct, while different portions of the rule base provide varying recall-precision tradeoffs. Following further error analysis we introduce rule filtering which improves inference performance. The rule base utility was evaluated within two lexical expansion applications, yielding better results than other automatically constructed baselines and comparable results to WordNet. A combination with WordNet achieved the best performance, indicating the significant marginal contribution of our rule base.

2 Background

Many works on machine readable dictionaries utilized definitions to identify semantic relations between words (Ide and Jean, 1993). Chodorow et al. (1985) observed that the head of the defining phrase is a genus term that describes the defined concept and suggested simple heuristics to find it. Other methods use a specialized parser or a set of regular expressions tuned to a particular dictionary (Wilks et al., 1996).

Some works utilized Wikipedia to build an ontology. Ponzetto and Strube (2007) identified the subsumption (IS-A) relation from Wikipedia's category tags, while in Yago (Suchanek et al., 2007) these tags, redirect links and WordNet were used to identify instances of 14 predefined specific semantic relations. These methods depend on Wikipedia's category system. The lexical reference relation we address subsumes most relations found in these works, while our extractions are not limited to a fixed set of predefined relations.

Several works examined Wikipedia texts, rather than just its structured features. Kazama and Torisawa (2007) explores the first sentence of an article and identifies the first noun phrase following the verb *be* as a label for the article title. We reproduce this part of their work as one of our baselines. Toral and Muñoz (2007) uses all nouns in the first sentence. Gabrilovich and Markovitch (2007) utilized Wikipedia-based concepts as the basis for a high-dimensional meaning representation space.

Hearst (1992) utilized a list of patterns indicative for the hyponym relation in general texts. Snow et al. (2006) use syntactic path patterns as features for supervised hyponymy and synonymy classifiers, whose training examples are derived automatically from WordNet. They use these classifiers to suggest extensions to the WordNet hierarchy, the largest one consisting of 400K new links. Their automatically created resource is regarded in our paper as a primary baseline for comparison.

Many works addressed the more general notion of *lexical associations*, or association rules (e.g. (Ruge, 1992; Rapp, 2002)). For example, *The Beatles*, *Abbey Road* and *Sgt. Pepper* would all be considered lexically associated. However this is a rather loose notion, which only indicates that terms are semantically "related" and are likely to co-occur with each other. On the other hand, lexical reference is a special case of lexical association, which specifies concretely that a reference to the meaning of one term may be inferred from the other. For example, *Abbey Road* provides a concrete reference to *The Beatles*, enabling to infer a sentence like "*I listened to The Beatles*" from "*I listened to Abbey Road*", while it does not refer specifically to *Sgt. Pepper*.

3 Extracting Rules from Wikipedia

Our goal is to utilize the broad knowledge of Wikipedia to extract a knowledge base of lexical reference rules. Each Wikipedia article provides a definition for the concept denoted by the *title* of the article. As the most concise definition we take the first sentence of each article, following (Kazama and Torisawa, 2007). Our preliminary evaluations showed that taking the entire first paragraph as the definition rarely introduces new valid rules while harming extraction precision significantly.

Since a concept definition usually employs more general terms than the defined concept (Ide and Jean, 1993), the concept title is more likely to refer to terms in its definition rather than vice versa. Therefore the title is taken as the LHS of the constructed rule while the extracted definition term is taken as its RHS. As Wikipedia's titles are mostly noun phrases, the terms we extract as RHSs are the nouns and noun phrases in the definition. The remainder of this section describes our methods for extracting rules from the definition sentence and from additional Wikipedia information.

Be-Comp Following the general idea in (Kazama and Torisawa, 2007), we identify the *IS-A* pattern in the definition sentence by extracting nominal complements of the verb 'be', taking

[1] For download see *Textual Entailment Resource Pool* at the ACL-wiki (http://aclweb.org/aclwiki)

No.	Extraction	Rule
James Eugene "Jim" Carrey is a Canadian-American actor and comedian		
1	*Be-Comp*	*Jim Carrey* $\Rightarrow$ *Canadian-American actor*
2	*Be-Comp*	*Jim Carrey* $\Rightarrow$ *actor*
3	*Be-Comp*	*Jim Carrey* $\Rightarrow$ *comedian*
Abbey Road is an album released by The Beatles		
4	*All-N*	*Abbey Road* $\Rightarrow$ *The Beatles*
5	*Parenthesis*	*Graph* $\Rightarrow$ *mathematics*
6	*Parenthesis*	*Graph* $\Rightarrow$ *data structure*
7	*Redirect*	*CPU* $\Leftrightarrow$ *Central processing unit*
8	*Redirect*	*Receptors IgG* $\Leftrightarrow$ *Antibody*
9	*Redirect*	*Hypertension* $\Leftrightarrow$ *Elevated blood-pressure*
10	*Link*	*pet* $\Rightarrow$ *Domesticated Animal*
11	*Link*	*Gestaltist* $\Rightarrow$ *Gestalt psychology*

Table 1: Examples of rule extraction methods

them as the RHS of a rule whose LHS is the article title. While Kazama and Torisawa used a chunker, we parsed the definition sentence using Minipar (Lin, 1998b). Our initial experiments showed that parse-based extraction is more accurate than chunk-based extraction. It also enables us extracting additional rules by splitting conjoined noun phrases and by taking both the head noun and the complete base noun phrase as the RHS for separate rules (examples 1–3 in Table 1).

All-N The *Be-Comp* extraction method yields mostly hypernym relations, which do not exploit the full range of lexical references within the concept definition. Therefore, we further create rules for all head nouns and base noun phrases within the definition (example 4). An unsupervised reliability score for rules extracted by this method is investigated in Section 4.3.

Title Parenthesis A common convention in Wikipedia to disambiguate ambiguous titles is adding a descriptive term in parenthesis at the end of the title, as in *The Siren (Musical)*, *The Siren (sculpture)* and *Siren (amphibian)*. From such titles we extract rules in which the descriptive term inside the parenthesis is the RHS and the rest of the title is the LHS (examples 5–6).

Redirect As any dictionary and encyclopedia, Wikipedia contains *Redirect* links that direct different search queries to the same article, which has a canonical title. For instance, there are 86 different queries that redirect the user to *United States* (e.g. *U.S.A.*, *America*, *Yankee land*). Redirect links are hand coded, specifying that both terms refer to the same concept. We therefore generate a bidirectional entailment rule for each redirect link (examples 7–9).

Link Wikipedia texts contain hyper links to articles. For each link we generate a rule whose LHS is the linking text and RHS is the title of the linked article (examples 10–11). In this case we generate a directional rule since links do not necessarily connect semantically equivalent entities.

We note that the last three extraction methods should not be considered as Wikipedia specific, since many Web-like knowledge bases contain redirects, hyper-links and disambiguation means. Wikipedia has additional structural features such as category tags, structured summary tablets for specific semantic classes, and articles containing lists which were exploited in prior work as reviewed in Section 2.

As shown next, the different extraction methods yield different precision levels. This may allow an application to utilize only a portion of the rule base whose precision is above a desired level, and thus choose between several possible recall-precision tradeoffs.

4 Extraction Methods Analysis

We applied our rule extraction methods over a version of Wikipedia available in a database constructed by (Zesch et al., 2007)[2]. The extraction yielded about 8 million rules altogether, with over 2.4 million distinct RHSs and 2.8 million distinct LHSs. As expected, the extracted rules involve mostly named entities and specific concepts, typically covered in encyclopedias.

4.1 Judging Rule Correctness

Following the spirit of the fine-grained human evaluation in (Snow et al., 2006), we randomly sampled 800 rules from our rule-base and presented them to an annotator who judged them for correctness, according to the lexical reference notion specified above. In cases which were too difficult to judge the annotator was allowed to abstain, which happened for 20 rules. 66% of the remaining rules were annotated as correct. 200 rules from the sample were judged by another annotator for agreement measurement. The resulting Kappa score was 0.7 (substantial agreement (Landis and

[2]English version from February 2007, containing 1.6 million articles. www.ukp.tu-darmstadt.de/software/JWPL

Extraction Method	Per Method		Accumulated	
	P	Est. #Rules	P	%obtained
Redirect	0.87	1,851,384	0.87	31
Be-Comp	0.78	1,618,913	0.82	60
Parenthesis	0.71	94,155	0.82	60
Link	0.7	485,528	0.80	68
All-N	0.49	1,580,574	0.66	100

Table 2: Manual analysis: precision and estimated number of correct rules per extraction method, and precision and % of correct rules obtained of rule-sets accumulated by method.

Koch, 1997)), either when considering all the abstained rules as correct or as incorrect.

The middle columns of Table 2 present, for each extraction method, the obtained percentage of correct rules (precision) and their estimated absolute number. This number is estimated by multiplying the number of annotated correct rules for the extraction method by the sampling proportion. In total, we estimate that our resource contains 5.6 million correct rules. For comparison, Snow's published extension to WordNet[3], which covers similar types of terms but is restricted to synonyms and hyponyms, includes 400,000 relations.

The right part of Table 2 shows the performance figures for accumulated rule bases, created by adding the extraction methods one at a time in order of their precision. *% obtained* is the percentage of correct rules in each rule base out of the total number of correct rules extracted jointly by all methods (the union set).

We can see that excluding the *All-N* method all extraction methods reach quite high precision levels of 0.7-0.87, with accumulated precision of 0.8[4]. By selecting only a subset of the extraction methods, according to their precision, one can choose different recall-precision tradeoff points that suit application preferences.

The less accurate *All-N* method may be used when high recall is important, accounting for 32% of the correct rules. An examination of the paths in *All-N* reveals, beyond standard hyponymy and synonymy, various semantic relations that satisfy lexical reference, such as *Location*, *Occupation* and *Creation*, as illustrated in Table 3. Typical relations covered by *Redirect* and *Link* rules include synonyms (*NY State Trooper* ⇒ *New York State Police*), morphological derivations (*irritate* ⇒ *irritation*), different spellings or naming (*Pytagoras* ⇒ *Pythagoras*) and acronyms (*AIS* ⇒ *Alarm Indication Signal*).

[3]http://ai.stanford.edu/∼rion/swn/

[4]As a non-comparable reference, Snow's fine-grained evaluation showed a precision of 0.84 on 10K rules and 0.68 on 20K rules; however, they were interested only in the hyponym relation while we evaluate our rules according to the broader LR relation.

4.2 Error Analysis

We sampled 100 rules which were annotated as incorrect and examined the causes of errors. Figure 1 shows the distribution of error types.

Wrong NP part - The most common error (35% of the errors) is taking an inappropriate part of a noun phrase (NP) as the rule right hand side (RHS). As described in Section 3, we create two rules from each extracted NP, by taking both the head noun and the complete base NP as RHSs. While both rules are usually correct, there are cases in which the left hand side (LHS) refers to the NP as a whole but not to part of it. For example, *Margaret Thatcher* refers to *United Kingdom* but not to *Kingdom*. In Section 5 we suggest a filtering method which addresses some of these errors. Future research may exploit methods for detecting multi-words expressions.

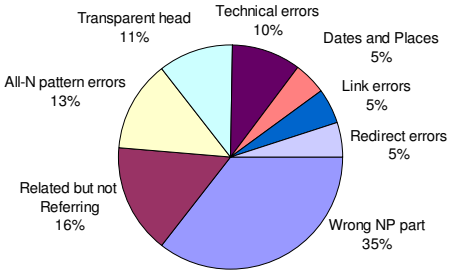


Figure 1: Error analysis: type of incorrect rules

Related but not Referring - Although all terms in a definition are highly related to the defined concept, not all are referred by it. For example the origin of a person (**The Beatles* ⇒ *Liverpool*[5]) or family ties such as 'daughter of' or 'sire of'.

All-N errors - Some of the articles start with a long sentence which may include information that is not directly referred by the title of the article. For instance, consider **Interstate 80* ⇒ *California* from "*Interstate 80 runs from California to New Jersey*". In Section 4.3 we further analyze this type of error and point at a possible direction for addressing it.

Transparent head - This is the phenomenon in which the syntactic head of a noun phrase does

[5]The asterisk denotes an incorrect rule

Relation	Rule	Path Pattern
Location	*Lovek* ⇒ *Cambodia*	*Lovek* city in *Cambodia*
Occupation	*Thomas H. Cormen* ⇒ *computer science*	*Thomas H. Cormen* professor of *computer science*
Creation	*Genocidal Healer* ⇒ *James White*	*Genocidal Healer* novel by *James White*
Origin	*Willem van Aelst* ⇒ *Dutch*	*Willem van Aelst Dutch* artist
Alias	*Dean Moriarty* ⇒ *Benjamin Linus*	*Dean Moriarty* is an alias of *Benjamin Linus* on Lost.
Spelling	*Egushawa* ⇒ *Agushaway*	*Egushawa*, also spelled *Agushaway*...

Table 3: *All-N* rules exemplifying various types of LR relations

not bear its primary meaning, while it has a modifier which serves as the semantic head (Fillmore et al., 2002; Grishman et al., 1986). Since parsers identify the syntactic head, we extract an incorrect rule in such cases. For instance, deriving **Prince William ⇒ member* instead of *Prince William ⇒ British Royal Family* from "*Prince William is a member of the British Royal Family*". Even though we implemented the common solution of using a list of typical transparent heads, this solution is partial since there is no closed set of such phrases.

Technical errors - Technical extraction errors were mainly due to erroneous identification of the title in the definition sentence or mishandling non-English texts.

Dates and Places - Dates and places where a certain person was born at, lived in or worked at often appear in definitions but do not comply to the lexical reference notion (**Galileo Galilei ⇒ 15 February 1564*).

Link errors - These are usually the result of wrong assignment of the reference direction. Such errors mostly occur when a general term, e.g. *revolution*, links to a more specific albeit typical concept, e.g. *French Revolution*.

Redirect errors - These may occur in some cases in which the extracted rule is not bidirectional. E.g. **Anti-globalization ⇒ Movement of Movements* is wrong but the opposite entailment direction is correct, as *Movement of Movements* is a popular term in Italy for *Anti-globalization*.

4.3 Scoring All-N Rules

We observed that the likelihood of nouns mentioned in a definition to be referred by the concept title depends greatly on the syntactic path connecting them (which was exploited also in (Snow et al., 2006)). For instance, the path produced by Minipar for example 4 in Table 1 is *title* $\overset{subj}{\longleftarrow}$album$\overset{vrel}{\longrightarrow}$released $\overset{by-subj}{\longrightarrow}by\overset{pcomp-n}{\longrightarrow}$ *noun*.

In order to estimate the likelihood that a syntactic path indicates lexical reference we collected from Wikipedia all paths connecting a title to a noun phrase in the definition sentence. We note that since there is no available resource which covers the full breadth of lexical reference we could not obtain sufficiently broad supervised training data for learning which paths correspond to correct references. This is in contrast to (Snow et al., 2005) which focused only on hyponymy and synonymy relations and could therefore extract positive and negative examples from WordNet.

We therefore propose the following unsupervised reference likelihood score for a syntactic path p within a definition, based on two counts: the number of times p connects an article *title* with a *noun* in its definition, denoted by $C_t(p)$, and the total number of p's occurrences in Wikipedia definitions, $C(p)$. The score of a path is then defined as $\frac{C_t(p)}{C(p)}$. The rational for this score is that $C(p) - C_t(p)$ corresponds to the number of times in which the path connects two nouns within the definition, none of which is the title. These instances are likely to be non-referring, since a concise definition typically does not contain terms that can be inferred from each other. Thus our score may be seen as an approximation for the probability that the two nouns connected by an arbitrary occurrence of the path would satisfy the reference relation. For instance, the path of example 4 obtained a score of 0.98.

We used this score to sort the set of rules extracted by the *All-N* method and split the sorted list into 3 thirds: *top*, *middle* and *bottom*. As shown in Table 4, this obtained reasonably high precision for the top third of these rules, relative to the other two thirds. This precision difference indicates that our unsupervised path score provides useful information about rule reliability.

It is worth noting that in our sample 57% of *All-N* errors, 62% of *Related but not Referring* incorrect rules and all incorrect rules of type *Dates and*

Extraction Method	Per Method		Accumulated	
	P	Est. #Rules	P	%obtained
All-N_{top}	0.60	684,238	0.76	83
All-N_{middle}	0.46	380,572	0.72	90
All-N_{bottom}	0.41	515,764	0.66	100

Table 4: Splitting *All-N* extraction method into 3 sub-types. These three rows replace the last row of Table 2

Places were extracted by the *All-N_{bottom}* method and thus may be identified as less reliable. However, this split was not observed to improve performance in the application oriented evaluations of Section 6. Further research is thus needed to fully exploit the potential of the syntactic path as an indicator for rule correctness.

5 Filtering Rules

Following our error analysis, future research is needed for addressing each specific type of error. However, during the analysis we observed that all types of erroneous rules tend to relate terms that are rather unlikely to co-occur together. We therefore suggest, as an optional filter, to recognize such rules by their co-occurrence statistics using the common Dice coefficient:

$$\frac{2 \cdot C(LHS, RHS)}{C(LHS) + C(RHS)}$$

where $C(x)$ is the number of articles in Wikipedia in which all words of x appear.

In order to partially overcome the *Wrong NP part* error, identified in Section 4.2 to be the most common error, we adjust the Dice equation for rules whose RHS is also part of a larger noun phrase (NP):

$$\frac{2 \cdot (C(LHS, RHS) - C(LHS, NP_{RHS}))}{C(LHS) + C(RHS)}$$

where NP_{RHS} is the complete NP whose part is the *RHS*. This adjustment counts only co-occurrences in which the LHS appears with the RHS alone and not with the larger NP. This substantially reduces the Dice score for those cases in which the LHS co-occurs mainly with the full NP.

Given the Dice score rules whose score does not exceed a threshold may be filtered. For example, the incorrect rule **aerial tramway* $\Rightarrow$ *car* was filtered, where the correct RHS for this LHS is the complete NP *cable car*. Another filtered rule is *magic* $\Rightarrow$ *cryptography* which is correct only for a very idiosyncratic meaning.[6]

We also examined another filtering score, the cosine similarity between the vectors representing the two rule sides in LSA (Latent Semantic Analysis) space (Deerwester et al., 1990). However, as the results with this filter resemble those for Dice we present results only for the simpler Dice filter.

6 Application Oriented Evaluations

Our primary application oriented evaluation is within an unsupervised lexical expansion scenario applied to a text categorization data set (Section 6.1). Additionally, we evaluate the utility of our rule base as a lexical resource for recognizing textual entailment (Section 6.2).

6.1 Unsupervised Text Categorization

Our categorization setting resembles typical query expansion in information retrieval (IR), where the category name is considered as the query. The advantage of using a text categorization test set is that it includes exhaustive annotation for *all* documents. Typical IR datasets, on the other hand, are partially annotated through a pooling procedure. Thus, some of our valid lexical expansions might retrieve non-annotated documents that were missed by the previously pooled systems.

6.1.1 Experimental Setting

Our categorization experiment follows a typical keywords-based text categorization scheme (McCallum and Nigam, 1999; Liu et al., 2004). Taking a lexical reference perspective, we assume that the characteristic expansion terms for a category should refer to the term (or terms) denoting the category name. Accordingly, we construct the category's feature vector by taking first the category name itself, and then expanding it with all left-hand sides of lexical reference rules whose right-hand side is the category name. For example, the category "Cars" is expanded by rules such as *Ferrari F50* $\Rightarrow$ *car*. During classification cosine similarity is measured between the feature vector of the classified document and the expanded vectors of all categories. The document is assigned to the category which yields the highest similarity score, following a single-class classification approach (Liu et al., 2004).

[6] Magic was the United States codename for intelligence derived from cryptanalysis during World War II.

Rule Base	R	P	F_1
Baselines:			
No Expansion	0.19	0.54	0.28
WikiBL	0.19	0.53	0.28
Snow^{400K}	0.19	0.54	0.28
Lin	0.25	0.39	0.30
WordNet	0.30	0.47	0.37
Extraction Methods from Wikipedia:			
Redirect + Be-Comp	0.22	**0.55**	0.31
All rules	0.31	0.38	0.34
All rules + Dice filter	0.31	0.49	**0.38**
Union:			
WordNet + Wiki$_{All\_rules+Dice}$	**0.35**	0.47	**0.40**

Table 5: Results of different rule bases for 20 newsgroups category name expansion

It should be noted that keyword-based text categorization systems employ various additional steps, such as bootstrapping, which generalize to multi-class settings and further improve performance. Our basic implementation suffices to evaluate comparatively the direct impact of different expansion resources on the initial classification.

For evaluation we used the test set of the "bydate" version of the 20-News Groups collection,[7] which contains 18,846 documents partitioned (nearly) evenly over the 20 categories[8].

6.1.2 Baselines Results

We compare the quality of our rule base expansions to 5 baselines (Table 5). The first avoids any expansion, classifying documents based on cosine similarity with category names only. As expected, it yields relatively high precision but low recall, indicating the need for lexical expansion.

The second baseline is our implementation of the relevant part of the Wikipedia extraction in (Kazama and Torisawa, 2007), taking the first noun after a *be* verb in the definition sentence, denoted as *WikiBL*. This baseline does not improve performance at all over no expansion.

The next two baselines employ state-of-the-art lexical resources. One uses Snow's extension to WordNet which was mentioned earlier. This resource did not yield a noticeable improvement, either over the *No Expansion* baseline or over *WordNet* when joined with its expansions. The second uses *Lin* dependency similarity, a syntactic-dependency based distributional word similarity resource described in (Lin, 1998a)[9]. We used various thresholds on the length of the expansion list derived from this resource. The best result, reported here, provides only a minor F_1 improvement over *No Expansion*, with modest recall increase and significant precision drop, as can be expected from such distributional method.

The last baseline uses *WordNet* for expansion. First we expand all the senses of each category name by their derivations and synonyms. Each obtained term is then expanded by its hyponyms, or by its meronyms if it has no hyponyms. Finally, the results are further expanded by their derivations and synonyms.[10] *WordNet* expansions improve substantially both Recall and F_1 relative to *No Expansion*, while decreasing precision.

6.1.3 Wikipedia Results

We then used for expansion different subsets of our rule base, producing alternative recall-precision tradeoffs. Table 5 presents the most interesting results. Using any subset of the rules yields better performance than any of the other automatically constructed baselines (*Lin*, *Snow* and *WikiBL*). Utilizing the most precise extraction methods of *Redirect* and *Be-Comp* yields the highest precision, comparable to *No Expansion*, but just a small recall increase. Using the entire rule base yields the highest recall, while filtering rules by the Dice coefficient (with 0.1 threshold) substantially increases precision without harming recall. With this configuration our automatically-constructed resource achieves comparable performance to the manually built *WordNet*.

Finally, since a dictionary and an encyclopedia are complementary in nature, we applied the union of *WordNet* and the filtered *Wikipedia* expansions. This configuration yields the best results: it maintains *WordNet*'s precision and adds nearly 50% to the recall increase of *WordNet* over *No Expansion*, indicating the substantial marginal contribution of *Wikipedia*. Furthermore, with the fast growth of Wikipedia the recall of our resource is expected to increase while maintaining its precision.

[7] www.ai.mit.edu/people/jrennie/20Newsgroups.

[8] The keywords used as category names are: atheism; graphic; microsoft windows; ibm,pc,hardware; mac,hardware; x11,x-windows; sale; car; motorcycle; baseball; hockey; cryptography; electronics; medicine; outer space; christian(noun & adj); gun; mideast,middle east; politics; religion

[9] Downloaded from www.cs.ualberta.ca/lindek/demos.htm

[10] We also tried expanding by the entire hyponym hierarchy and considering only the first sense of each synset, but the method described above achieved the best performance.

Category Name	Expanding Terms
Politics	opposition, coalition, whip$^{(a)}$
Cryptography	adversary, cryptosystem, key
Mac	PowerBook, Radius$^{(b)}$, Grab$^{(c)}$
Religion	heaven, creation, belief, missionary
Medicine	doctor, physician, treatment, clinical
Computer Graphics	radiosity$^{(d)}$, rendering, siggraph$^{(e)}$

Table 6: Some *Wikipedia* rules not in *WordNet*, which contributed to text categorization. (a) a legislator who enforce leadership desire (b) a hardware firm specializing in Macintosh equipment (c) a Macintosh screen capture software (d) an illumination algorithm (e) a computer graphics conference

Configuration	Accuracy	Accuracy Drop
WordNet + Wikipedia	60.0 %	-
Without WordNet	57.7 %	2.3 %
Without Wikipedia	58.9 %	1.1 %

Table 7: RTE accuracy results for ablation tests.

Table 6 illustrates few examples of useful rules that were found in *Wikipedia* but not in *WordNet*. We conjecture that in other application settings the rules extracted from Wikipedia might show even greater marginal contribution, particularly in specialized domains not covered well by WordNet. Another advantage of a resource based on Wikipedia is that it is available in many more languages than WordNet.

6.2 Recognizing Textual Entailment (RTE)

As a second application-oriented evaluation we measured the contributions of our (filtered) Wikipedia resource and WordNet to RTE inference (Giampiccolo et al., 2007). To that end, we incorporated both resources within a typical basic RTE system architecture (Bar-Haim et al., 2008). This system determines whether a text entails another sentence based on various matching criteria that detect syntactic, logical and lexical correspondences (or mismatches). Most relevant for our evaluation, lexical matches are detected when a Wikipedia rule's LHS appears in the text and its RHS in the hypothesis, or similarly when pairs of WordNet synonyms, hyponyms-hypernyms and derivations appear across the text and hypothesis. The system's weights were trained on the development set of RTE-3 and tested on RTE-4 (which included this year only a test set).

To measure the marginal contribution of the two resources we performed ablation tests, comparing the accuracy of the full system to that achieved when removing either resource. Table 7 presents the results, which are similar in nature to those obtained for text categorization. *Wikipedia* obtained a marginal contribution of 1.1%, about half of the analogous contribution of WordNet's manually-constructed information. We note that for current RTE technology it is very typical to gain just a few percents in accuracy thanks to external knowledge resources, while individual resources usually contribute around 0.5–2% (Iftene and Balahur-Dobrescu, 2007; Dinu and Wang, 2009). Some *Wikipedia* rules not in WordNet which contributed to RTE inference are *Jurassic Park* $\Rightarrow$ *Michael Crichton*, *GCC* $\Rightarrow$ *Gulf Cooperation Council*.

7 Conclusions and Future Work

We presented construction of a large-scale resource of lexical reference rules, as useful in applied lexical inference. Extensive rule-level analysis showed that different recall-precision tradeoffs can be obtained by utilizing different extraction methods. It also identified major reasons for errors, pointing at potential future improvements. We further suggested a filtering method which significantly improved performance.

Even though the resource was constructed by quite simple extraction methods, it was proven to be beneficial within two different application setting. While being an automatically built resource, extracted from a knowledge-base created for human consumption, it showed comparable performance to WordNet, which was manually created for computational purposes. Most importantly, it also provides complementary knowledge to WordNet, with unique lexical reference rules.

Future research is needed to improve resource's precision, especially for the *All-N* method. As a first step, we investigated a novel unsupervised score for rules extracted from definition sentences. We also intend to consider the rule base as a directed graph and exploit the graph structure for further rule extraction and validation.

Acknowledgments

The authors would like to thank Idan Szpektor for valuable advices. This work was partially supported by the NEGEV project (www.negev-initiative.org), the PASCAL-2 Network of Excellence of the European Community FP7-ICT-2007-1-216886 and by the Israel Science Foundation grant 1112/08.

References

Roy Bar-Haim, Jonathan Berant, Ido Dagan, Iddo Greental, Shachar Mirkin, Eyal Shnarch, and Idan Szpektor. 2008. Efficient semantic deduction and approximate matching over compact parse forests. In *Proceedings of TAC*.

Martin S. Chodorow, Roy J. Byrd, and George E. Heidorn. 1985. Extracting semantic hierarchies from a large on-line dictionary. In *Proceedings of ACL*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Lecture Notes in Computer Science*, volume 3944, pages 177–190.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

Georgiana Dinu and Rui Wang. 2009. Inference rules for recognizing textual entailment. In *Proceedings of the IWCS*.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press.

Charles J. Fillmore, Collin F. Baker, and Hiroaki Sato. 2002. Seeing arguments through transparent structures. In *Proceedings of LREC*.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of IJCAI*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of ACL-WTEP Workshop*.

Oren Glickman, Eyal Shnarch, and Ido Dagan. 2006. Lexical reference: a semantic matching subtask. In *Proceedings of EMNLP*.

Ralph Grishman, Lynette Hirschman, and Ngo Thanh Nhan. 1986. Discovery procedures for sublanguage selectional patterns: Initial experiments. *Computational Linguistics*, 12(3):205–215.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING*.

Nancy Ide and Véronis Jean. 1993. Extracting knowledge bases from machine-readable dictionaries: Have we wasted our time? In *Proceedings of KB & KS Workshop*.

Adrian Iftene and Alexandra Balahur-Dobrescu. 2007. Hypothesis transformation and semantic variability rules used in recognizing textual entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Jun'ichi Kazama and Kentaro Torisawa. 2007. Exploiting Wikipedia as external knowledge for named entity recognition. In *Proceedings of EMNLP-CoNLL*.

J. Richard Landis and Gary G. Koch. 1997. The measurements of observer agreement for categorical data. In *Biometrics*, pages 33:159–174.

Dekang Lin. 1998a. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL*.

Dekang Lin. 1998b. Dependency-based evaluation of MINIPAR. In *Proceedings of the Workshop on Evaluation of Parsing Systems at LREC*.

Bing Liu, Xiaoli Li, Wee Sun Lee, and Philip S. Yu. 2004. Text classification by labeling words. In *Proceedings of AAAI*.

Andrew McCallum and Kamal Nigam. 1999. Text classification by bootstrapping with keywords, EM and shrinkage. In *Proceedings of ACL Workshop for unsupervised Learning in NLP*.

Dan Moldovan and Vasile Rus. 2001. Logic form transformation of wordnet and its applicability to question answering. In *Proceedings of ACL*.

Simone P. Ponzetto and Michael Strube. 2007. Deriving a large scale taxonomy from wikipedia. In *Proceedings of AAAI*.

Reinhard Rapp. 2002. The computation of word associations: comparing syntagmatic and paradigmatic approaches. In *Proceedings of COLING*.

Gerda Ruge. 1992. Experiment on linguistically-based term associations. *Information Processing & Management*, 28(3):317–332.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of COLING-ACL*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge - unifying wordnet and wikipedia. In *Proceedings of WWW*.

Antonio Toral and Rafael Muñoz. 2007. A proposal to automatically build and maintain gazetteers for named entity recognition by using wikipedia. In *Proceedings of NAACL/HLT*.

Yorick A. Wilks, Brian M. Slator, and Louise M. Guthrie. 1996. *Electric words: dictionaries, computers, and meanings*. MIT Press, Cambridge, MA, USA.

Torsten Zesch, Iryna Gurevych, and Max Mühlhäuser. 2007. Analyzing and accessing wikipedia as a lexical semantic resource. In *Data Structures for Linguistic Resources and Applications*, pages 197–205.

Employing Topic Models for Pattern-based Semantic Class Discovery

Huibin Zhang[1*] **Mingjie Zhu**[2*] **Shuming Shi**[3] **Ji-Rong Wen**[3]
[1]Nankai University
[2]University of Science and Technology of China
[3]Microsoft Research Asia
{v-huibzh, v-mingjz, shumings, jrwen}@microsoft.com

Abstract

A semantic class is a collection of items (words or phrases) which have semantically peer or sibling relationship. This paper studies the employment of topic models to automatically construct semantic classes, taking as the source data a collection of *raw semantic classes* (RASCs), which were extracted by applying predefined patterns to web pages. The primary requirement (and challenge) here is dealing with multi-membership: An item may belong to multiple semantic classes; and we need to discover as many as possible the different semantic classes the item belongs to. To adopt topic models, we treat RASCs as "documents", items as "words", and the final semantic classes as "topics". Appropriate preprocessing and postprocessing are performed to improve results quality, to reduce computation cost, and to tackle the fixed-*k* constraint of a typical topic model. Experiments conducted on 40 million web pages show that our approach could yield better results than alternative approaches.

1 Introduction

Semantic class construction (Lin and Pantel, 2001; Pantel and Lin, 2002; Pasca, 2004; Shinzato and Torisawa, 2005; Ohshima et al., 2006) tries to discover the peer or sibling relationship among terms or phrases by organizing them into semantic classes. For example, {red, white, black…} is a semantic class consisting of color instances. A popular way for semantic class discovery is pattern-based approach, where predefined patterns (Table 1) are applied to a collection of web pages or an online web search engine to produce some *raw semantic classes* (abbreviated as RASCs, Table 2). RASCs cannot be treated as the ultimate semantic classes, because they are typically noisy and incomplete, as shown in Table 2. In addition, the information of one real semantic class may be distributed in lots of RASCs (R_2 and R_3 in Table 2).

Type	Pattern
SENT	NP {, NP}*{,} (and|or) {other} NP
TAG	<UL> <LI>item</LI> … <LI>item</LI> </UL>
TAG	<SELECT> <OPTION>item…<OPTION>item </SELECT>

* SENT: Sentence structure patterns; TAG: HTML Tag patterns

Table 1. Sample patterns

R_1: {gold, silver, copper, coal, iron, uranium}
R_2: {red, yellow, *color*, gold, silver, copper}
R_3: {red, green, blue, yellow}
R_4: {HTML, Text, PDF, MS Word, *Any file type*}
R_5: {*Today*, *Tomorrow*, Wednesday, Thursday, Friday, Saturday, Sunday}
R_6: {*Bush*, Iraq, *Photos*, USA, *War*}

Table 2. Sample raw semantic classes (RASCs)

This paper aims to discover high-quality semantic classes from a large collection of noisy RASCs. The primary requirement (and challenge) here is to deal with *multi-membership*, i.e., one item may belong to multiple different semantic classes. For example, the term "Lincoln" can simultaneously represent a person, a place, or a car brand name. Multi-membership is more popular than at a first glance, because quite a lot of English common words have also been borrowed as company names, places, or product names. For a given item (as a query) which belongs to multiple semantic classes, we intend to return the semantic classes separately, rather than mixing all their items together.

Existing pattern-based approaches only provide very limited support to multi-membership. For example, RASCs with the same labels (or hypernyms) are merged in (Pasca, 2004) to gen-

[*] This work was performed when the authors were interns at Microsoft Research Asia

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 459–467,
Suntec, Singapore, 2-7 August 2009.

erate the ultimate semantic classes. This is problematic, because RASCs may not have (accurate) hypernyms with them.

In this paper, we propose to use topic models to address the problem. In some topic models, a document is modeled as a mixture of hidden topics. The words of a document are generated according to the word distribution over the topics corresponding to the document (see Section 2 for details). Given a corpus, the latent topics can be obtained by a parameter estimation procedure. Topic modeling provides a formal and convenient way of dealing with multi-membership, which is our primary motivation of adopting topic models here. To employ topic models, we treat RASCs as "documents", items as "words", and the final semantic classes as "topics".

There are, however, several challenges in applying topic models to our problem. To begin with, the computation is intractable for processing a large collection of RASCs (our dataset for experiments contains 2.7 million unique RASCs extracted from 40 million web pages). Second, typical topic models require the number of topics (k) to be given. But it lacks an easy way of acquiring the ideal number of semantic classes from the source RASC collection. For the first challenge, we choose to apply topic models to the RASCs containing an item q, rather than the whole RASC collection. In addition, we also perform some preprocessing operations in which some items are discarded to further improve efficiency. For the second challenge, considering that most items only belong to a small number of semantic classes, we fix (for all items q) a topic number which is slightly larger than the number of classes an item could belong to. And then a postprocessing operation is performed to merge the results of topic models to generate the ultimate semantic classes.

Experimental results show that, our topic model approach is able to generate higher-quality semantic classes than popular clustering algorithms (e.g., K-Medoids and DBSCAN).

We make two contributions in the paper: On one hand, we find an effective way of constructing high-quality semantic classes in the pattern-based category which deals with multi-membership. On the other hand, we demonstrate, for the first time, that topic modeling can be utilized to help mining the *peer* relationship among words. In contrast, the general *related* relationship between words is extracted in existing topic modeling applications. Thus we expand the application scope of topic modeling.

2 Topic Models

In this section we briefly introduce the two widely used topic models which are adopted in our paper. Both of them model a document as a mixture of hidden topics. The words of every document are assumed to be generated via a generative probability process. The parameters of the model are estimated from a training process over a given corpus, by maximizing the likelihood of generating the corpus. Then the model can be utilized to inference a new document.

pLSI: The probabilistic Latent Semantic Indexing Model (pLSI) was introduced in Hofmann (1999), arose from Latent Semantic Indexing (Deerwester et al., 1990). The following process illustrates how to generate a document d in pLSI:

1. Pick a topic mixture distribution $p(\cdot|d)$.
2. For each word w_i in d
   a. Pick a latent topic z with the probability $p(z|d)$ for w_i
   b. Generate w_i with probability $p(w_i|z)$

So with k latent topics, the likelihood of generating a document d is

$$p(d) = \prod_i \sum_z p(w_i|z)p(z|d) \qquad (2.1)$$

LDA (Blei et al., 2003): In LDA, the topic mixture is drawn from a conjugate Dirichlet prior that remains the same for all documents (Figure 1). The generative process for each document in the corpus is,

1. Choose document length N from a Poisson distribution *Poisson(ξ)*.
2. Choose θ from a Dirichlet distribution with parameter α.
3. For each of the N words w_i.
   a. Choose a topic z from a Multinomial distribution with parameter θ.
   b. Pick a word w_i from $p(w_i|z,\beta)$.

So the likelihood of generating a document is

$$p(d) = \int_\theta p(\theta|\alpha) \prod_i \sum_z p(z|\theta)p(w_i|z,\beta)\, d\theta \qquad (2.2)$$

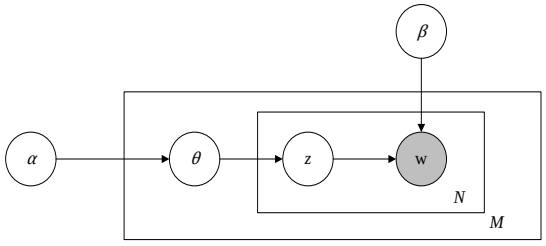


Figure 1. Graphical model representation of LDA, from Blei et al. (2003)

3 Our Approach

The source data of our approach is a collection (denoted as C_R) of RASCs extracted via applying patterns to a large collection of web pages. Given an item as an input query, the output of our approach is one or multiple semantic classes for the item. To be applicable in real-world dataset, our approach needs to be able to process at least millions of RASCs.

3.1 Main Idea

As reviewed in Section 2, topic modeling provides a formal and convenient way of grouping *documents* and *words* to *topics*. In order to apply topic models to our problem, we map RASCs to *documents*, items to *words*, and treat the output *topics* yielded from topic modeling as our semantic classes (Table 3). The motivation of utilizing topic modeling to solve our problem and building the above mapping comes from the following observations.

1) In our problem, one item may belong to multiple semantic classes; similarly in topic modeling, a word can appear in multiple topics.

2) We observe from our source data that some RASCs are comprised of items in multiple semantic classes. And at the same time, one document could be related to multiple topics in some topic models (e.g., pLSI and LDA).

Topic modeling	Semantic class construction
word	item (word or phrase)
document	RASC
topic	semantic class

Table 3. The mapping from the concepts in topic modeling to those in semantic class construction

Due to the above observations, we hope topic modeling can be employed to construct semantic classes from RASCs, just as it has been used in assigning documents and words to topics.

There are some critical challenges and issues which should be properly addressed when topic models are adopted here.

Efficiency: Our RASC collection C_R contains about 2.7 million unique RASCs and 26 million (1 million unique) items. Building topic models directly for such a large dataset may be computationally intractable. To overcome this challenge, we choose to *apply topic models to the RASCs containing a specific item* rather than the whole RASC collection. Please keep in mind that our goal in this paper is to construct the semantic classes for an item when the item is given as a query. For one item q, we denote $C_R(q)$ to be all the RASCs in C_R containing the item. We believe building a topic model over $C_R(q)$ is much more effective because it contains significantly fewer "documents", "words", and "topics". To further improve efficiency, we also perform preprocessing (refer to Section 3.4 for details) before building topic models for $C_R(q)$, where some low-frequency items are removed.

Determine the number of topics: Most topic models require the number of topics to be known beforehand[1]. However, it is not an easy task to automatically determine the exact number of semantic classes an item q should belong to. Actually the number may vary for different q. Our solution is to set (for all items q) the topic number to be a fixed value (k=5 in our experiments) which is slightly larger than the number of semantic classes most items could belong to. Then we perform postprocessing for the k topics to produce the final properly semantic classes.

In summary, our approach contains three phases (Figure 2). We build topic models for every $C_R(q)$, rather than the whole collection C_R. A preprocessing phase and a postprocessing phase are added before and after the topic modeling phase to improve efficiency and to overcome the fixed-k problem. The details of each phase are presented in the following subsections.

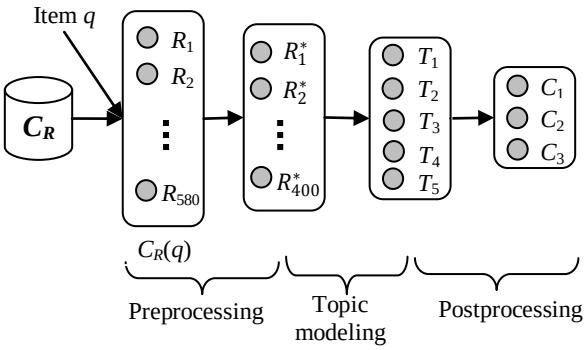


Figure 2. Main phases of our approach

3.2 Adopting Topic Models

For an item q, topic modeling is adopted to process the RASCs in $C_R(q)$ to generate k semantic classes. Here we use LDA as an example to

[1] Although there is study of non-parametric Bayesian models (Li et al., 2007) which need no prior knowledge of topic number, the computational complexity seems to exceed our efficiency requirement and we shall leave this to future work.

illustrate the process. The case of other generative topic models (e.g., pLSI) is very similar.

According to the assumption of LDA and our concept mapping in Table 3, a RASC ("document") is viewed as a mixture of hidden semantic classes ("topics"). The generative process for a RASC R in the "corpus" $C_R(q)$ is as follows,

1) Choose a RASC size (i.e., the number of items in R): $N_R \sim \text{Poisson}(\xi)$.
2) Choose a k-dimensional vector θ_R from a Dirichlet distribution with parameter α.
3) For each of the N_R items a_n:
   a) Pick a semantic class z_n from a multinomial distribution with parameter θ_R.
   b) Pick an item a_n from $p(a_n|z_n,\beta)$, where the item probabilities are parameterized by the matrix β.

There are three parameters in the model: ξ (a scalar), α (a k-dimensional vector), and β (a $k \times V$ matrix where V is the number of distinct items in $C_R(q)$). The parameter values can be obtained from a training (or called parameter estimation) process over $C_R(q)$, by maximizing the likelihood of generating the corpus. Once β is determined, we are able to compute $p(a|z,\beta)$, the probability of item a belonging to semantic class z. Therefore we can determine the members of a semantic class z by selecting those items with high $p(a|z,\beta)$ values.

The number of topics k is assumed known and fixed in LDA. As has been discussed in Section 3.1, we set a constant k value for all different $C_R(q)$. And we rely on the postprocessing phase to merge the semantic classes produced by the topic model to generate the ultimate semantic classes.

When topic modeling is used in document classification, an inference procedure is required to determine the topics for a new document. Please note that inference is not needed in our problem.

One natural question here is: Considering that in most topic modeling applications, the words within a resultant topic are typically semantically *related* but may not be in peer relationship, then what is the intuition that the resultant topics here are semantic classes rather than lists of *generally related* words? The magic lies in the "documents" we used in employing topic models. Words co-occurred in real documents tend to be semantically *related*; while items co-occurred in RASCs tend to be peers. Experimental results show that most items in the same output semantic class have peer relationship.

It might be noteworthy to mention the exchangeability or "bag-of-words" assumption in most topic models. Although the order of words in a document may be important, standard topic models neglect the order for simplicity and other reasons[2]. The order of items in a RASC is clearly much weaker than the order of words in an ordinary document. In some sense, topic models are more suitable to be used here than in processing an ordinary document corpus.

3.3 Preprocessing and Postprocessing

Preprocessing is applied to $C_R(q)$ before we build topic models for it. In this phase, we discard from all RASCs the items with frequency (i.e., the number of RASCs containing the item) less than a threshold h. A RASC itself is discarded from $C_R(q)$ if it contains less than two items after the item-removal operations. We choose to remove low-frequency items, because we found that low-frequency items are seldom important members of any semantic class for q. So the goal is to reduce the topic model training time (by reducing the training data) without sacrificing results quality too much. In the experiments section, we compare the approaches with and without preprocessing in terms of results quality and efficiency. Interestingly, experimental results show that, for some small threshold values, the results quality becomes *higher* after preprocessing is performed. We will give more discussions in Section 4.

In the postprocessing phase, the output semantic classes ("topics") of topic modeling are merged to generate the ultimate semantic classes. As indicated in Sections 3.1 and 3.2, we fix the number of topics (k=5) for different corpus $C_R(q)$ in employing topic models. For most items q, this is a larger value than the real number of semantic classes the item belongs to. As a result, one real semantic class may be divided into multiple topics. Therefore one core operation in this phase is to merge those topics into one semantic class. In addition, the items in each semantic class need to be properly ordered. Thus main operations include,

1) Merge semantic classes
2) Sort the items in each semantic class

Now we illustrate how to perform the operations.

Merge semantic classes: The merge process is performed by repeatedly calculating the simi-

[2] There are topic model extensions considering word order in documents, such as Griffiths et al. (2005).

larity between two semantic classes and merging the two ones with the highest similarity until the similarity is under a threshold. One simple and straightforward similarity measure is the Jaccard coefficient,

$$sim(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|} \quad (3.1)$$

where $C_1 \cap C_2$ and $C_1 \cup C_2$ are respectively the intersection and union of semantic classes C_1 and C_2. This formula might be over-simple, because the similarity between two different items is not exploited. So we propose the following measure,

$$sim(C_1, C_2) = \frac{\sum_{a \in C_1} \sum_{b \in C_2} sim(a, b)}{|C_1| \cdot |C_2|} \quad (3.2)$$

where $|C|$ is the number of items in semantic class C, and $sim(a,b)$ is the similarity between items a and b, which will be discussed shortly. In Section 4, we compare the performance of the above two formulas by experiments.

Sort items: We assign an importance score to every item in a semantic class and sort them according to the importance scores. Intuitively, an item should get a high rank if the average similarity between the item and the other items in the semantic class is high, and if it has high similarity to the query item q. Thus we calculate the importance of item a in a semantic class C as follows,

$$g(a|C) = \lambda \cdot sim(a,C) + (1-\lambda) \cdot sim(a,q) \quad (3.3)$$

where λ is a parameter in [0,1], $sim(a,q)$ is the similarity between a and the query item q, and $sim(a,C)$ is the similarity between a and C, calculated as,

$$sim(a, C) = \frac{\sum_{b \in C} sim(a, b)}{|C|} \quad (3.4)$$

Item similarity calculation: Formulas 3.2, 3.3, and 3.4 rely on the calculation of the similarity between two items.

One simple way of estimating item similarity is to count the number of RASCs containing both of them. We extend such an idea by distinguishing the reliability of different patterns and punishing term similarity contributions from the same site. The resultant similarity formula is,

$$sim(a, b) = \sum_{i=1}^{m} \log(1 + \sum_{j=1}^{k_i} w(P(C_{i,j}))) \quad (3.5)$$

where $C_{i,j}$ is a RASC containing both a and b, $P(C_{i,j})$ is the pattern via which the RASC is extracted, and $w(P)$ is the weight of pattern P. Assume all these RASCs belong to m sites with $C_{i,j}$ extracted from a page in site i, and k_i being the number of RASCs corresponding to site i. To determine the weight of every type of pattern, we randomly selected 50 RASCs for each pattern and labeled their quality. The weight of each kind of pattern is then determined by the average quality of all labeled RASCs corresponding to it.

The efficiency of postprocessing is not a problem, because the time cost of postprocessing is much less than that of the topic modeling phase.

3.4 Discussion

3.4.1 *Efficiency of processing popular items*

Our approach receives a query item q from users and returns the semantic classes containing the query. The maximal query processing time should not be larger than several seconds, because users would not like to wait more time. Although the average query processing time of our approach is much shorter than 1 second (see Table 4 in Section 4), it takes several minutes to process a popular item such as "Washington", because it is contained in a lot of RASCs. In order to reduce the maximal online processing time, our solution is offline processing popular items and storing the resultant semantic classes on disk. The time cost of offline processing is feasible, because we spent about 15 hours on a 4-core machine to complete the offline processing for *all* the items in our RASC collection.

3.4.2 *Alternative approaches*

One may be able to easily think of other approaches to address our problem. Here we discuss some alternative approaches which are treated as our baseline in experiments.

RASC clustering: Given a query item q, run a clustering algorithm over $C_R(q)$ and merge all RASCs in the same cluster as one semantic class. Formula 3.1 or 3.2 can be used to compute the similarity between RASCs in performing clustering. We try two clustering algorithms in experiments: K-Medoids and DBSCAN. Please note k-means cannot be utilized here because coordinates are not available for RASCs. One drawback of RASC clustering is that it cannot deal with the case of one RASC containing the items from multiple semantic classes.

Item clustering: By Formula 3.5, we are able to construct an item graph G_I to record the neighbors (in terms of similarity) of each item. Given a query item q, we first retrieve its neighbors from G_I, and then run a clustering algorithm over the neighbors. As in the case of RASC clustering, we try two clustering algorithms in experiments: K-Medoids and DBSCAN. The primary disadvantage of item clustering is that it cannot assign an item (except for the query item q) to

multiple semantic classes. As a result, when we input "gold" as the query, the item "silver" can only be assigned to one semantic class, although the term can simultaneously represents a color and a chemical element.

4 Experiments

4.1 Experimental Setup

Datasets: By using the Open Directory Project (ODP[3]) URLs as seeds, we crawled about 40 million English web pages in a breadth-first way. RASCs are extracted via applying a list of sentence structure patterns and HTML tag patterns (see Table 1 for some examples). Our RASC collection C_R contains about 2.7 million unique RASCs and 1 million distinct items.

Query set and labeling: We have volunteers to try Google Sets[4], record their queries being used, and select overall 55 queries to form our query set. For each query, the results of all approaches are mixed together and labeled by following two steps. In the first step, the *standard* (or *ideal*) *semantic classes* (SSCs) for the query are manually determined. For example, the ideal semantic classes for item "Georgia" may include Countries, and U.S. states. In the second step, each item is assigned a label of "Good", "Fair", or "Bad" with respect to each SSC. For example, "silver" is labeled "Good" with respect to "colors" and "chemical elements". We adopt metric MnDCG (Section 4.2) as our evaluation metric.

Approaches for comparison: We compare our approach with the alternative approaches discussed in Section 3.4.2.

LDA: Our approach with LDA as the topic model. The implementation of LDA is based on Blei's code of variational EM for LDA[5].

pLSI: Our approach with pLSI as the topic model. The implementation of pLSI is based on Schein, et al. (2002).

KMedoids-RASC: The *RASC clustering* approach illustrated in Section 3.4.2, with the K-Medoids clustering algorithm utilized.

DBSCAN-RASC: The *RASC clustering* approach with DBSCAN utilized.

KMedoids-Item: The *item clustering* approach with the K-Medoids utilized.

DBSCAN-Item: The *item clustering* approach with the DBSCAN clustering algorithm utilized.

K-Medoids clustering needs to predefine the cluster number *k*. We fix the *k* value for all different query item *q*, as has been done for the topic model approach. For fair comparison, the same postprocessing is made for all the approaches. And the same preprocessing is made for all the approaches except for the item clustering ones (to which the preprocessing is not applicable).

4.2 Evaluation Methodology

Each produced semantic class is an ordered list of items. A couple of metrics in the information retrieval (IR) community like Precision@10, MAP (mean average precision), and nDCG (normalized discounted cumulative gain) are available for evaluating a *single* ranked list of items per query (Croft et al., 2009). Among the metrics, nDCG (Jarvelin and Kekalainen, 2000) can handle our three-level judgments ("Good", "Fair", and "Bad", refer to Section 4.1),

$$nDCG@k = \frac{\sum_{i=1}^{k} G(i)/\log(i+1)}{\sum_{i=1}^{k} G^*(i)/\log(i+1)} \qquad (4.1)$$

where $G(i)$ is the gain value assigned to the i'th item, and $G^*(i)$ is the gain value assigned to the i'th item of an ideal (or perfect) ranking list.

Here we extend the IR metrics to the evaluation of *multiple* ordered lists per query. We use nDCG as the basic metric and extend it to MnDCG.

Assume labelers have determined m SSCs (SSC_1~SSC_m, refer to Section 4.1) for query q and the weight (or importance) of SSC_i is w_i. Assume n semantic classes are generated by an approach and n_1 of them have corresponding SSCs (i.e., no appropriate SSC can be found for the remaining n-n_1 semantic classes). We define the *MnDCG* score of an approach (with respect to query q) as,

$$MnDCG(q) = \frac{n_1}{n} \cdot \frac{\sum_{i=1}^{m} w_i \cdot Score(\mathrm{SSC}_i)}{\sum_{i=1}^{m} w_i} \qquad (4.2)$$

where

$$Score(SSC_i) = \begin{cases} 0 & if\ k_i = 0 \\ \frac{1}{k_i} \max_{j \in [1, k_i]} (nDCG(G_{i,j})) & if\ k_i \neq 0 \end{cases} \qquad (4.3)$$

In the above formula, nDCG($G_{i,j}$) is the nDCG score of semantic class $G_{i,j}$; and k_i denotes the number of semantic classes assigned to SSC_i. For a list of queries, the MnDCG score of an algorithm is the average of all scores for the queries.

The metric is designed to properly deal with the following cases,

[3] http://www.dmoz.org
[4] http://labs.google.com/sets
[5] http://www.cs.princeton.edu/~blei/lda-c/

i). One semantic class is wrongly split into multiple ones: Punished by dividing k_i in Formula 4.3;

ii). A semantic class is too noisy to be assigned to any SSC: Processed by the "n_1/n" in Formula 4.2;

iii). Fewer semantic classes (than the number of SSCs) are produced: Punished in Formula 4.3 by assigning a zero value.

iv). Wrongly merge multiple semantic classes into one: The nDCG score of the merged one will be small because it is computed with respect to only one single SSC.

The gain values of nDCG for the three relevance levels ("Bad", "Fair", and "Good") are respectively -1, 1, and 2 in experiments.

4.3 Experimental Results

4.3.1 *Overall performance comparison*

Figure 3 shows the performance comparison between the approaches listed in Section 4.1, using metrics MnDCG@*n* (*n*=1…10). Postprocessing is performed for all the approaches, where Formula 3.2 is adopted to compute the similarity between semantic classes. The results show that that the topic modeling approaches produce higher-quality semantic classes than the other approaches. It indicates that the topic mixture assumption of topic modeling can handle the multi-membership problem very well here. Among the alternative approaches, RASC clustering behaves better than item clustering. The reason might be that an item cannot belong to multiple clusters in the two item clustering approaches, while RASC clustering allows this. For the RASC clustering approaches, although one item has the chance to belong to different semantic classes, one RASC can only belong to one semantic class.

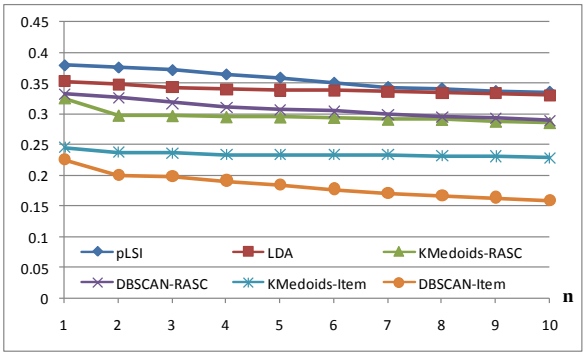


Figure 3. Quality comparison (MnDCG@n) among approaches (frequency threshold h = 4 in preprocessing; k = 5 in topic models)

4.3.2 *Preprocessing experiments*

Table 4 shows the average query processing time and results quality of the LDA approach, by varying frequency threshold *h*. Similar results are observed for the pLSI approach. In the table, *h*=1 means no preprocessing is performed. The average query processing time is calculated over all items in our dataset. As the threshold *h* increases, the processing time decreases as expected, because the input of topic modeling gets smaller. The second column lists the results quality (measured by MnDCG@10). Interestingly, we get the best results quality when *h*=4 (i.e., the items with frequency less than 4 are discarded). The reason may be that most low-frequency items are noisy ones. As a result, preprocessing can improve both results quality and processing efficiency; and *h*=4 seems a good choice in preprocessing for our dataset.

h	Avg. Query Proc. Time (seconds)	Quality (MnDCG@10)
1	0.414	0.281
2	0.375	0.294
3	0.320	0.322
4	0.268	**0.331**
5	0.232	0.328
6	0.210	0.315
7	0.197	0.315
8	0.184	0.313
9	0.173	0.288

Table 4. Time complexity and quality comparison among LDA approaches of different thresholds

4.3.3 *Postprocessing experiments*

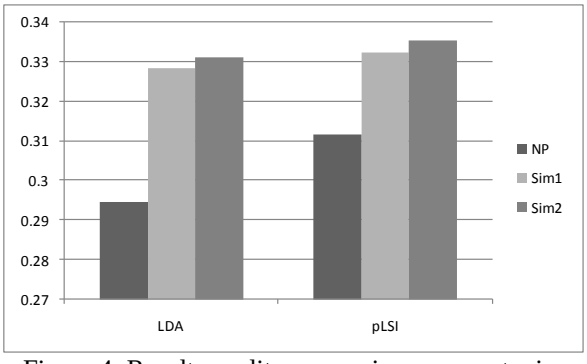


Figure 4. Results quality comparison among topic modeling approaches with and without postprocessing (metric: MnDCG@10)

The effect of postprocessing is shown in Figure 4. In the figure, NP means no postprocessing is performed. Sim1 and Sim2 respectively mean Formula 3.1 and Formula 3.2 are used in postprocessing as the similarity measure between

semantic classes. The same preprocessing (h=4) is performed in generating the data. It can be seen that postprocessing improves results quality. Sim2 achieves more performance improvement than Sim1, which demonstrates the effectiveness of the similarity measure in Formula 3.2.

4.3.4 *Sample results*

Table 5 shows the semantic classes generated by our LDA approach for some sample queries in which the bad classes or bad members are highlighted (to save space, 10 items are listed here, and the query itself is omitted in the resultant semantic classes).

Query	Semantic Classes
apple	C1: ibm, microsoft, sony, dell, toshiba, samsung, panasonic, canon, nec, sharp … C2: peach, strawberry, cherry, orange, banana, lemon, pineapple, raspberry, pear, grape …
gold	C1: silver, copper, platinum, zinc, lead, iron, nickel, tin, aluminum, manganese … C2: silver, red, black, white, blue, purple, orange, pink, brown, navy … C3: silver, platinum, earrings, diamonds, rings, bracelets, necklaces, pendants, jewelry, watches … *C4*: silver, home, money, business, metal, furniture, shoes, gypsum, hematite, fluorite …
lincoln	C1: ford, mazda, toyota, dodge, nissan, honda, bmw, chrysler, mitsubishi, audi … C2: bristol, manchester, birmingham, leeds, london, cardiff, nottingham, newcastle, sheffield, southampton … C3: jefferson, jackson, washington, madison, franklin, *sacramento*, *new york city*, monroe, *Louisville*, marion …
computer science	C1: chemistry, mathematics, physics, biology, psychology, education, history, music, business, economics …

Table 5. Semantic classes generated by our approach for some sample queries (topic model = LDA)

5 Related Work

Several categories of work are related to ours. The first category is about set expansion (i.e., retrieving one semantic class given one term or a couple of terms). Syntactic context information is used (Hindle, 1990; Ruge, 1992; Lin, 1998) to compute term similarities, based on which similar words to a particular word can directly be returned. Google sets is an online service which, given one to five items, predicts other items in the set. Ghahramani and Heller (2005) introduce a Bayesian Sets algorithm for set expansion. Set expansion is performed by feeding queries to web search engines in Wang and Cohen (2007) and Kozareva (2008). All of the above work only yields one semantic class for a given query. Second, there are pattern-based approaches in the literature which only do limited integration of RASCs (Shinzato and Torisawa, 2004; Shinzato and Torisawa, 2005; Pasca, 2004), as discussed in the introduction section. In Shi et al. (2008), an ad-hoc approach was proposed to discover the multiple semantic classes for one item. The third category is distributional similarity approaches which provide multi-membership support (Harris, 1985; Lin and Pantel, 2001; Pantel and Lin, 2002). Among them, the CBC algorithm (Pantel and Lin, 2002) addresses the multi-membership problem. But it relies on term vectors and centroids which are not available in pattern-based approaches. It is therefore not clear whether it can be borrowed to deal with multi-membership here.

Among the various applications of topic modeling, maybe the efforts of using topic model for Word Sense Disambiguation (WSD) are most relevant to our work. In Cai et al (2007), LDA is utilized to capture the global context information as the topic features for better performing the WSD task. In Boyd-Graber et al. (2007), Latent Dirichlet with WordNet (LDAWN) is developed for simultaneously disambiguating a corpus and learning the domains in which to consider each word. They do not generate semantic classes.

6 Conclusions

We presented an approach that employs topic modeling for semantic class construction. Given an item q, we first retrieve all RASCs containing the item to form a collection $C_R(q)$. Then we perform some preprocessing to $C_R(q)$ and build a topic model for it. Finally, the output semantic classes of topic modeling are post-processed to generate the final semantic classes. For the $C_R(q)$ which contains a lot of RASCs, we perform offline processing according to the above process and store the results on disk, in order to reduce the online query processing time.

We also proposed an evaluation methodology for measuring the quality of semantic classes. We show by experiments that our topic modeling approach outperforms the item clustering and RASC clustering approaches.

Acknowledgments

We wish to acknowledge help from Xiaokang Liu for mining RASCs from web pages, Changliang Wang and Zhongkai Fu for data process.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Bruce Croft, Donald Metzler, and Trevor Strohman. 2009. *Search Engines: Information Retrieval in Practice.* Addison Wesley.

Jordan Boyd-Graber, David Blei, and Xiaojin Zhu.2007. A topic model for word sense disambiguation. In *Proceedings EMNLP-CoNLL 2007*, pages 1024–1033, Prague, Czech Republic, June. Association for Computational Linguistics.

Jun Fu Cai, Wee Sun Lee, and Yee Whye Teh. 2007. NUS-ML: Improving word sense disambiguation using topic features. In *Proceedings of the International Workshop on Semantic Evaluations*, volume 4.

Scott Deerwester, Susan T. Dumais, GeorgeW. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

Zoubin Ghahramani and Katherine A. Heller. 2005. Bayesian Sets. In *Advances in Neural Information Processing Systems (NIPS05)*.

Thomas L. Griffiths, Mark Steyvers, David M. Blei,and Joshua B. Tenenbaum. 2005. Integrating topics and syntax. In *Advances in Neural Information Processing Systems* 17, pages 537–544. MIT Press

Zellig Harris. *Distributional Structure. The Philosophy of Linguistics*. New York: Oxford University Press. 1985.

Donald Hindle. 1990. Noun Classification from Predicate-Argument Structures. In *Proceedings of ACL90*, pages 268–275.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR99*, pages 50–57, New York, NY, USA. ACM.

Kalervo Jarvelin, and Jaana Kekalainen. 2000. IR Evaluation Methods for Retrieving Highly Relevant Documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR2000)*.

Zornitsa Kozareva, Ellen Riloff and Eduard Hovy. 2008. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs, In *Proceedings of ACL-08.*

Wei Li, David M. Blei, and Andrew McCallum. Nonparametric Bayes Pachinko Allocation. In *Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

Dekang Lin. 1998. Automatic Retrieval and Clustering of Similar Words. In *Proceedings of COLING-ACL98*, pages 768-774.

Dekang Lin and Patrick Pantel. 2001. Induction of Semantic Classes from Natural Language Text. In *Proceedings of SIGKDD01*, pages 317-322.

Hiroaki Ohshima, Satoshi Oyama, and Katsumi Tanaka. 2006. Searching coordinate terms with their context from the web. In *WISE*06, pages 40–47.

Patrick Pantel and Dekang Lin. 2002. Discovering Word Senses from Text. In *Proceedings of SIGKDD02*.

Marius Pasca. 2004. Acquisition of Categorized Named Entities for Web Search. In *Proc. of 2004 CIKM*.

Gerda Ruge. 1992. Experiments on Linguistically-Based Term Associations. In *Information Processing & Management, 28(3)*, pages 317-32.

Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar and David M. Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of SIGIR02*, pages 253-260.

Shuming Shi, Xiaokang Liu and Ji-Rong Wen. 2008. Pattern-based Semantic Class Discovery with Multi-Membership Support. In *CIKM2008*, pages 1453-1454.

Keiji Shinzato and Kentaro Torisawa. 2004. Acquiring Hyponymy Relations from Web Documents. In *HLT/NAACL04*, pages 73–80.

Keiji Shinzato and Kentaro Torisawa. 2005. A Simple WWW-based Method for Semantic Word Class Acquisition. In *RANLP05*.

Richard C. Wang and William W. Cohen. 2007. Langusage-Independent Set Expansion of Named Entities Using the Web. In *ICDM2007*.

Paraphrase Identification as Probabilistic Quasi-Synchronous Recognition

Dipanjan Das and **Noah A. Smith**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{dipanjan,nasmith}@cs.cmu.edu

Abstract

We present a novel approach to deciding whether two sentences hold a paraphrase relationship. We employ a generative model that generates a paraphrase of a given sentence, and we use probabilistic inference to reason about whether two sentences share the paraphrase relationship. The model cleanly incorporates both syntax and lexical semantics using quasi-synchronous dependency grammars (Smith and Eisner, 2006). Furthermore, using a product of experts (Hinton, 2002), we combine the model with a complementary logistic regression model based on state-of-the-art lexical overlap features. We evaluate our models on the task of distinguishing true paraphrase pairs from false ones on a standard corpus, giving competitive state-of-the-art performance.

1 Introduction

The problem of modeling *paraphrase* relationships between natural language utterances (McKeown, 1979) has recently attracted interest. For computational linguists, solving this problem may shed light on how best to model the semantics of sentences. For natural language engineers, the problem bears on information management systems like abstractive summarizers that must measure semantic overlap between sentences (Barzilay and Lee, 2003), question answering modules (Marsi and Krahmer, 2005) and machine translation (Callison-Burch et al., 2006).

The paraphrase identification problem asks whether two sentences have essentially the same meaning. Although paraphrase identification is defined in semantic terms, it is usually solved using statistical classifiers based on shallow lexical, n-gram, and syntactic "overlap" features. Such overlap features give the best-published classification accuracy for the paraphrase identification task (Zhang and Patrick, 2005; Finch et al., 2005; Wan et al., 2006; Corley and Mihalcea, 2005, *inter alia*), but do not explicitly model correspondence structure (or "alignment") between the parts of two sentences. In this paper, we adopt a model that posits correspondence between the words in the two sentences, defining it in *loose syntactic* terms: if two sentences are paraphrases, we expect their dependency trees to align closely, though some divergences are also expected, with some more likely than others. Following Smith and Eisner (2006), we adopt the view that the syntactic structure of sentences paraphrasing some sentence $\mathbf{s}$ should be "inspired" by the structure of $\mathbf{s}$.

Because dependency syntax is still only a crude approximation to *semantic* structure, we augment the model with a lexical semantics component, based on WordNet (Miller, 1995), that models how *words* are probabilistically altered in generating a paraphrase. This combination of loose syntax and lexical semantics is similar to the "Jeopardy" model of Wang et al. (2007).

This syntactic framework represents a major departure from useful and popular surface similarity features, and the latter are difficult to incorporate into our probabilistic model. We use a product of experts (Hinton, 2002) to bring together a logistic regression classifier built from n-gram overlap features and our syntactic model. This combined model leverages complementary strengths of the two approaches, outperforming a strong state-of-the-art baseline (Wan et al., 2006).

This paper is organized as follows. We introduce our probabilistic model in §2. The model makes use of three quasi-synchronous grammar models (Smith and Eisner, 2006, QG, hereafter) as components (one modeling paraphrase, one modeling not-paraphrase, and one a base grammar); these are detailed, along with latent-variable inference and discriminative training algorithms, in §3. We discuss the Microsoft Research Paraphrase Corpus, upon which we conduct experiments, in §4. In §5, we present experiments on paraphrase

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 468–476,
Suntec, Singapore, 2-7 August 2009.

identification with our model and make comparisons with the existing state-of-the-art. We describe the product of experts and our lexical overlap model, and discuss the results achieved in §6. We relate our approach to prior work (§7) and conclude (§8).

2 Probabilistic Model

Since our task is a classification problem, we require our model to provide an estimate of the posterior probability of the relationship (i.e., "paraphrase," denoted p, or "not paraphrase," denoted n), given the pair of sentences.[1] Here, p_Q denotes model probabilities, c is a relationship class (p or n), and $\mathbf{s}_1$ and $\mathbf{s}_2$ are the two sentences. We choose the class according to:

$$\begin{aligned}\hat{c} &= \operatorname*{argmax}_{c \in \{\mathsf{p},\mathsf{n}\}} p_Q(c \mid \mathbf{s}_1, \mathbf{s}_2) \\ &= \operatorname*{argmax}_{c \in \{\mathsf{p},\mathsf{n}\}} p_Q(c) \times p_Q(\mathbf{s}_1, \mathbf{s}_2 \mid c) \quad (1)\end{aligned}$$

We define the class-conditional probabilities of the two sentences using the following generative story. First, grammar G_0 generates a sentence $\mathbf{s}$. Then a class c is chosen, corresponding to a class-specific *probabilistic quasi-synchronous grammar* G_c. (We will discuss QG in detail in §3. For the present, consider it a specially-defined probabilistic model that generates sentences with a specific property, like "paraphrases $\mathbf{s}$," when $c = \mathsf{p}$.) Given $\mathbf{s}$, G_c generates the other sentence in the pair, $\mathbf{s}'$.

When we observe a pair of sentences $\mathbf{s}_1$ and $\mathbf{s}_2$ we do not presume to know which came first (i.e., which was $\mathbf{s}$ and which was $\mathbf{s}'$). Both orderings are assumed to be equally probable. For class c,

$$\begin{aligned}p_Q(\mathbf{s}_1, \mathbf{s}_2 \mid c) = \\ 0.5 \times p_Q(\mathbf{s}_1 \mid G_0) \times p_Q(\mathbf{s}_2 \mid G_c(\mathbf{s}_1)) \\ + \quad 0.5 \times p_Q(\mathbf{s}_2 \mid G_0) \times p_Q(\mathbf{s}_1 \mid G_c(\mathbf{s}_2)) \quad (2)\end{aligned}$$

where c can be p or n; $G_{\mathsf{p}}(\mathbf{s})$ is the QG that generates paraphrases for sentence $\mathbf{s}$, while $G_{\mathsf{n}}(\mathbf{s})$ is the QG that generates sentences that are *not* paraphrases of sentence $\mathbf{s}$. This latter model may seem counter-intuitive: since the vast majority of possible sentences are not paraphrases of $\mathbf{s}$, why is a special grammar required? Our use of a G_{n} follows from the properties of the corpus currently used for learning, in which the negative examples were selected to have high lexical overlap. We return to this point in §4.

[1] Although we do not explore the idea here, the model could be adapted for other sentence-pair relationships like entailment or contradiction.

3 QG for Paraphrase Modeling

Here, we turn to the models G_{p} and G_{n} in detail.

3.1 Background

Smith and Eisner (2006) introduced the quasi-synchronous grammar formalism. Here, we describe some of its salient aspects. The model arose out of the empirical observation that translated sentences have *some* isomorphic syntactic structure, but divergences are possible. Therefore, rather than an isomorphic structure over a pair of source and target sentences, the syntactic tree over a target sentence is modeled by a source sentence-specific grammar "inspired" by the source sentence's tree. This is implemented by associating with each node in the target tree a subset of the nodes in the source tree. Since it loosely links the two sentences' syntactic structures, QG is well suited for problems like word alignment for MT (Smith and Eisner, 2006) and question answering (Wang et al., 2007).

Consider a very simple quasi-synchronous context-free dependency grammar that generates one dependent per production rule.[2] Let $\mathbf{s} = \langle s_1, ..., s_m \rangle$ be the source sentence. The grammar rules will take one of the two forms:

$$\langle t, l \rangle \rightarrow \langle t, l \rangle \langle t', k \rangle \text{ or } \langle t, l \rangle \rightarrow \langle t', k \rangle \langle t, l \rangle$$

where t and t' range over the vocabulary of the target language, and l and $k \in \{0, ..., m\}$ are indices in the source sentence, with 0 denoting null.[3] Hard or soft constraints can be applied between l and k in a rule. These constraints imply permissible "configurations." For example, requiring $l \neq 0$ and, if $k \neq 0$ then s_k must be a child of s_l in the source tree, we can implement a synchronous dependency grammar similar to (Melamed, 2004).

Smith and Eisner (2006) used a quasi-synchronous grammar to *discover* the correspondence between words implied by the correspondence between the trees. We follow Wang et al. (2007) in treating the correspondences as latent variables, and in using a WordNet-based lexical semantics model to generate the target words.

[2] Our actual model is more complicated; see §3.2.

[3] A more general QG could allow one-to-many alignments, replacing l and k with *sets* of indices.

3.2 Detailed Model

We describe how we model $p_Q(\mathbf{t} \mid G_{\mathsf{p}}(\mathbf{s}))$ and $p_Q(\mathbf{t} \mid G_{\mathsf{n}}(\mathbf{s}))$ for source and target sentences $\mathbf{s}$ and $\mathbf{t}$ (appearing in Eq. 2 alternately as $\mathbf{s}_1$ and $\mathbf{s}_2$).

A dependency tree on a sequence $\mathbf{w} = \langle w_1, ..., w_k \rangle$ is a mapping of indices of words to indices of syntactic parents, $\tau_p : \{1, ..., k\} \rightarrow \{0, ..., k\}$, and a mapping of indices of words to dependency relation types in $\mathcal{L}$, $\tau_\ell : \{1, ..., k\} \rightarrow \mathcal{L}$. The set of indices children of w_i to its left, $\{j : \tau^{\mathbf{w}}(j) = i, j < i\}$, is denoted $\lambda^{\mathbf{w}}(i)$, and $\rho^{\mathbf{w}}(i)$ is used for right children. w_i has a single parent, denoted by $w_{\tau_p(i)}$. Cycles are not allowed, and w_0 is taken to be the dummy "wall" symbol, \$, whose only child is the root word of the sentence (normally the main verb). The label for w_i is denoted by $\tau_\ell(i)$. We denote the whole tree of a sentence $\mathbf{w}$ by $\tau^{\mathbf{w}}$, the subtree rooted at the ith word by $\tau^{\mathbf{w},i}$.

Consider two sentences: let the source sentence $\mathbf{s}$ contain m words and the target sentence $\mathbf{t}$ contain n words. Let the correspondence $x : \{1, ..., n\} \rightarrow \{0, ..., m\}$ be a mapping from indices of words in $\mathbf{t}$ to indices of words in $\mathbf{s}$. (We require each target word to map to at most one source word, though multiple target words can map to the same source word, i.e., $x(i) = x(j)$ while $i \neq j$.) When $x(i) = 0$, the ith target word maps to the wall symbol, equivalently a "null" word. Each of our QGs G_{p} and G_{n} generates the alignments x, the target tree $\tau^{\mathbf{t}}$, and the sentence $\mathbf{t}$. Both G_{p} and G_{n} are structured in the same way, differing only in their parameters; henceforth we discuss G_{p}; G_{n} is similar.

We assume that the parse trees of $\mathbf{s}$ and $\mathbf{t}$ are known.[4] Therefore our model defines:

$$\begin{aligned} p_Q(\mathbf{t} \mid G_{\mathsf{p}}(\mathbf{s})) &= p(\tau^{\mathbf{t}} \mid G_{\mathsf{p}}(\tau^{\mathbf{s}})) \\ &= \textstyle\sum_x p(\tau^{\mathbf{t}}, x \mid G_{\mathsf{p}}(\tau^{\mathbf{s}})) \quad (3) \end{aligned}$$

Because the QG is essentially a context-free dependency grammar, we can factor it into recursive steps as follows (let i be an arbitrary index in $\{1, ..., n\}$):

$$P(\tau^{\mathbf{t},i} \mid t_i, x(i), \tau^{\mathbf{s}}) = p_{val}(|\lambda^{\mathbf{t}}(i)|, |\rho^{\mathbf{t}}(i)| \mid t_i)$$
$$\times \prod_{j \in \lambda^{\mathbf{t}}(i) \cup \rho^{\mathbf{t}}(i)} \sum_{x(j)=0}^{m} P(\tau^{\mathbf{t},j} \mid t_j, x(j), \tau^{\mathbf{s}})$$
$$\times p_{kid}(t_j, \tau^{\mathbf{t}}_\ell(j), x(j) \mid t_i, x(i), \tau^{\mathbf{s}}) \quad (4)$$

where p_{val} and p_{kid} are valence and child-production probabilities parameterized as discussed in §3.4. Note the recursion in the second-to-last line.

We next describe a dynamic programming solution for calculating $p(\tau^{\mathbf{t}} \mid G_{\mathsf{p}}(\tau^{\mathbf{s}}))$. In §3.4 we discuss the parameterization of the model.

3.3 Dynamic Programming

Let $C(i, l)$ refer to the probability of $\tau^{\mathbf{t},i}$, assuming that the parent of t_i, $t_{\tau^{\mathbf{t}}_p(i)}$, is aligned to s_l. For leaves of $\tau^{\mathbf{t}}$, the base case is:

$$\begin{aligned} C(i, l) &= p_{val}(0, 0 \mid t_i) \times \quad (5) \\ &\quad \textstyle\sum_{k=0}^{m} p_{kid}(t_i, \tau^{\mathbf{t}}_\ell(i), k \mid t_{\tau^{\mathbf{t}}_p(i)}, l, \tau^{\mathbf{s}}) \end{aligned}$$

where k ranges over possible values of $x(i)$, the source-tree node to which t_i is aligned. The recursive case is:

$$\begin{aligned} C(i, l) &= p_{val}(|\lambda^{\mathbf{t}}(i)|, |\rho^{\mathbf{t}}(i)| \mid t_i) \quad (6) \\ &\quad \times \textstyle\sum_{k=0}^{m} p_{kid}(t_i, \tau^{\mathbf{t}}_\ell(i), k \mid t_{\tau^{\mathbf{t}}_p(i)}, l, \tau^{\mathbf{s}}) \\ &\quad \times \textstyle\prod_{j \in \lambda^{\mathbf{t}}(i) \cup \rho^{\mathbf{t}}(i)} C(j, k) \end{aligned}$$

We assume that the wall symbols t_0 and s_0 are aligned, so $p(\tau^{\mathbf{t}} \mid G_p(\tau^{\mathbf{s}})) = C(r, 0)$, where r is the index of the root word of the target tree $\tau^{\mathbf{t}}$. It is straightforward to show that this algorithm requires $O(m^2 n)$ runtime and $O(mn)$ space.

3.4 Parameterization

The valency distribution p_{val} in Eq. 4 is estimated in our model using the transformed treebank (see footnote 4). For unobserved cases, the conditional probability is estimated by backing off to the parent POS tag and child direction.

We discuss next how to parameterize the probability p_{kid} that appears in Equations 4, 5, and 6. This conditional distribution forms the core of our QGs, and we deviate from earlier research using QGs in defining p_{kid} in a fully generative way.

In addition to assuming that dependency parse trees for $\mathbf{s}$ and $\mathbf{t}$ are observable, we also assume each word w_i comes with POS and named entity tags. In our experiments these were obtained automatically using MXPOST (Ratnaparkhi, 1996) and BBN's Identifinder (Bikel et al., 1999).

[4] In our experiments, we use the parser described by McDonald et al. (2005), trained on sections 2–21 of the *WSJ* Penn Treebank, transformed to dependency trees following Yamada and Matsumoto (2003). (The same treebank data were also to estimate many of the parameters of our model, as discussed in the text.) Though it leads to a partial "pipeline" approximation of the posterior probability $p(c \mid \mathbf{s}, \mathbf{t})$, we believe that the relatively high quality of English dependency parsing makes this approximation reasonable.

For clarity, let $j = \tau_p^{\mathbf{t}}(i)$ and let $l = x(j)$.

$$p_{kid}(t_i, \tau_\ell^{\mathbf{t}}(i), x(i) \mid t_j, l, \tau^{\mathbf{s}}) =$$
$$p_{config}(config(t_i, t_j, s_{x(i)}, s_l) \mid t_j, l, \tau^{\mathbf{s}}) \quad (7)$$
$$\times p_{unif}(x(i) \mid config(t_i, t_j, s_{x(i)}, s_l)) \quad (8)$$
$$\times p_{lab}(\tau_\ell^{\mathbf{t}}(i) \mid config(t_i, t_j, s_{x(i)}, s_l)) \quad (9)$$
$$\times p_{pos}(pos(t_i) \mid pos(s_{x(i)})) \quad (10)$$
$$\times p_{ne}(ne(t_i) \mid ne(s_{x(i)})) \quad (11)$$
$$\times p_{lsrel}(lsrel(t_i) \mid s_{x(i)}) \quad (12)$$
$$\times p_{word}(t_i \mid lsrel(t_i), s_{x(i)}) \quad (13)$$

We consider each of the factors above in turn.

Configuration In QG, "configurations" refer to the tree relationship among source-tree nodes (above, s_l and $s_{x(i)}$) aligned to a pair of parent-child target-tree nodes (above, t_j and t_i). In deriving $\tau^{\mathbf{t},j}$, the model first chooses the configuration that will hold among t_i, t_j, $s_{x(i)}$ (which has yet to be chosen), and s_l (line 7). This is defined for configuration c log-linearly by:[5]

$$p_{config}(\mathsf{c} \mid t_j, l, \tau^{\mathbf{s}}) = \frac{\alpha_{\mathsf{c}}}{\sum_{\mathsf{c}': \exists s_k, config(t_i, t_j, s_k, s_l) = \mathsf{c}'} \alpha_{\mathsf{c}'}} \quad (14)$$

Permissible configurations in our model are shown in Table 1. These are identical to prior work (Smith and Eisner, 2006; Wang et al., 2007), except that we add a "root" configuration that aligns the target parent-child pair to null and the head word of the source sentence, respectively. Using many permissible configurations helps remove negative effects from noisy parses, which our learner treats as evidence. Fig. 1 shows some examples of major configurations that G_{p} discovers in the data.

Source tree alignment After choosing the configuration, the specific node in $\tau^{\mathbf{s}}$ that t_i will align to, $s_{x(i)}$ is drawn uniformly (line 8) from among those in the configuration selected.

Dependency label, POS, and named entity class The newly generated target word's dependency label, POS, and named entity class drawn from multinomial distributions p_{lab}, p_{pos}, and p_{ne} that condition, respectively, on the configuration and the POS and named entity class of the aligned source-tree word $s_{x(i)}$ (lines 9–11).

[5]We use log-linear models three times: for the configuration, the lexical semantics class, and the word. Each time, we are essentially assigning one weight per outcome and renormalizing among the subset of outcomes that are possible given what has been derived so far.

Configuration	Description
parent-child	$\tau_p^{\mathbf{s}}(x(i)) = x(j)$, appended with $\tau_\ell^{\mathbf{s}}(x(i))$
child-parent	$x(i) = \tau_p^{\mathbf{s}}(x(j))$, appended with $\tau_\ell^{\mathbf{s}}(x(j))$
grandparent-grandchild	$\tau_p^{\mathbf{s}}(\tau_p^{\mathbf{s}}(x(i))) = x(j)$, appended with $\tau_\ell^{\mathbf{s}}(x(i))$
siblings	$\tau_p^{\mathbf{s}}(x(i)) = \tau_p^{\mathbf{s}}(x(j)), x(i) \neq x(j)$
same-node	$x(i) = x(j)$
c-command	the parent of one source-side word is an ancestor of the other source-side word
root	$x(j) = 0, x(i)$ is the root of $\mathbf{s}$
child-null	$x(i) = 0$
parent-null	$x(j) = 0$, $x(i)$ is something other than root of $\mathbf{s}$
other	catch-all for all other types of configurations, which are permitted

Table 1: Permissible configurations. i is an index in $\mathbf{t}$ whose configuration is to be chosen; $j = \tau_p^{\mathbf{t}}(i)$ is i's parent.

WordNet relation(s) The model next chooses a lexical semantics relation between $s_{x(i)}$ and the yet-to-be-chosen word t_i (line 12). Following Wang et al. (2007),[6] we employ a 14-feature log-linear model over all logically possible combinations of the 14 WordNet relations (Miller, 1995).[7] Similarly to Eq. 14, we normalize this log-linear model based on the set of relations that are non-empty in WordNet for the word $s_{x(i)}$.

Word Finally, the target word is randomly chosen from among the set of words that bear the lexical semantic relationship just chosen (line 13). This distribution is, again, defined log-linearly:

$$p_{word}(t_i \mid lsrel(t_i) = \mathsf{R}, s_{x(i)}) = \frac{\alpha_{t_i}}{\sum_{w': s_{x(i)} \mathsf{R} w'} \alpha_{w'}} \quad (15)$$

Here α_w is the Good-Turing unigram probability estimate of a word w from the Gigaword corpus (Graff, 2003).

3.5 Base Grammar G_0

In addition to the QG that generates a second sentence bearing the desired relationship (paraphrase or not) to the first sentence $\mathbf{s}$, our model in §2 also requires a base grammar G_0 over $\mathbf{s}$.

We view this grammar as a trivial special case of the same QG model already described. G_0 assumes the empty source sentence consists only of

[6]Note that Wang et al. (2007) designed p_{kid} as an interpolation between a log-linear lexical semantics model and a word model. Our approach is more fully generative.

[7]These are: identical-word, synonym, antonym (including extended and indirect antonym), hypernym, hyponym, derived form, morphological variation (e.g., plural form), verb group, entailment, entailed-by, see-also, causal relation, whether the two words are same and is a number, and no relation.

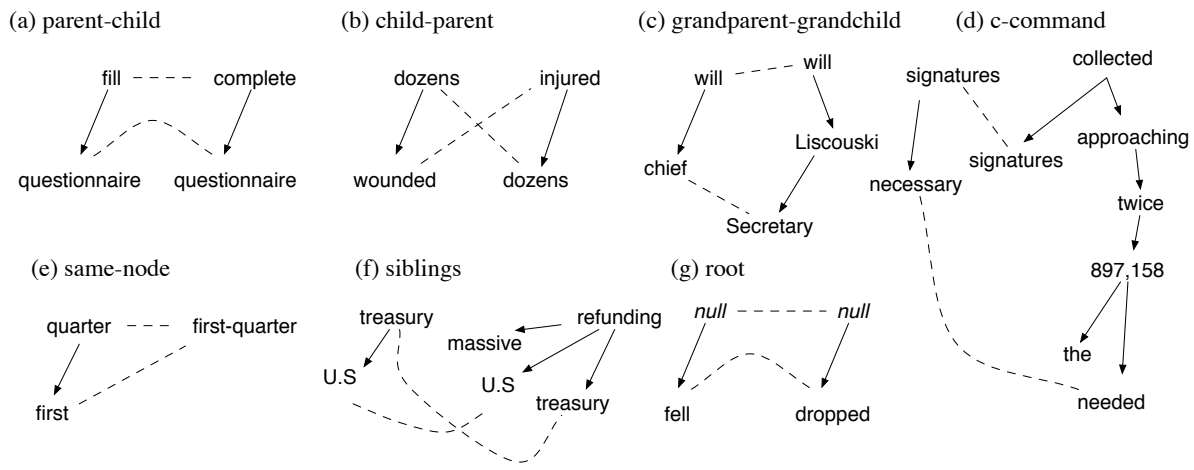


Figure 1: Some example configurations from Table 1 that G_{p} discovers in the dev. data. Directed arrows show head-modifier relationships, while dotted arrows show alignments.

a single wall node. Thus every word generated under G_0 aligns to null, and we can simplify the dynamic programming algorithm that scores a tree $\tau^{\mathbf{s}}$ under G_0:

$$\begin{aligned} C'(i) &= p_{val}(|\lambda^{\mathbf{t}}(i)|, |\rho^{\mathbf{t}}(i)| \mid s_i) \\ &\quad \times p_{lab}(\tau_\ell^{\mathbf{t}}(i)) \times p_{pos}(pos(t_i)) \times p_{ne}(ne(t_i)) \\ &\quad \times p_{word}(t_i) \times \textstyle\prod_{j:\tau^{\mathbf{t}}(j)=i} C'(j) \end{aligned} \tag{16}$$

where the final product is 1 when t_i has no children. It should be clear that $p(\mathbf{s} \mid G_0) = C'(0)$.

We estimate the distributions over dependency labels, POS tags, and named entity classes using the transformed treebank (footnote 4). The distribution over words is taken from the Gigaword corpus (as in §3.4).

It is important to note that G_0 is designed to give a smoothed estimate of the probability of a particular parsed, named entity-tagged sentence. It is never used for parsing or for generation; it is only used as a component in the generative probability model presented in §2 (Eq. 2).

3.6 Discriminative Training

Given training data $\left\langle \langle \mathbf{s}_1^{(i)}, \mathbf{s}_2^{(i)}, c^{(i)} \rangle \right\rangle_{i=1}^{N}$, we train the model discriminatively by maximizing regularized conditional likelihood:

$$\max_{\Theta} \sum_{i=1}^{N} \log \underbrace{p_Q(c^{(i)} \mid \mathbf{s}_1^{(i)}, \mathbf{s}_2^{(i)}, \Theta)}_{\text{Eq. 2 relates this to } G_{\{0,\text{p},\text{n}\}}} - C\|\Theta\|_2^2 \tag{17}$$

The parameters Θ to be learned include the class priors, the conditional distributions of the dependency labels given the various configurations, the POS tags given POS tags, the NE tags given NE tags appearing in expressions 9–11, the configuration weights appearing in Eq. 14, and the weights of the various features in the log-linear model for the lexical-semantics model. As noted, the distributions p_{val}, the word unigram weights in Eq. 15, and the parameters of the base grammar are fixed using the treebank (see footnote 4) and the Gigaword corpus.

Since there is a hidden variable (x), the objective function is non-convex. We locally optimize using the L-BFGS quasi-Newton method (Liu and Nocedal, 1989). Because many of our parameters are multinomial probabilities that are constrained to sum to one and L-BFGS is not designed to handle constraints, we treat these parameters as unnormalized weights that get renormalized (using a softmax function) before calculating the objective.

4 Data and Task

In all our experiments, we have used the Microsoft Research Paraphrase Corpus (Dolan et al., 2004; Quirk et al., 2004). The corpus contains 5,801 pairs of sentences that have been marked as "equivalent" or "not equivalent." It was constructed from thousands of news sources on the web. Dolan and Brockett (2005) remark that this corpus was created semi-automatically by first training an SVM classifier on a disjoint annotated 10,000 sentence pair dataset and then applying the SVM on an unseen 49,375 sentence pair corpus, with its output probabilities skewed towards over-identification, i.e., towards generating some false paraphrases. 5,801 out of these 49,375 pairs were randomly selected and presented to human judges for refinement into true and false paraphrases. 3,900 of the pairs were marked as having

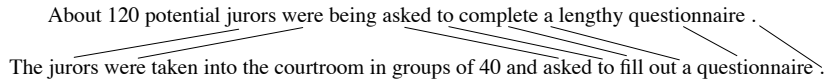


Figure 2: Discovered alignment of Ex. 19 produced by G_{p}. Observe that the model aligns identical words and also "complete" and "fill" in this specific case. This kind of alignment provides an edge over a simple lexical overlap model.

"mostly bidirectional entailment," a standard definition of the paraphrase relation. Each sentence was labeled first by two judges, who averaged 83% agreement, and a third judge resolved conflicts.

We use the standard data split into 4,076 (2,753 paraphrase, 1,323 not) training and 1,725 (1147 paraphrase, 578 not) test pairs. We reserved a randomly selected 1,075 training pairs for tuning.We cite some examples from the training set here:

(18) Revenue in the first quarter of the year dropped 15 percent from the same period a year earlier.

With the scandal hanging over Stewart's company, revenue in the first quarter of the year dropped 15 percent from the same period a year earlier.

(19) About 120 potential jurors were being asked to complete a lengthy questionnaire.

The jurors were taken into the courtroom in groups of 40 and asked to fill out a questionnaire.

Ex. 18 is a true paraphrase pair. Notice the high lexical overlap between the two sentences (unigram overlap of 100% in one direction and 72% in the other). Ex. 19 is another true paraphrase pair with much lower lexical overlap (unigram overlap of 50% in one direction and 30% in the other). Notice the use of similar-meaning phrases and irrelevant modifiers that retain the same meaning in both sentences, which a lexical overlap model cannot capture easily, but a model like a QG might. Also, in both pairs, the relationship cannot be called total bidirectional equivalence because there is some extra information in one sentence which cannot be inferred from the other.

Ex. 20 was labeled "not paraphrase":

(20) "There were a number of bureaucratic and administrative missed signals - there's not one person who's responsible here," Gehman said.

In turning down the NIMA offer, Gehman said, "there were a number of bureaucratic and administrative missed signals here.

There is significant content overlap, making a decision difficult for a naïve lexical overlap classifier. (In fact, p_Q labels this example n while the lexical overlap models label it p.)

The fact that negative examples in this corpus were selected because of their high lexical overlap is important. It means that any discriminative model is expected to learn to distinguish mere overlap from paraphrase. This seems appropriate, but it does mean that the "not paraphrase" relation ought to be denoted "not paraphrase but deceptively similar on the surface." It is for this reason that we use a special QG for the n relation.

5 Experimental Evaluation

Here we present our experimental evaluation using p_Q. We trained on the training set (3,001 pairs) and tuned model metaparameters (C in Eq. 17) and the effect of different feature sets on the development set (1,075 pairs). We report accuracy on the official MSRPC test dataset. If the posterior probability $p_Q(\mathsf{p} \mid \mathbf{s}_1, \mathbf{s}_2)$ is greater than 0.5, the pair is labeled "paraphrase" (as in Eq. 1).

5.1 Baseline

We replicated a state-of-the-art baseline model for comparison. Wan et al. (2006) report the best published accuracy, to our knowledge, on this task, using a support vector machine. Our baseline is a reimplementation of Wan et al. (2006), using features calculated directly from $\mathbf{s}_1$ and $\mathbf{s}_2$ without recourse to any hidden structure: proportion of word unigram matches, proportion of lemmatized unigram matches, BLEU score (Papineni et al., 2001), BLEU score on lemmatized tokens, F measure (Turian et al., 2003), difference of sentence length, and proportion of dependency relation overlap. The SVM was trained to classify positive and negative examples of paraphrase using SVM^{light} (Joachims, 1999).[8] Metaparameters, tuned on the development data, were the regularization constant and the degree of the polynomial kernel (chosen in $[10^{-5}, 10^{2}]$ and 1–5 respectively.).[9]

It is unsurprising that the SVM performs very well on the MSRPC because of the corpus creation process (see Sec. 4) where an SVM was applied as well, with very similar features and a skewed decision process (Dolan and Brockett, 2005).

[8] `http://svmlight.joachims.org`

[9] Our replication of the Wan et al. model is approximate, because we used different preprocessing tools: MXPOST for POS tagging (Ratnaparkhi, 1996), MSTParser for parsing (McDonald et al., 2005), and Dan Bikel's interface (`http://www.cis.upenn.edu/~dbikel/software.html#wn`) to WordNet (Miller, 1995) for lemmatization information. Tuning led to $C = 17$ and polynomial degree 4.

	Model	Accuracy	Precision	Recall
baselines	all p	66.49	66.49	100.00
	Wan et al. SVM (reported)	75.63	77.00	90.00
	Wan et al. SVM (replication)	75.42	76.88	90.14
p_Q	lexical semantics features removed	68.64	68.84	96.51
	all features	73.33	74.48	91.10
	c-command disallowed (best; see text)	73.86	74.89	91.28
§6	p_L	75.36	78.12	87.44
	product of experts	76.06	79.57	86.05
oracles	Wan et al. SVM and p_L	*80.17*	*100.00*	*92.07*
	Wan et al. SVM and p_Q	*83.42*	*100.00*	*96.60*
	p_Q and p_L	*83.19*	*100.00*	*95.29*

Table 2: Accuracy, p-class precision, and p-class recall on the test set (N = 1,725). See text for differences in implementation between Wan et al. and our replication; their reported score does not include the full test set.

5.2 Results

Tab. 2 shows performance achieved by the baseline SVM and variations on p_Q on the test set. We performed a few feature ablation studies, evaluating on the development data. We removed the lexical semantics component of the QG,[10] and disallowed the syntactic configurations one by one, to investigate which components of p_Q contributes to system performance. The lexical semantics component is critical, as seen by the drop in accuracy from the table (without this component, p_Q behaves almost like the "all p" baseline). We found that the most important configurations are "parent-child," and "child-parent" while damage from ablating other configurations is relatively small. Most interestingly, disallowing the "c-command" configuration resulted in the best absolute accuracy, giving us the best version of p_Q. The c-command configuration allows more distant nodes in a source sentence to align to parent-child pairs in a target (see Fig. 1d). Allowing this configuration guides the model in the wrong direction, thus reducing test accuracy. We tried disallowing more than one configuration at a time, without getting improvements on development data. We also tried ablating the WordNet relations, and observed that the "identical-word" feature hurt the model the most. Ablating the rest of the features did not produce considerable changes in accuracy.

The development data-selected p_Q achieves higher recall by 1 point than Wan et al.'s SVM, but has precision 2 points worse.

5.3 Discussion

It is quite promising that a linguistically-motivated probabilistic model comes so close to a string-similarity baseline, *without* incorporating string-local phrases. We see several reasons to prefer the more intricate QG to the straightforward SVM. First, the QG discovers hidden alignments between words. Alignments have been leveraged in related tasks such as textual entailment (Giampiccolo et al., 2007); they make the model more interpretable in analyzing system output (e.g., Fig. 2). Second, the paraphrases of a sentence can be considered to be monolingual translations. We model the paraphrase problem using a direct machine translation model, thus providing a translation interpretation of the problem. This framework could be extended to permit paraphrase *generation*, or to exploit other linguistic annotations, such as representations of semantics (see, e.g., Qiu et al., 2006).

Nonetheless, the usefulness of surface overlap features is difficult to ignore. We next provide an efficient way to combine a surface model with p_Q.

6 Product of Experts

Incorporating structural alignment and surface overlap features inside a single model can make exact inference infeasible. As an example, consider features like n-gram overlap percentages that provide cues of content overlap between two sentences. One intuitive way of including these features in a QG could be including these only at the root of the target tree, i.e. while calculating $C(r, 0)$. These features have to be included in estimating p_{kid}, which has log-linear component models (Eq. 7- 13). For these bigram or trigram overlap features, a similar log-linear model has to be normalized with a partition function, which considers the (unnormalized) scores of *all* possible target sentences, given the source sentence.

We therefore combine p_Q with a lexical overlap model that gives another posterior probability estimate $p_L(c \mid \mathbf{s}_1, \mathbf{s}_2)$ through a product of experts (PoE; Hinton, 2002), $p_J(c \mid \mathbf{s}_1, \mathbf{s}_2)$

$$= \frac{p_Q(c \mid \mathbf{s}_1, \mathbf{s}_2) \times p_L(c \mid \mathbf{s}_1, \mathbf{s}_2)}{\sum_{c' \in \{\mathsf{p},\mathsf{n}\}} p_Q(c' \mid \mathbf{s}_1, \mathbf{s}_2) \times p_L(c' \mid \mathbf{s}_1, \mathbf{s}_2)} \quad (21)$$

[10] This is accomplished by eliminating lines 12 and 13 from the definition of p_{kid} and redefining p_{word} to be the unigram word distribution estimated from the Gigaword corpus, as in G_0, without the help of WordNet.

Eq. 21 takes the product of the two models' posterior probabilities, then normalizes it to sum to one. PoE models are used to efficiently combine several expert models that individually constrain different dimensions in high-dimensional data, the product therefore constraining all of the dimensions. Combining models in this way grants to each expert component model the ability to "veto" a class by giving it low probability; the most probable class is the one that is least objectionable to all experts.

Probabilistic Lexical Overlap Model We devised a logistic regression (LR) model incorporating 18 simple features, computed directly from $\mathbf{s}_1$ and $\mathbf{s}_2$, without modeling any hidden correspondence. LR (like the QG) provides a probability distribution, but uses surface features (like the SVM). The features are of the form $precision_n$ (number of n-gram matches divided by the number of n-grams in $\mathbf{s}_1$), $recall_n$ (number of n-gram matches divided by the number of n-grams in $\mathbf{s}_2$) and F_n (harmonic mean of the previous two features), where $1 \leq n \leq 3$. We also used lemmatized versions of these features. This model gives the posterior probability $p_L(c \mid \mathbf{s}_1, \mathbf{s}_2)$, where $c \in \{\mathsf{p}, \mathsf{n}\}$. We estimated the model parameters analogously to Eq. 17. Performance is reported in Tab. 2; this model is on par with the SVM, though trading recall in favor of precision. We view it as a probabilistic simulation of the SVM more suitable for combination with the QG.

Training the PoE Various ways of training a PoE exist. We first trained p_Q and p_L separately as described, then initialized the PoE with those parameters. We then continued training, maximizing (unregularized) conditional likelihood.

Experiment We used p_Q with the "c-command" configuration excluded, and the LR model in the product of experts. Tab. 2 includes the final results achieved by the PoE. The PoE model outperforms all the other models, achieving an accuracy of 76.06%.[11] The PoE is conservative, labeling a pair as p only if the LR and the QG give it strong p probabilities. This leads to high precision, at the expense of recall.

Oracle Ensembles Tab. 2 shows the results of three different oracle ensemble systems that correctly classify a pair if *either* of the two individual systems in the combination is correct. Note that the combinations involving p_Q achieve 83%, the human agreement level for the MSRPC. The LR and SVM are highly similar, and their oracle combination does not perform as well.

[11] This accuracy is significant over p_Q under a paired t-test ($p < 0.04$), but is not significant over the SVM.

7 Related Work

There is a growing body of research that uses the MSRPC (Dolan et al., 2004; Quirk et al., 2004) to build models of paraphrase. As noted, the most successful work has used edit distance (Zhang and Patrick, 2005) or bag-of-words features to measure sentence similarity, along with shallow syntactic features (Finch et al., 2005; Wan et al., 2006; Corley and Mihalcea, 2005). Qiu et al. (2006) used predicate-argument annotations.

Most related to our approach, Wu (2005) used inversion transduction grammars—a synchronous context-free formalism (Wu, 1997)—for this task. Wu reported only positive-class (p) precision (not accuracy) on the test set. He obtained 76.1%, while our PoE model achieves 79.6% on that measure. Wu's model can be understood as a strict hierarchical maximum-alignment method. In contrast, our alignments are soft (we sum over them), and we do not require strictly isomorphic syntactic structures. Most importantly, our approach is founded on a stochastic generating process and estimated discriminatively for this task, while Wu did not estimate any parameters from data at all.

8 Conclusion

In this paper, we have presented a probabilistic model of paraphrase incorporating syntax, lexical semantics, and hidden loose alignments between two sentences' trees. Though it fully defines a generative process for both sentences and their relationship, the model is discriminatively trained to maximize conditional likelihood. We have shown that this model is competitive for determining whether there exists a semantic relationship between them, and can be improved by principled combination with more standard lexical overlap approaches.

Acknowledgments

The authors thank the three anonymous reviewers for helpful comments and Alan Black, Frederick Crabbe, Jason Eisner, Kevin Gimpel, Rebecca Hwa, David Smith, and Mengqiu Wang for helpful discussions. This work was supported by DARPA grant NBCH-1080004.

References

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proc. of NAACL*.

Daniel M. Bikel, Richard L. Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proc. of HLT-NAACL*.

Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proc. of ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proc. of IWP*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *Proc. of COLING*.

Andrew Finch, Young Sook Hwang, and Eiichiro Sumita. 2005. Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In *Proc. of IWP*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proc. of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

David Graff. 2003. English Gigaword. Linguistic Data Consortium.

Geoffrey E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Math. Programming (Ser. B)*, 45(3):503–528.

Erwin Marsi and Emiel Krahmer. 2005. Explorations in sentence fusion. In *Proc. of EWNLG*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of ACL*.

Kathleen R. McKeown. 1979. Paraphrasing using given and new information in a question-answer system. In *Proc. of ACL*.

I. Dan Melamed. 2004. Statistical machine translation by parsing. In *Proc. of ACL*.

George A. Miller. 1995. Wordnet: a lexical database for English. *Commun. ACM*, 38(11):39–41.

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

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2006. Paraphrase recognition via dissimilarity significance classification. In *Proc. of EMNLP*.

Chris Quirk, Chris Brockett, and William B. Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proc. of EMNLP*.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proc. of EMNLP*.

David A. Smith and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proc. of the HLT-NAACL Workshop on Statistical Machine Translation*.

Joseph P. Turian, Luke Shen, and I. Dan Melamed. 2003. Evaluation of machine translation and its evaluation. In *Proc. of Machine Translation Summit IX*.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2006. Using dependency-based features to take the "para-farce" out of paraphrase. In *Proc. of ALTW*.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proc. of EMNLP-CoNLL*.

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

Dekai Wu. 2005. Recognizing paraphrases and textual entailment using inversion transduction grammars. In *Proc. of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of IWPT*.

Yitao Zhang and Jon Patrick. 2005. Paraphrase identification by text canonicalization. In *Proc. of ALTW*.

Stochastic Gradient Descent Training for L1-regularized Log-linear Models with Cumulative Penalty

Yoshimasa Tsuruoka[†‡] **Jun'ichi Tsujii**[†‡*] **Sophia Ananiadou**[†‡]
[†] School of Computer Science, University of Manchester, UK
[‡] National Centre for Text Mining (NaCTeM), UK
[*] Department of Computer Science, University of Tokyo, Japan
{yoshimasa.tsuruoka,j.tsujii,sophia.ananiadou}@manchester.ac.uk

Abstract

Stochastic gradient descent (SGD) uses approximate gradients estimated from subsets of the training data and updates the parameters in an online fashion. This learning framework is attractive because it often requires much less training time in practice than batch training algorithms. However, L1-regularization, which is becoming popular in natural language processing because of its ability to produce compact models, cannot be efficiently applied in SGD training, due to the large dimensions of feature vectors and the fluctuations of approximate gradients. We present a simple method to solve these problems by penalizing the weights according to cumulative values for L1 penalty. We evaluate the effectiveness of our method in three applications: text chunking, named entity recognition, and part-of-speech tagging. Experimental results demonstrate that our method can produce compact and accurate models much more quickly than a state-of-the-art quasi-Newton method for L1-regularized log-linear models.

1 Introduction

Log-linear models (a.k.a maximum entropy models) are one of the most widely-used probabilistic models in the field of natural language processing (NLP). The applications range from simple classification tasks such as text classification and history-based tagging (Ratnaparkhi, 1996) to more complex structured prediction tasks such as part-of-speech (POS) tagging (Lafferty et al., 2001), syntactic parsing (Clark and Curran, 2004) and semantic role labeling (Toutanova et al., 2005). Log-linear models have a major advantage over other discriminative machine learning models such as support vector machines—their probabilistic output allows the information on the confidence of the decision to be used by other components in the text processing pipeline.

The training of log-liner models is typically performed based on the maximum likelihood criterion, which aims to obtain the weights of the features that maximize the conditional likelihood of the training data. In maximum likelihood training, *regularization* is normally needed to prevent the model from overfitting the training data,

The two most common regularization methods are called L1 and L2 regularization. L1 regularization penalizes the weight vector for its L1-norm (i.e. the sum of the absolute values of the weights), whereas L2 regularization uses its L2-norm. There is usually not a considerable difference between the two methods in terms of the accuracy of the resulting model (Gao et al., 2007), but L1 regularization has a significant advantage in practice. Because many of the weights of the features become zero as a result of L1-regularized training, the size of the model can be much smaller than that produced by L2-regularization. Compact models require less space on memory and storage, and enable the application to start up quickly. These merits can be of vital importance when the application is deployed in resource-tight environments such as cell-phones.

A common way to train a large-scale L1-regularized model is to use a quasi-Newton method. Kazama and Tsujii (2003) describe a method for training a L1-regularized log-linear model with a bound constrained version of the BFGS algorithm (Nocedal, 1980). Andrew and Gao (2007) present an algorithm called Orthant-Wise Limited-memory Quasi-Newton (OWL-QN), which can work on the BFGS algorithm without bound constraints and achieve faster convergence.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 477–485,
Suntec, Singapore, 2-7 August 2009.

An alternative approach to training a log-linear model is to use stochastic gradient descent (SGD) methods. SGD uses approximate gradients estimated from subsets of the training data and updates the weights of the features in an online fashion—the weights are updated much more frequently than batch training algorithms. This learning framework is attracting attention because it often requires much less training time in practice than batch training algorithms, especially when the training data is large and redundant. SGD was recently used for NLP tasks including machine translation (Tillmann and Zhang, 2006) and syntactic parsing (Smith and Eisner, 2008; Finkel et al., 2008). Also, SGD is very easy to implement because it does not need to use the Hessian information on the objective function. The implementation could be as simple as the perceptron algorithm.

Although SGD is a very attractive learning framework, the direct application of L1 regularization in this learning framework does not result in efficient training. The first problem is the inefficiency of applying the L1 penalty to the weights of all features. In NLP applications, the dimension of the feature space tends to be very large—it can easily become several millions, so the application of L1 penalty to all features significantly slows down the weight updating process. The second problem is that the naive application of L1 penalty in SGD does not always lead to compact models, because the approximate gradient used at each update is very noisy, so the weights of the features can be easily moved away from zero by those fluctuations.

In this paper, we present a simple method for solving these two problems in SGD learning. The main idea is to keep track of the total penalty and the penalty that has been applied to each weight, so that the L1 penalty is applied based on the difference between those cumulative values. That way, the application of L1 penalty is needed only for the features that are used in the current sample, and also the effect of noisy gradient is smoothed away.

We evaluate the effectiveness of our method by using linear-chain conditional random fields (CRFs) and three traditional NLP tasks, namely, text chunking (shallow parsing), named entity recognition, and POS tagging. We show that our enhanced SGD learning method can produce compact and accurate models much more quickly than the OWL-QN algorithm.

This paper is organized as follows. Section 2 provides a general description of log-linear models used in NLP. Section 3 describes our stochastic gradient descent method for L1-regularized log-linear models. Experimental results are presented in Section 4. Some related work is discussed in Section 5. Section 6 gives some concluding remarks.

2 Log-Linear Models

In this section, we briefly describe log-linear models used in NLP tasks and L1 regularization.

A log-linear model defines the following probabilistic distribution over possible structure $\mathbf{y}$ for input $\mathbf{x}$:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp \sum_i w_i f_i(\mathbf{y}, \mathbf{x}),$$

where $f_i(\mathbf{y}, \mathbf{x})$ is a function indicating the occurrence of feature i, w_i is the weight of the feature, and $Z(\mathbf{x})$ is a partition (normalization) function:

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \exp \sum_i w_i f_i(\mathbf{y}, \mathbf{x}).$$

If the structure is a sequence, the model is called a linear-chain CRF model, and the marginal probabilities of the features and the partition function can be efficiently computed by using the forward-backward algorithm. The model is used for a variety of sequence labeling tasks such as POS tagging, chunking, and named entity recognition.

If the structure is a tree, the model is called a tree CRF model, and the marginal probabilities can be computed by using the inside-outside algorithm. The model can be used for tasks like syntactic parsing (Finkel et al., 2008) and semantic role labeling (Cohn and Blunsom, 2005).

2.1 Training

The weights of the features in a log-linear model are optimized in such a way that they maximize the regularized conditional log-likelihood of the training data:

$$\mathcal{L}_{\mathbf{w}} = \sum_{j=1}^{N} \log p(\mathbf{y}_j|\mathbf{x}_j; \mathbf{w}) - R(\mathbf{w}), \qquad (1)$$

where N is the number of training samples, $\mathbf{y}_j$ is the correct output for input $\mathbf{x}_j$, and $R(\mathbf{w})$ is the

regularization term which prevents the model from overfitting the training data. In the case of L1 regularization, the term is defined as:

$$R(\mathbf{w}) = C \sum_i |w_i|,$$

where C is the meta-parameter that controls the degree of regularization, which is usually tuned by cross-validation or using the heldout data.

In what follows, we denote by $L(j, \mathbf{w})$ the conditional log-likelihood of each sample $\log p(\mathbf{y}_j | \mathbf{x}_j; \mathbf{w})$. Equation 1 is rewritten as:

$$\mathcal{L}_{\mathbf{w}} = \sum_{j=1}^{N} L(j, \mathbf{w}) - C \sum_i |w_i|. \qquad (2)$$

3 Stochastic Gradient Descent

SGD uses a small randomly-selected subset of the training samples to approximate the gradient of the objective function given by Equation 2. The number of training samples used for this approximation is called the *batch size*. When the batch size is N, the SGD training simply translates into gradient descent (hence is very slow to converge). By using a small batch size, one can update the parameters more frequently than gradient descent and speed up the convergence. The extreme case is a batch size of 1, and it gives the maximum frequency of updates and leads to a very simple perceptron-like algorithm, which we adopt in this work.[1]

Apart from using a single training sample to approximate the gradient, the optimization procedure is the same as simple gradient descent,[2] so the weights of the features are updated at training sample j as follows:

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \eta_k \frac{\partial}{\partial \mathbf{w}} (L(j, \mathbf{w}) - \frac{C}{N} \sum_i |w_i|),$$

where k is the iteration counter and η_k is the learning rate, which is normally designed to decrease as the iteration proceeds. The actual learning rate scheduling methods used in our experiments are described later in Section 3.3.

[1] In the actual implementation, we randomly shuffled the training samples at the beginning of each pass, and then picked them up sequentially.

[2] What we actually do here is gradient ascent, but we stick to the term "gradient descent".

3.1 L1 regularization

The update equation for the weight of each feature i is as follows:

$${w_i}^{k+1} = {w_i}^k + \eta_k \frac{\partial}{\partial w_i} (L(j, \mathbf{w}) - \frac{C}{N} |w_i|).$$

The difficulty with L1 regularization is that the last term on the right-hand side of the above equation is not differentiable when the weight is zero. One straightforward solution to this problem is to consider a subgradient at zero and use the following update equation:

$${w_i}^{k+1} = {w_i}^k + \eta_k \frac{\partial L(j, \mathbf{w})}{\partial w_i} - \frac{C}{N} \eta_k \text{sign}(w_i^k),$$

where $\text{sign}(x) = 1$ if $x > 0$, $\text{sign}(x) = -1$ if $x < 0$, and $\text{sign}(x) = 0$ if $x = 0$. In this paper, we call this weight updating method "SGD-L1 (Naive)".

This naive method has two serious problems. The first problem is that, at each update, we need to perform the application of L1 penalty to all features, including the features that are not used in the current training sample. Since the dimension of the feature space can be very large, it can significantly slow down the weight update process.

The second problem is that it does not produce a compact model, i.e. most of the weights of the features do not become zero as a result of training. Note that the weight of a feature does not become zero unless it happens to fall on zero exactly, which rarely happens in practice.

Carpenter (2008) describes an alternative approach. The weight updating process is divided into two steps. First, the weight is updated without considering the L1 penalty term. Then, the L1 penalty is applied to the weight to the extent that it does not change its sign. In other words, the weight is clipped when it crosses zero. Their weight update procedure is as follows:

$$w_i^{k+\frac{1}{2}} = w_i^k + \eta_k \left. \frac{\partial L(j, \mathbf{w})}{\partial w_i} \right|_{\mathbf{w}=\mathbf{w}^k},$$

if $w_i^{k+\frac{1}{2}} > 0$ **then**

$$w_i^{k+1} = \max(0, w_i^{k+\frac{1}{2}} - \frac{C}{N} \eta_k),$$

else if $w_i^{k+\frac{1}{2}} < 0$ **then**

$$w_i^{k+1} = \min(0, w_i^{k+\frac{1}{2}} + \frac{C}{N} \eta_k).$$

In this paper, we call this update method "SGD-L1 (Clipping)". It should be noted that this method

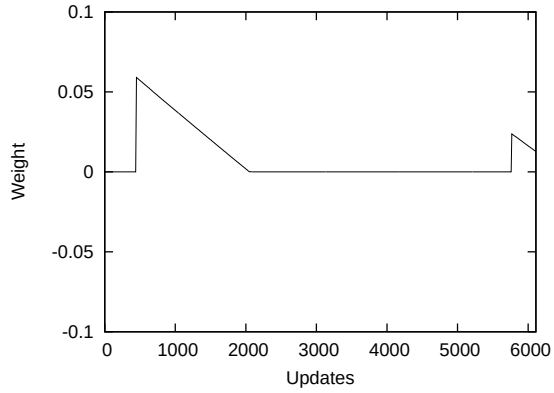


Figure 1: An example of weight updates.

is actually a special case of the FOLOS algorithm (Duchi and Singer, 2008) and the truncated gradient method (Langford et al., 2009).

The obvious advantage of using this method is that we can expect many of the weights of the features to become zero during training. Another merit is that it allows us to perform the application of L1 penalty in a lazy fashion, so that we do not need to update the weights of the features that are not used in the current sample, which leads to much faster training when the dimension of the feature space is large. See the aforementioned papers for the details. In this paper, we call this efficient implementation "SGD-L1 (Clipping + Lazy-Update)".

3.2 L1 regularization with cumulative penalty

Unfortunately, the clipping-at-zero approach does not solve all problems. Still, we often end up with many features whose weights are not zero. Recall that the gradient used in SGD is a crude approximation to the true gradient and is very noisy. The weight of a feature is, therefore, easily moved away from zero when the feature is used in the current sample.

Figure 1 gives an illustrative example in which the weight of a feature fails to become zero. The figure shows how the weight of a feature changes during training. The weight goes up sharply when it is used in the sample and then is pulled back toward zero gradually by the L1 penalty. Therefore, the weight fails to become zero if the feature is used toward the end of training, which is the case in this example. Note that the weight would become zero if the true (fluctuationless) gradient were used—at each update the weight would go up a little and be pulled back to zero straightaway.

Here, we present a different strategy for applying the L1 penalty to the weights of the features. The key idea is to smooth out the effect of fluctuating gradients by considering the cumulative effects from L1 penalty.

Let u_k be the absolute value of the total L1-penalty that each weight could have received up to the point. Since the absolute value of the L1 penalty does not depend on the weight and we are using the same regularization constant C for all weights, it is simply accumulated as:

$$u_k = \frac{C}{N} \sum_{t=1}^{k} \eta_t. \tag{3}$$

At each training sample, we update the weights of the features that are used in the sample as follows:

$$w_i^{k+\frac{1}{2}} = w_i^k + \eta_k \left. \frac{\partial L(j, \mathbf{w})}{\partial w_i} \right|_{\mathbf{w}=\mathbf{w}^k},$$

if $w_i^{k+\frac{1}{2}} > 0$ **then**

$$w_i^{k+1} = \max(0, w_i^{k+\frac{1}{2}} - (u_k + q_i^{k-1})),$$

else if $w_i^{k+\frac{1}{2}} < 0$ **then**

$$w_i^{k+1} = \min(0, w_i^{k+\frac{1}{2}} + (u_k - q_i^{k-1})),$$

where q_i^k is the total L1-penalty that w_i has actually received up to the point:

$$q_i^k = \sum_{t=1}^{k} (w_i^{t+1} - w_i^{t+\frac{1}{2}}). \tag{4}$$

This weight updating method penalizes the weight according to the difference between u_k and q_i^{k-1}. In effect, it forces the weight to receive the total L1 penalty that would have been applied if the weight had been updated by the true gradients, assuming that the current weight vector resides in the same orthant as the true weight vector.

It should be noted that this method is basically equivalent to a "SGD-L1 (Clipping + Lazy-Update)" method if we were able to use the true gradients instead of the stochastic gradients.

In this paper, we call this weight updating method "SGD-L1 (Cumulative)". The implementation of this method is very simple. Figure 2 shows the whole SGD training algorithm with this strategy in pseudo-code.

```
1:  procedure TRAIN(C)
2:      u ← 0
3:      Initialize w_i and q_i with zero for all i
4:      for k = 0 to MaxIterations
5:          η ← LEARNINGRATE(k)
6:          u ← u + ηC/N
7:          Select sample j randomly
8:          UPDATEWEIGHTS(j)
9:
10: procedure UPDATEWEIGHTS(j)
11:     for i ∈ features used in sample j
12:         w_i ← w_i + η ∂L(j,w)/∂w_i
13:         APPLYPENALTY(i)
14:
15: procedure APPLYPENALTY(i)
16:     z ← w_i
17:     if w_i > 0 then
18:         w_i ← max(0, w_i − (u + q_i))
19:     else if w_i < 0 then
20:         w_i ← min(0, w_i + (u − q_i))
21:     q_i ← q_i + (w_i − z)
22:
```

Figure 2: Stochastic gradient descent training with cumulative L1 penalty. z is a temporary variable.

3.3 Learning Rate

The scheduling of learning rates often has a major impact on the convergence speed in SGD training.

A typical choice of learning rate scheduling can be found in (Collins et al., 2008):

$$\eta_k = \frac{\eta_0}{1 + k/N}, \tag{5}$$

where η_0 is a constant. Although this scheduling guarantees ultimate convergence, the actual speed of convergence can be poor in practice (Darken and Moody, 1990).

In this work, we also tested simple exponential decay:

$$\eta_k = \eta_0 \alpha^{-k/N}, \tag{6}$$

where α is a constant. In our experiments, we found this scheduling more practical than that given in Equation 5. This is mainly because exponential decay sweeps the range of learning rates more smoothly—the learning rate given in Equation 5 drops too fast at the beginning and too slowly at the end.

It should be noted that exponential decay is not a good choice from a theoretical point of view, because it does not satisfy one of the necessary conditions for convergence—the sum of the learning rates must diverge to infinity (Spall, 2005). However, this is probably not a big issue for practitioners because normally the training has to be terminated at a certain number of iterations in practice.[3]

4 Experiments

We evaluate the effectiveness our training algorithm using linear-chain CRF models and three NLP tasks: text chunking, named entity recognition, and POS tagging.

To compare our algorithm with the state-of-the-art, we present the performance of the OWL-QN algorithm on the same data. We used the publicly available OWL-QN optimizer developed by Andrew and Gao.[4] The meta-parameters for learning were left unchanged from the default settings of the software: the convergence tolerance was 1e-4; and the L-BFGS memory parameter was 10.

4.1 Text Chunking

The first set of experiments used the text chunking data set provided for the CoNLL 2000 shared task.[5] The training data consists of 8,936 sentences in which each token is annotated with the "IOB" tags representing text chunks such as noun and verb phrases. We separated 1,000 sentences from the training data and used them as the heldout data. The test data provided by the shared task was used only for the final accuracy report.

The features used in this experiment were unigrams and bigrams of neighboring words, and unigrams, bigrams and trigrams of neighboring POS tags.

To avoid giving any advantage to our SGD algorithms over the OWL-QN algorithm in terms of the accuracy of the resulting model, the OWL-QN algorithm was used when tuning the regularization parameter C. The tuning was performed in such a way that it maximized the likelihood of the heldout data. The learning rate parameters for SGD were then tuned in such a way that they maximized the value of the objective function in 30 passes. We first determined η_0 by testing 1.0, 0.5, 0.2, and 0.1. We then determined α by testing 0.9, 0.85, and 0.8 with the fixed η_0.

[3] This issue could also be sidestepped by, for example, adding a small $O(1/k)$ term to the learning rate.

[4] Available from the original developers' websites: http://research.microsoft.com/en-us/people/galena/ or http://research.microsoft.com/en-us/um/people/jfgao/

[5] http://www.cnts.ua.ac.be/conll2000/chunking/

	Passes	$\mathcal{L}_{\mathbf{w}}/N$	# Features	Time (sec)	F-score
OWL-QN	160	-1.583	18,109	598	93.62
SGD-L1 (Naive)	30	-1.671	455,651	1,117	93.64
SGD-L1 (Clipping + Lazy-Update)	30	-1.671	87,792	144	93.65
SGD-L1 (Cumulative)	30	-1.653	28,189	149	93.68
SGD-L1 (Cumulative + Exponential-Decay)	30	-1.622	23,584	148	93.66

Table 1: CoNLL-2000 Chunking task. Training time and accuracy of the trained model on the test data.

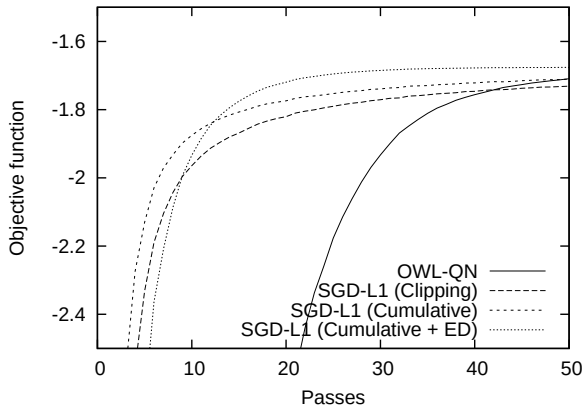


Figure 3: CoNLL 2000 chunking task: Objective

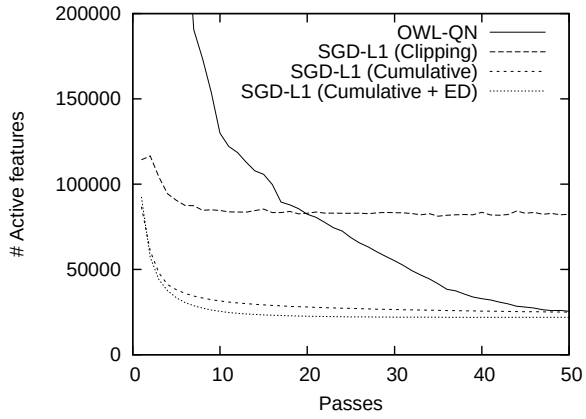


Figure 4: CoNLL 2000 chunking task: Number of active features.

Figures 3 and 4 show the training process of the model. Each figure contains four curves representing the results of the OWL-QN algorithm and three SGD-based algorithms. "SGD-L1 (Cumulative + ED)" represents the results of our cumulative penalty-based method that uses exponential decay (ED) for learning rate scheduling.

Figure 3 shows how the value of the objective function changed as the training proceeded. SGD-based algorithms show much faster convergence than the OWL-QN algorithm. Notice also that "SGD-L1 (Cumulative)" improves the objective slightly faster than "SGD-L1 (Clipping)". The result of "SGD-L1 (Naive)" is not shown in this figure, but the curve was almost identical to that of "SGD-L1 (Clipping)".

Figure 4 shows the numbers of active features (the features whose weight are not zero). It is clearly seen that the clipping-at-zero approach fails to reduce the number of active features, while our algorithms succeeded in reducing the number of active features to the same level as OWL-QN.

We then trained the models using the whole training data (including the heldout data) and evaluated the accuracy of the chunker on the test data. The number of passes performed over the training data in SGD was set to 30. The results are shown in Table 1. The second column shows the number of passes performed in the training. The third column shows the final value of the objective function per sample. The fourth column shows the number of resulting active features. The fifth column show the training time. The last column shows the f-score (harmonic mean of recall and precision) of the chunking results. There was no significant difference between the models in terms of accuracy. The naive SGD training took much longer than OWL-QN because of the overhead of applying L1 penalty to all dimensions.

Our SGD algorithms finished training in 150 seconds on Xeon 2.13GHz processors. The CRF++ version 0.50, a popular CRF library developed by Taku Kudo,[6] is reported to take 4,021 seconds on Xeon 3.0GHz processors to train the model using a richer feature set.[7] CRFsuite version 0.4, a much faster library for CRFs, is reported to take 382 seconds on Xeon 3.0GHz, using the same feature set as ours.[8] Their library uses the OWL-QN algorithm for optimization. Although direct comparison of training times is not impor-

[6] http://crfpp.sourceforge.net/

[7] http://www.chokkan.org/software/crfsuite/benchmark.html

[8] ditto

tant due to the differences in implementation and hardware platforms, these results demonstrate that our algorithm can actually result in a very fast implementation of a CRF trainer.

4.2 Named Entity Recognition

The second set of experiments used the named entity recognition data set provided for the BioNLP/NLPBA 2004 shared task (Kim et al., 2004).[9] The training data consist of 18,546 sentences in which each token is annotated with the "IOB" tags representing biomedical named entities such as the names of proteins and RNAs.

The training and test data were preprocessed by the GENIA tagger,[10] which provided POS tags and chunk tags. We did not use any information on the named entity tags output by the GENIA tagger. For the features, we used unigrams of neighboring chunk tags, substrings (shorter than 10 characters) of the current word, and the shape of the word (e.g. "IL-2" is converted into "AA-#"), on top of the features used in the text chunking experiments.

The results are shown in Figure 5 and Table 2. The trend in the results is the same as that of the text chunking task: our SGD algorithms show much faster convergence than the OWL-QN algorithm and produce compact models.

Okanohara et al. (2006) report an f-score of 71.48 on the same data, using semi-Markov CRFs.

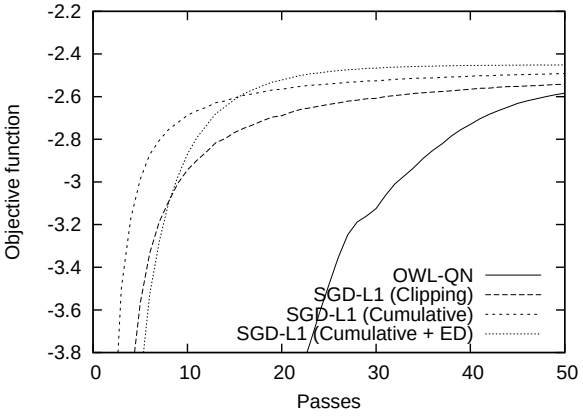


Figure 5: NLPBA 2004 named entity recognition task: Objective.

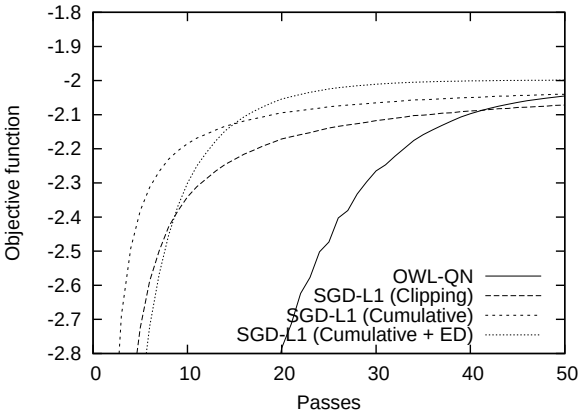


Figure 6: POS tagging task: Objective.

4.3 Part-Of-Speech Tagging

The third set of experiments used the POS tagging data in the Penn Treebank (Marcus et al., 1994). Following (Collins, 2002), we used sections 0-18 of the Wall Street Journal (WSJ) corpus for training, sections 19-21 for development, and sections 22-24 for final evaluation. The POS tags were extracted from the parse trees in the corpus. All experiments for this work, including the tuning of features and parameters for regularization, were carried out using the training and development sets. The test set was used only for the final accuracy report.

It should be noted that training a CRF-based POS tagger using the whole WSJ corpus is not a trivial task and was once even deemed impractical in previous studies. For example, Wellner and Vilain (2006) abandoned maximum likelihood training because it was "prohibitive" (7-8 days for sections 0-18 of the WSJ corpus).

For the features, we used unigrams and bigrams of neighboring words, prefixes and suffixes of the current word, and some characteristics of the word. We also normalized the current word by lowering capital letters and converting all the numerals into '#', and used the normalized word as a feature.

The results are shown in Figure 6 and Table 3. Again, the trend is the same. Our algorithms finished training in about 30 minutes, producing accurate models that are as compact as that produced by OWL-QN.

Shen et al., (2007) report an accuracy of 97.33% on the same data set using a perceptron-based bidirectional tagging model.

5 Discussion

An alternative approach to producing compact models for log-linear models is to reformulate the

[9] The data is available for download at http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/ERtask/report.html

[10] http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/tagger/

	Passes	$\mathcal{L}_{\mathbf{w}}/N$	# Features	Time (sec)	F-score
OWL-QN	161	-2.448	30,710	2,253	71.76
SGD-L1 (Naive)	30	-2.537	1,032,962	4,528	71.20
SGD-L1 (Clipping + Lazy-Update)	30	-2.538	279,886	585	71.20
SGD-L1 (Cumulative)	30	-2.479	31,986	631	71.40
SGD-L1 (Cumulative + Exponential-Decay)	30	-2.443	25,965	631	71.63

Table 2: NLPBA 2004 Named entity recognition task. Training time and accuracy of the trained model on the test data.

	Passes	$\mathcal{L}_{\mathbf{w}}/N$	# Features	Time (sec)	Accuracy
OWL-QN	124	-1.941	50,870	5,623	97.16%
SGD-L1 (Naive)	30	-2.013	2,142,130	18,471	97.18%
SGD-L1 (Clipping + Lazy-Update)	30	-2.013	323,199	1,680	97.18%
SGD-L1 (Cumulative)	30	-1.987	62,043	1,777	97.19%
SGD-L1 (Cumulative + Exponential-Decay)	30	-1.954	51,857	1,774	97.17%

Table 3: POS tagging on the WSJ corpus. Training time and accuracy of the trained model on the test data.

problem as a L1-constrained problem (Lee et al., 2006), where the conditional log-likelihood of the training data is maximized under a fixed constraint of the L1-norm of the weight vector. Duchi et al. (2008) describe efficient algorithms for projecting a weight vector onto the L1-ball. Although L1-regularized and L1-constrained learning algorithms are not directly comparable because the objective functions are different, it would be interesting to compare the two approaches in terms of practicality. It should be noted, however, that the efficient algorithm presented in (Duchi et al., 2008) needs to employ a red-black tree and is rather complex.

In SGD learning, the need for tuning the meta-parameters for learning rate scheduling can be annoying. In the case of exponential decay, the setting of $\alpha = 0.85$ turned out to be a good rule of thumb in our experiments—it always produced near best results in 30 passes, but the other parameter η_0 needed to be tuned. It would be very useful if those meta-parameters could be tuned in a fully automatic way.

There are some sophisticated algorithms for adaptive learning rate scheduling in SGD learning (Vishwanathan et al., 2006; Huang et al., 2007). However, those algorithms use second-order information (i.e. Hessian information) and thus need access to the weights of the features that are not used in the current sample, which should slow down the weight updating process for the same reason discussed earlier. It would be interesting to investigate whether those sophisticated learning scheduling algorithms can actually result in fast training in large-scale NLP tasks.

6 Conclusion

We have presented a new variant of SGD that can efficiently train L1-regularized log-linear models. The algorithm is simple and extremely easy to implement.

We have conducted experiments using CRFs and three NLP tasks, and demonstrated empirically that our training algorithm can produce compact and accurate models much more quickly than a state-of-the-art quasi-Newton method for L1-regularization.

Acknowledgments

We thank N. Okazaki, N. Yoshinaga, D. Okanohara and the anonymous reviewers for their useful comments and suggestions. The work described in this paper has been funded by the Biotechnology and Biological Sciences Research Council (BBSRC; BB/E004431/1). The research team is hosted by the JISC/BBSRC/EPSRC sponsored National Centre for Text Mining.

References

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of ICML*, pages 33–40.

Bob Carpenter. 2008. Lazy sparse stochastic gradient descent for regularized multinomial logistic regression. Technical report, Alias-i.

Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proceedings of COLING 2004*, pages 103–110.

Trevor Cohn and Philip Blunsom. 2005. Semantic role labeling with tree conditional random fields. In *Proceedings of CoNLL*, pages 169–172.

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L. Bartlett. 2008. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *The Journal of Machine Learning Research (JMLR)*, 9:1775–1822.

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

Christian Darken and John Moody. 1990. Note on learning rate schedules for stochastic optimization. In *Proceedings of NIPS*, pages 832–838.

Juhn Duchi and Yoram Singer. 2008. Online and batch learning using forward-looking subgradients. In *NIPS Workshop: OPT 2008 Optimization for Machine Learning*.

Juhn Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. 2008. Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of ICML*, pages 272–279.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL-08:HLT*, pages 959–967.

Jianfeng Gao, Galen Andrew, Mark Johnson, and Kristina Toutanova. 2007. A comparative study of parameter estimation methods for statistical natural language processing. In *Proceedings of ACL*, pages 824–831.

Han-Shen Huang, Yu-Ming Chang, and Chun-Nan Hsu. 2007. Training conditional random fields by periodic step size adaptation for large-scale text mining. In *Proceedings of ICDM*, pages 511–516.

Jun'ichi Kazama and Jun'ichi Tsujii. 2003. Evaluation and extension of maximum entropy models with inequality constraints. In *Proceedings of EMNLP 2003*.

J.-D. Kim, T. Ohta, Y. Tsuruoka, Y. Tateisi, and N. Collier. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA)*, pages 70–75.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

John Langford, Lihong Li, and Tong Zhang. 2009. Sparse online learning via truncated gradient. *The Journal of Machine Learning Research (JMLR)*, 10:777–801.

Su-In Lee, Honglak Lee, Pieter Abbeel, and Andrew Y. Ng. 2006. Efficient l1 regularized logistic regression. In *Proceedings of AAAI-06*, pages 401–408.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Jorge Nocedal. 1980. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782.

Daisuke Okanohara, Yusuke Miyao, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2006. Improving the scalability of semi-markov conditional random fields for named entity recognition. In *Proceedings of COLING/ACL*, pages 465–472.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP 1996*, pages 133–142.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of ACL*, pages 760–767.

David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of EMNLP*, pages 145–156.

James C. Spall. 2005. *Introduction to Stochastic Search and Optimization*. Wiley-IEEE.

Christoph Tillmann and Tong Zhang. 2006. A discriminative global training algorithm for statistical MT. In *Proceedings of COLING/ACL*, pages 721–728.

Kristina Toutanova, Aria Haghighi, and Christopher Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL*, pages 589–596.

S. V. N. Vishwanathan, Nicol N. Schraudolph, Mark W. Schmidt, and Kevin P. Murphy. 2006. Accelerated training of conditional random fields with stochastic gradient methods. In *Proceedings of ICML*, pages 969–976.

Ben Wellner and Marc Vilain. 2006. Leveraging machine readable dictionaries in discriminative sequence models. In *Proceedings of LREC 2006*.

A global model for joint lemmatization and part-of-speech prediction

Kristina Toutanova
Microsoft Research
Redmond, WA 98052
kristout@microsoft.com

Colin Cherry
Microsoft Research
Redmond, WA 98052
colinc@microsoft.com

Abstract

We present a global joint model for lemmatization and part-of-speech prediction. Using only morphological lexicons and unlabeled data, we learn a partially-supervised part-of-speech tagger and a lemmatizer which are combined using features on a dynamically linked dependency structure of words. We evaluate our model on English, Bulgarian, Czech, and Slovene, and demonstrate substantial improvements over both a direct transduction approach to lemmatization and a pipelined approach, which predicts part-of-speech tags before lemmatization.

1 Introduction

The traditional problem of morphological analysis is, given a word form, to predict the set of all of its possible morphological analyses. A morphological analysis consists of a part-of-speech tag (POS), possibly other morphological features, and a lemma (basic form) corresponding to this tag and features combination (see Table 1 for examples). We address this problem in the setting where we are given a morphological dictionary for training, and can additionally make use of un-annotated text in the language. We present a new machine learning model for this task setting.

In addition to the morphological analysis task we are interested in performance on two subtasks: *tag-set prediction* (predicting the set of possible tags of words) and *lemmatization* (predicting the set of possible lemmas). The result of these subtasks is directly useful for some applications.[1] If we are interested in the results of each of these two subtasks in isolation, we might build independent solutions which ignore the other subtask.

In this paper, we show that there are strong dependencies between the two subtasks and we can improve performance on both by sharing information between them. We present a joint model for these two subtasks: it is joint not only in that it performs both tasks simultaneously, sharing information, but also in that it reasons about *multiple words* jointly. It uses component tag-set and lemmatization models and combines their predictions while incorporating joint features in a log-linear model, defined on a dynamically linked dependency structure of words.

The model is formalized in Section 5 and evaluated in Section 6. We report results on English, Bulgarian, Slovene, and Czech and show that joint modeling reduces the lemmatization error by up to 19%, the tag-prediction error by up to 26% and the error on the complete morphological analysis task by up to 22.6%.

2 Task formalization

The main task that we would like to solve is as follows: given a lexicon L which contains all morphological analyses for a set of words $\{w_1, \ldots, w_n\}$, learn to predict all morphological analyses for other words which are outside of L. In addition to the lexicon, we are allowed to make use of unannotated text T in the language. We will predict morphological analyses for words which occur in T. Note that the task is defined on word types and not on words in context.

A morphological analysis of a word w consists of a (possibly structured) POS tag t, together with one or several lemmas, which are the possible basic forms of w when it has tag t. As an example, Table 1 illustrates the morphological analyses of several words taken from the CELEX lexical database of English (Baayen et al., 1995) and the Multext-East lexicon of Bulgarian (Erjavec, 2004). The Bulgarian words are transcribed in

[1] Tag sets are useful, for example, as a basis of sparsity-reducing features for text labeling tasks; lemmatization is useful for information retrieval and machine translation from a morphologically rich to a morphologically poor language, where full analysis may not be important.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 486–494,
Suntec, Singapore, 2-7 August 2009.

Word Forms	Morphological Analyses	Tags	Lemmas
tell	verb base (VB), *tell*	VB	*tell*
told	verb past tense (VBD), *tell* verb past participle (VBN), *tell*	VBD,VBN	*tell*
tells	verb present 3rd person sing (VBZ), *tell*	VBZ	*tell*
telling	verb present continuous (VBG), *tell* adjective (JJ), *telling*	VBG,JJ	*tell* *telling*
izpravena	adjective fem sing indef (A–FS-N), *izpraven* verb main part past sing fem pass indef (VMPS-SFP-N), *izpravia*	A–FS-N VMPS-SFP-N	*izpraven* *izpravia*
izpraviha	verb main indicative 3rd person plural (VMIA3P), *izpravia*	VMIA3P	*izpravia*

Table 1: Examples of morphological analyses of words in English and Bulgarian.

Latin characters. Here by "POS tags" we mean both simple main pos-tags such as *noun* or *verb*, and detailed tags which include grammatical features, such as VBZ for English indicating present tense third person singular verb and A–FS-N for Bulgarian indicating a feminine singular adjective in indefinite form. In this work we predict only main POS tags for the Multext-East languages, as detailed tags were less useful for lemmatization.

Since the predicted elements are sets, we use precision, recall, and F-measure (F1) to evaluate performance. The two subtasks, tag-set prediction and lemmatization are also evaluated in this way. Table 1 shows the correct tag-sets and lemmas for each of the example words in separate columns. Our task setting differs from most work on lemmatization which uses either no or a complete rootlist (Wicentowski, 2002; Dreyer et al., 2008).[2] We can use all forms occurring in the unlabeled text T but there are no guarantees about the coverage of the target lemmas or the number of noise words which may occur in T (see Table 2 for data statistics). Our setting is thus more realistic since it is what one would have in a real application scenario.

3 Related work

In work on morphological analysis using machine learning, the task is rarely addressed in the form described above. Some exceptions are the work (Bosch and Daelemans, 1999) which presents a model for segmenting, stemming, and tagging words in Dutch, and requires the prediction of all possible analyses, and (Antal van den Bosch and Soudi, 2007) which similarly requires the prediction of all morpho-syntactically annotated segmentations of words for Arabic. As opposed to our work, these approaches do not make use of unlabeled data and make predictions for each word type in isolation.

In machine learning work on lemmatization for highly inflective languages, it is most often assumed that a word form and a POS tag are given, and the task is to predict the set of corresponding lemma(s) (Mooney and Califf, 1995; Clark, 2002; Wicentowski, 2002; Erjavec and Džeroski, 2004; Dreyer et al., 2008). In our task setting, we do not assume the availability of gold-standard POS tags. As a component model, we use a lemmatizing string transducer which is related to these approaches and draws on previous work in this and related string transduction areas. Our transducer is described in detail in Section 4.1.

Another related line of work approaches the disambiguation problem directly, where the task is to predict the correct analysis of word-forms in context (in sentences), and not all possible analyses. In such work it is often assumed that the correct POS tags can be predicted with high accuracy using *labeled* POS-disambiguated sentences (Erjavec and Džeroski, 2004; Habash and Rambow, 2005). A notable exception is the work of (Adler et al., 2008), which uses unlabeled data and a morphological analyzer to learn a semi-supervised HMM model for disambiguation in context, and also guesses analyses for unknown words using a guesser of likely POS-tags. It is most closely related to our work, but does not attempt to predict all possible analyses, and does not have to tackle a complex string transduction problem for lemmatization since segmentation is mostly sufficient for the focus language of that study (Hebrew).

The idea of solving two related tasks jointly to improve performance on both has been successful for other pairs of tasks (e.g., (Andrew et al., 2004)). Doing joint inference instead of taking a pipeline approach has also been shown useful for other problems (e.g., (Finkel et al., 2006; Cohen and Smith, 2007)).

[2]These settings refer to the availability of a set of word forms which are possible lemmas; in the no rootlist setting, no other word forms in the language are given in addition to the forms in the training set; in the complete rootlist setting, a set of word forms which consists of exactly all correct lemmas for the words in the test set is given.

4 Component models

We use two component models as the basis of addressing the task: one is a partially-supervised POS tagger which is trained using L and the unlabeled text T; the other is a lemmatizing transducer which is trained from L and can use T. The transducer can optionally be given input POS tags in training and testing, which can inform the lemmatization. The tagger is described in Section 4.2 and the transducer is described in Section 4.1.

In a pipeline approach to combining the tagging and lemmatization components, we first predict a set of tags for each word using the tagger, and then ask the lemmatizer to predict one lemma for each of the possible tags. In a direct transduction approach to the lemmatization subtask, we train the lemmatizer without access to tags and ask it to predict a single lemma for each word in testing. Our joint model, described in Section 5, is defined in a re-ranking framework, and can choose from among k-best predictions of tag-sets and lemmas generated from the component tagger and lemmatizer models.

4.1 Morphological analyser

We employ a discriminative character transducer as a component morphological analyzer. The input to the transducer is an inflected word (the source) and possibly an estimated part-of-speech; the output is the lemma of the word (the target). The transducer is similar to the one described by Jiampojamarn et al. (2008) for letter-to-phoneme conversion, but extended to allow for whole-word features on both the input and the output. The core of our engine is the dynamic programming algorithm for monotone phrasal decoding (Zens and Ney, 2004). The main feature of this algorithm is its capability to transduce many consecutive characters with a single operation; the same algorithm is employed to tag subsequences in semi-Markov CRFs (Sarawagi and Cohen, 2004).

We employ three main categories of features: context, transition, and vocabulary (rootlist) features. The first two are described in detail by Jiampojamarn et al. (2008), while the final is novel to this work. Context features are centered around a transduction operation such as $es \rightarrow e$, as employed in $gives \rightarrow give$. Context features include an indicator for the operation itself, conjoined with indicators for all n-grams of source context within a fixed window of the operation. We also employ a copy feature that indicates if the operation simply copies the source character, such as $e \rightarrow e$. Transition features are our Markov, or n-gram features on transduction operations. Vocabulary features are defined on complete target words, according to the frequency of said word in a provided unlabeled text T. We have chosen to bin frequencies; experiments on a development set suggested that two indicators are sufficient: the first fires for any word that occurred fewer than five times, while a second also fires for those words that did not occur at all. By encoding our vocabulary in a trie and adding the trie index to the target context tracked by our dynamic programming chart, we can efficiently track these frequencies during transduction.

We incorporate the source part-of-speech tag by appending it to each feature, thus the context feature $es \rightarrow e$ may become $es \rightarrow e, \text{VBZ}$. To enable communication between the various parts-of-speech, a universal set of unannotated features also fires, regardless of the part-of-speech, acting as a back-off model of how words in general behave during stemming.

Linear weights are assigned to each of the transducer's features using an averaged perceptron for structure prediction (Collins, 2002). Note that our features are defined in terms of the operations employed during transduction, therefore to create gold-standard feature vectors, we require not only target outputs, but also derivations to produce those outputs. We employ a deterministic heuristic to create these derivations; given a gold-standard source-target pair, we construct a derivation that uses only trivial copy operations until the first character mismatch. The remainder of the transduction is performed with a single multi-character replacement. For example, the derivation for $living \rightarrow live$ would be $l \rightarrow l$, $i \rightarrow i$, $v \rightarrow v$, $ing \rightarrow e$. For languages with morphologies affecting more than just the suffix, one can either develop a more complex heuristic, or determine the derivations using a separate aligner such as that of Ristad and Yianilos (1998).

4.2 Tag-set prediction model

The tag-set model uses a training lexicon L and unlabeled text T to learn to predict sets of tags for words. It is based on the semi-supervised tagging model of (Toutanova and Johnson, 2008). It has two sub-models: one is an ambiguity class

or a tag-set model, which can assign probabilities for possible sets of tags of words $P_{TSM}(ts|w)$ and the other is a word context model, which can assign probabilities $P_{CM}(contexts_w|w, ts)$ to all contexts of occurrence of word w in an unlabeled text T. The word-context model is Bayesian and utilizes a sparse Dirichlet prior on the distributions of tags given words. In addition, it uses information on a four word context of occurrences of w in the unlabeled text.

Note that the (Toutanova and Johnson, 2008) model is a tagger that assigns tags to occurrences of words in the text, whereas we only need to predict sets of possible tags for word types, such as the set $\{\text{VBD}, \text{VBN}\}$ for the word *told*. Their component sub-model P_{TSM} predicts sets of tags and it is possible to use it on its own, but by also using the context model we can take into account information from the context of occurrence of words and compute probabilities of tag-sets given the observed occurrences in T. The two are combined to make a prediction for a tag-set of a test word w, given unlabeled text T, using Bayes rule: $p(ts|w) \propto P_{TSM}(ts|w)P_{CM}(contexts_w|w, ts)$.

We use a direct re-implementation of the word-context model, using variational inference following (Toutanova and Johnson, 2008). For the tag-set sub-model, we employ a more sophisticated approach. First, we learn a log-linear classifier instead of a Naive Bayes model, and second, we use features derived from related words appearing in T. The possible classes predicted by the classifier are as many as the observed tag-sets in L. The sparsity is relieved by adding features for individual tags t which get shared across tag-sets containing t.

There are two types of features in the model: (*i*) word-internal features: word suffixes, capitalization, existence of hyphen, and word prefixes (such features were also used in (Toutanova and Johnson, 2008)), and (*ii*) features based on related words. These latter features are inspired by (Cucerzan and Yarowsky, 2000) and are defined as follows: for a word w such as *telling*, there is an indicator feature for every combination of two suffixes α and β, such that there is a prefix p where *telling*= $p\alpha$ and $p\beta$ exists in T. For example, if the word $tells$ is found in T, there would be a feature for the suffixes α=ing,β=s that fires. The suffixes are defined as all character suffixes up to length three which occur with at least 100 words.

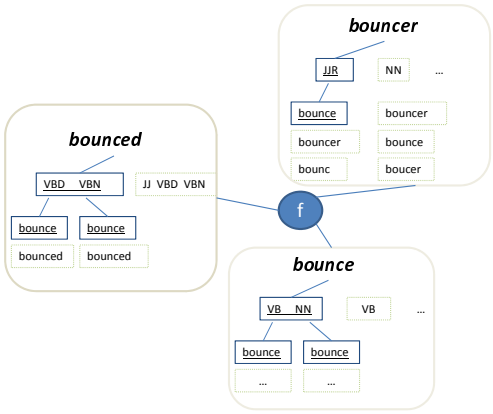


Figure 1: A small subset of the graphical model. The tag-sets and lemmas active in the illustrated assignment are shown in bold. The extent of joint features firing for the lemma *bounce* is shown as a factor indicated by the blue circle and connected to the assignments of the three words.

5 A global joint model for morphological analysis

The idea of this model is to jointly predict the set of possible tags and lemmas of words. In addition to modeling dependencies between the tags and lemmas of a single word, we incorporate dependencies between the predictions for *multiple words*. The dependencies among words are determined dynamically. Intuitively, if two words have the same lemma, their tag-sets are dependent. For example, imagine that we need to determine the tag-set and lemmas of the word *bouncer*. The tag-set model may guess that the word is an adjective in comparative form, because of its suffix, and because its occurrences in T might not strongly indicate that it is a noun. The lemmatizer can then lemmatize the word like an adjective and come up with *bounce* as a lemma. If the tag-set model is fairly certain that *bounce* is not an adjective, but is a verb or a noun, a joint model which looks simultaneously at the tags and lemmas of *bouncer* and *bounce* will detect a problem with this assignments and will be able to correct the tagging and lemmatization error for *bouncer*.

The main source of information our joint model uses is information about the assignments of all words that have the same lemma l. If the tag-set model is better able to predict the tags of some of these words, the information can propagate to the other words. If some of them are lemmatized correctly, the model can be pushed to lemmatize the others correctly as well. Since the lemmas of test words are not given, the dependencies between as-

signments of words are determined dynamically by the currently chosen set of lemmas.

As an example, Figure 1 shows three sample English words and their possible tag-sets and lemmas determined by the component models. It also illustrates the dependencies between the variables induced by the features of our model active for the current (incorrect) assignment.

5.1 Formal model description

Given a set of test words $w_1, \ldots w_n$ and additional word forms occurring in unlabeled data T, we derive an extended set of words $w_1, \ldots, w_m$ which contains the original test words and additional related words, which can provide useful information about the test words. For example, if *bouncer* is a test word and *bounce* and *bounced* occur in T these two words can be added to the set of test words because they can contribute to the classification of *bouncer*. The algorithm for selecting related words is simple: we add any word for which the pipelined model predicts a lemma which is also predicted as one of the top k lemmas for a word from the test set.

We define a joint model over tag-sets and lemmas for all words in the extended set, using features defined on a dynamically linked structure of words and their assigned analyses. It is a reranking model because the tag-sets and possible lemmas are limited to the top k options provided by the pipelined model.[3] Our model is defined on a very large set of variables, each of which can take a large set of values. For example, for a test set of size about 4,000 words for Slovene an additional about 9,000 words from T were added to the extended set. Each of these words has a corresponding variable which indicates its tag-set and lemma assignment. The possible assignments range over all combinations available from the tagging and lemmatizer component models; using the top three tag-sets per word and top three lemmas per tag gives an average of around 11.2 possible assignments per word. This is because the tag-sets have about 1.2 tags on average and we need to choose a lemma for each. While it is not the case that all variables are connected to each other by features, the connectivity structure can be complex.

More formally, let ts_i^j denote possible tag-sets for word w_i, for $j = 1 \ldots k$. Also, let $l_i(t)^j$ denote the top lemmas for word w_i given tag t. An assignment of a tag-set and lemmas to a word w_i consists of a choice of a tag-set, ts_i (one of the possible k tag-sets for the word) and, for each tag t in the chosen tag-set, a choice of a lemma out of the possible lemmas for that tag and word. For brevity, we denote such joint assignment by tl_i. As a concrete example, in Figure 1, we can see the current assignments for three words: the assigned tag-sets are shown underlined and in bolded boxes (e.g., for *bounced*, the tag-set {VBD,VBN} is chosen; for both tags, the lemma *bounce* is assigned). Other possible tag-sets and other possible lemmas for each chosen tag are shown in greyed boxes.

Our joint model defines a distribution over assignments to all words $w_1, \ldots, w_m$. The form of the model is as follows:

$$P(tl_1, \ldots, tl_m) = \frac{e^{F(tl_1,\ldots,tl_m)'\theta}}{\sum_{tl'_1,\ldots,tl'_m} e^{F(tl'_1,\ldots,tl'_m)'\theta}}$$

Here F denotes the vector of features defined over an assignment for all words in the set and θ is a vector of parameters for the features. Next we detail the types of features used.

Word-local features. The aim of such features is to look at the set of all tags assigned to a word together with all lemmas and capture coarse-grained dependencies at this level. These features introduce joint dependencies between the tags and lemmas of a word, but they are still local to the assignment of single words. One such feature is the number of distinct lemmas assigned across the different tags in the assigned tag-set. Another such feature is the above joined with the identity of the tag-set. For example, if a word's tag-set is {VBD,VBN}, it will likely have the same lemma for both tags and the number of distinct lemmas will be one (e.g., the word *bounced*), whereas if it has the tags VBG, JJ the lemmas will be distinct for the two tags (e.g. *telling*). In this class of features are also the log-probabilities from the tag-set and lemmatizer models.

Non-local features. Our non-local features look, for every lemma l, at all words which have that lemma as the lemma for at least one of their assigned tags, and derive several predicates on the joint assignment to these words. For example, using our word graph in the figure, the lemma *bounce* is assigned to *bounced* for tags VBD and VBN, to *bounce* for tags VB and NN, and to *bouncer* for tag JJR. One feature looks at the combination of tags corresponding to the differ-

[3] We used top three tag-sets and top three lemmas for each tag for training.

ent forms of the lemma. In this case this would be [JJR,NN+VB-*lem*,VBD+VBN]. The feature also indicates any word which is exactly equal to the lemma with *lem* as shown for the NN and VB tags corresponding to *bounce*. Our model learns a negative weight for this feature, because the lemma of a word with tag JJR is most often a word with at least one tag equal to JJ. A variant of this feature also appends the final character of each word, like this: [JJR+r,NN+VB+e-*lem*,VBD+VBN-d]. This variant was helpful for the Slavic languages because when using only main POS tags, the granularity of the feature is too coarse. Another feature simply counts the number of distinct words having the same lemma, encouraging reusing the same lemma for different words. An additional feature fires for every distinct lemma, in effect counting the number of assigned lemmas.

5.2 Training and inference

Since the model is defined to re-rank candidates from other component models, we need two different training sets: one for training the component models, and another for training the joint model features. This is because otherwise the accuracy of the component models would be overestimated by the joint model. Therefore, we train the component models on the training lexicons LTrain and select their hyperparameters on the LDev lexicons. We then train the joint model on the LDev lexicons and evaluate it on the LTest lexicons. When applying models to the LTest set, the component models are first retrained on the union of LTrain and LDev so that all models can use the same amount of training data, without giving unfair advantage to the joint model. Such set-up is also used for other re-ranking models (Collins, 2000).

For training the joint model, we maximize the log-likelihood of the correct assignment to the words in LDev, marginalizing over the assignments of other related words added to the graphical model. We compute the gradient approximately by computing expectations of features given the observed assignments and marginal expectations of features. For computing these expectations we use Gibbs sampling to sample complete assignments to all words in the graph.[4] We use gradient descent with a small learning rate, selected to optimize the accuracy on the LDev set. For finding a most likely assignment at test time, we use the sampling procedure, this time using a slower annealing schedule before taking a single sample to output as a guessed answer.

For the Gibbs sampler, we need to sample an assignment for each word in turn, given the current assignments of all other words. Let us denote the current assignment to all words except w_i as $\mathbf{tl}^{-i}$. The conditional probability of an assignment tl_i for word w_i is given by:

$$P(tl_i|\mathbf{tl}^{-i}) = \frac{e^{F(tl_i,\mathbf{tl}^{-i})'\theta}}{\sum_{tl'_i} e^{F(tl'_i,\mathbf{tl}^{-i})'\theta}}$$

The summation in the denominator is over all possible assignments for word w_i. To compute these quantities we need to consider only the features involving the current word. Because of the nature of the features in our model, it is possible to isolate separate connected components which do not share features for any assignment. If two words do not share lemmas for any of their possible assignments, they will be in separate components. Block sampling within a component could be used if the component is relatively small; however, for the common case where there are five or more words in a fully connected component approximate inference is necessary.

6 Experiments

6.1 Data

We use datasets for four languages: English, Bulgarian, Slovene, and Czech. For each of the languages, we need a lexicon with morphological analyses L and unlabeled text.

For English we derive the lexicon from CELEX (Baayen et al., 1995), and for the other languages we use the Multext-East resources (Erjavec, 2004). For English we use only open-class words (nouns, verbs, adjectives, and adverbs), and for the other languages we use words of all classes. The unlabeled data for English we use is the union of the Penn Treebank tagged WSJ data (Marcus et al., 1993) and the BLLIP corpus.[5] For the rest of the languages we use only the text of George Orwell's novel *1984*, which is provided in morphologically disambiguated form as part of Multext-East (but we don't use the annotations). Table 2

[4] We start the Gibbs sampler by the assignments found by the pipeline method and then use an annealing schedule to find a neighborhood of high-likelihood assignments, before taking about 10 complete samples from the graph to compute expectations.

[5] The BLLIP corpus contains approximately 30 million words of automatically parsed WSJ data. We used these corpora as plain text, without the annotations.

Lang	LTrain			LDev			LTest			Text
	ws	tl	nf	ws	tl	nf	ws	tl	nf	
Eng	5.2	1.5	0.3	7.4	1.4	0.8	7.4	1.4	0.8	320
Bgr	6.9	1.2	40.8	3.8	1.1	53.6	3.8	1.1	52.8	16.3
Slv	7.5	1.2	38.3	4.2	1.2	49.1	4.2	1.2	49.8	17.8
Cz	7.9	1.1	32.8	4.5	1.1	43.2	4.5	1.1	43.0	19.1

Table 2: Data sets used in experiments. The number of word types (ws) is shown approximately in thousands. Also shown are average number of complete analyses (tl) and percent target lemmas not found in the unlabeled text (nf).

details statistics about the data set sizes for different languages.

We use three different lexicons for each language: one for training (LTrain), one for development (LDev), and one for testing (LTest). The global model weights are trained on the development set as described in section 5.2. The lexicons are derived such that very frequent words are likely to be in the training lexicon and less frequent words in the dev and test lexicons, to simulate a natural process of lexicon construction. The English lexicons were constructed as follows: starting with the full CELEX dictionary and the text of the Penn Treebank corpus, take all word forms appearing in the first 2000 sentences (and are found in CELEX) to form the training lexicon, and then take all other words occurring in the corpus and split them equally between the development and test lexicons (every second word is placed in the test set, in the order of first occurrence in the corpus). For the rest of the languages, the same procedure is applied, starting with the full Multext-East lexicons and the text of the novel *1984*. Note that while it is not possible for training words to be included in the other lexicons, it is possible for different forms of the same lemma to be in different lexicons. The size of the training lexicons is relatively small and we believe this is a realistic scenario for application of such models. In Table 2 we can see the number of words in each lexicon and the unlabeled corpora (by type), the average number of tag-lemma combinations per word,[6] as well as the percentage of word lemmas which do not occur in the unlabeled text. For English, the large majority of target lemmas are available in T (with only 0.8% missing), whereas for the Multext-East languages around 40 to 50% of the target lemmas are not found in T; this partly explains the lower performance on these languages.

[6]The tags are main tags for the Multext-East languages and detailed tags for English.

Language	Tag Model	Tag	Lem	T+L
English	none	–	94.0	–
	full	89.9	95.3	88.9
	no unlab data	80.0	94.1	78.3
Bulgarian	none	–	73.2	–
	full	87.9	79.9	75.3
	no unlab data	80.2	76.3	70.4

Table 3: Development set results using different tag-set models and pipelined prediction.

6.2 Evaluation of direct and pipelined models for lemmatization

As a first experiment which motivates our joint modeling approach, we present a comparison on lemmatization performance in two settings: (*i*) when no tags are used in training or testing by the transducer, and (*ii*) when correct tags are used in training and tags predicted by the tagging model are used in testing. In this section, we report performance on English and Bulgarian only. Comparable performance on the other Multext-East languages is shown in Section 6.

Results are presented in Table 3. The experiments are performed using LTrain for training and LDev for testing. We evaluate the models on tag-set F-measure (Tag), lemma-set F-measure(Lem) and complete analysis F-measure (T+L). We show the performance on lemmatization when tags are not predicted (Tag Model is none), and when tags are predicted by the tag-set model. We can see that on both languages lemmatization is significantly improved when a latent tag-set variable is used as a basis for prediction: the relative error reduction in Lem F-measure is 21.7% for English and 25% for Bulgarian. For Bulgarian and the other Slavic languages we predicted only main POS tags, because this resulted in better lemmatization performance.

It is also interesting to evaluate the contribution of the unlabeled data T to the performance of the tag-set model. This can be achieved by removing the word-context sub-model of the tagger and also removing related word features. The results achieved in this setting for English and Bulgarian are shown in the rows labeled "no unlab data". We can see that the tag-set F-measure of such models is reduced by 8 to 9 points and the lemmatization F-measure is similarly reduced. Thus a large portion of the positive impact tagging has on lemmatization is due to the ability of tagging models to exploit unlabeled data.

The results of this experiment show there are strong dependencies between the tagging and

lemmatization subtasks, which a joint model could exploit.

6.3 Evaluation of joint models

Since our joint model re-ranks candidates produced by the component tagger and lemmatizer, there is an upper bound on the achievable performance. We report these upper bounds for the four languages in Table 4, at the rows which list m-best oracle under **Model**. The oracle is computed using five-best tag-set predictions and three-best lemma predictions per tag. We can see that the oracle performance on tag F-measure is quite high for all languages, but the performance on lemmatization and the complete task is close to only 90 percent for the Slavic languages. As a second oracle we also report the perfect tag oracle, which selects the lemmas determined by the transducer using the correct part-of-speech tags. This shows how well we could do if we made the tagging model perfect without changing the lemmatizer. For the Slavic languages this is quite a bit lower than the m-best oracles, showing that the majority of errors of the pipelined approach cannot be fixed by simply improving the tagging model. Our global model has the potential to improve lemma assignments even given correct tags, by sharing information among multiple words.

The actual achieved performance for three different models is also shown. For comparison, the lemmatization performance of the direct transduction approach which makes no use of tags is also shown. The pipelined models select one-best tag-set predictions from the tagging model, and the 1-best lemmas for each tag, like the models used in Section 6.2. The model name *local FS* denotes a joint log-linear model which has only word-internal features. Even with only word-internal features, performance is improved for most languages. The the highest improvement is for Slovene and represents a 7.8% relative reduction in F-measure error on the complete task.

When features looking at the joint assignments of multiple words are added, the model achieves much larger improvements (models *joint FS* in the Table) across all languages.[7] The highest overall improvement compared to the pipelined approach is again for Slovene and represents 22.6% reduction in error for the full task; the reduction is 40% relative to the upper bound achieved by the m-best oracle. The smallest overall improvement is for English, representing a 10% error reduction overall, which is still respectable. The larger improvement for Slavic languages might be due to the fact that there are many more forms of a single lemma and joint reasoning allows us to pool information across the forms.

[7] Since the optimization is stochastic, the results are averaged over four runs. The standard deviations are between 0.02 and 0.11.

Language	Model	Tag	Lem	T+L
English	tag oracle	100	98.9	98.7
English	m-best oracle	97.9	99.0	97.5
English	no tags	–	94.3	–
English	pipelined	90.9	95.9	90.0
English	local FS	90.8	95.9	90.0
English	joint FS	91.7	96.1	91.0
Bulgarian	tag oracle	100	84.3	84.3
Bulgarian	m-best oracle	98.4	90.7	89.9
Bulgarian	no tags	–	73.2	–
Bulgarian	pipelined	87.9	78.5	74.6
Bulgarian	local FS	88.9	79.2	75.8
Bulgarian	joint FS	89.5	81.0	77.8
Slovene	tag oracle	100	85.9	85.9
Slovene	m-best oracle	98.7	91.2	90.5
Slovene	no tags	–	78.4	–
Slovene	pipelined	89.7	82.1	78.3
Slovene	local FS	90.8	82.7	80.0
Slovene	joint FS	92.4	85.5	83.2
Czech	tag oracle	100	83.2	83.2
Czech	m-best oracle	98.1	88.7	87.4
Czech	no tags	–	78.7	–
Czech	pipelined	92.3	80.7	77.5
Czech	local FS	92.3	80.9	78.0
Czech	joint FS	93.7	83.0	80.5

Table 4: Results on the test set achieved by joint and pipelined models and oracles. The numbers represent tag-set prediction F-measure (**Tag**), lemma-set prediction F-measure (**Lem**) and F-measure on predicting complete tag, lemma analysis sets (**T+L**).

7 Conclusion

In this paper we concentrated on the task of morphological analysis, given a lexicon and unannotated data. We showed that the tasks of tag prediction and lemmatization are strongly dependent and that by building state-of-the art models for the two subtasks and performing joint inference we can improve performance on both tasks. The main contribution of our work was that we introduced a joint model for the two subtasks which incorporates dependencies between predictions for *multiple word types*. We described a set of features and an approximate inference procedure for a global log-linear model capturing such dependencies, and demonstrated its effectiveness on English and three Slavic languages.

Acknowledgements

We would like to thank Galen Andrew and Lucy Vanderwende for useful discussion relating to this work.

References

Meni Adler, Yoav Goldberg, and Michael Elhadad. 2008. Unsupervised lexicon-based resolution of unknown words for full morpholological analysis. In *Proceedings of ACL-08: HLT*.

Galen Andrew, Trond Grenager, and Christopher Manning. 2004. Verb sense and subcategorization: Using joint inference to improve performance on complementary tasks. In *EMNLP*.

Erwin Marsi Antal van den Bosch and Abdelhadi Soudi. 2007. Memory-based morphological analysis and part-of-speech tagging of arabic. In Abdelhadi Soudi, Antal van den Bosch, and Gunter Neumann, editors, *Arabic Computational Morphology Knowledge-based and Empirical Methods*. Springer.

R. H. Baayen, R. Piepenbrock, and L. Gulikers. 1995. The CELEX lexical database.

Antal Van Den Bosch and Walter Daelemans. 1999. Memory-based morphological analysis. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.

Alexander Clark. 2002. Memory-based learning of morphology with stochastic transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 513–520.

Shay B. Cohen and Noah A. Smith. 2007. Joint morphological and syntactic disambiguation. In *EMNLP*.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *ICML*.

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

S. Cucerzan and D. Yarowsky. 2000. Language independent minimally supervised induction of lexical probabilities. In *Proceedings of ACL 2000*.

Markus Dreyer, Jason R. Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1080–1089, Honolulu, October.

Tomaž Erjavec and Saăo Džeroski. 2004. Machine learning of morphosyntactic structure: lemmatizing unknown Slovene words. *Applied Artificial Intelligence*, 18:17—41.

Tomaž Erjavec. 2004. Multext-east version 3: Multilingual morphosyntactic specifications, lexicons and corpora. In *Proceedings of LREC-04*.

Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *EMNLP*.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. In *Proceedings of ACL-08: HLT*, pages 905–913, Columbus, Ohio, June.

M. Marcus, B. Santorini, and Marcinkiewicz. 1993. Building a large annotated coprus of english: the penn treebank. *Computational Linguistics*, 19.

Raymond J. Mooney and Mary Elaine Califf. 1995. Induction of first-order decision lists: Results on learning the past tense of english verbs. *Journal of Artificial Intelligence Research*, 3:1—24.

Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532.

Sunita Sarawagi and William Cohen. 2004. Semimarkov conditional random fields for information extraction. In *ICML*.

Kristina Toutanova and Mark Johnson. 2008. A bayesian LDA-based model for semi-supervised part-of-speech tagging. In *nips08*.

Richard Wicentowski. 2002. *Modeling and Learning Multilingual Inflectional Morphology in a Minimally Supervised Framework*. Ph.D. thesis, Johns-Hopkins University.

R. Zens and H. Ney. 2004. Improvements in phrase-based statistical machine translation. In *HLT-NAACL*, pages 257–264, Boston, USA, May.

Distributional Representations for Handling Sparsity in Supervised Sequence-Labeling

Fei Huang
Temple University
1805 N. Broad St.
Wachman Hall 324
tub58431@temple.edu

Alexander Yates
Temple University
1805 N. Broad St.
Wachman Hall 324
yates@temple.edu

Abstract

Supervised sequence-labeling systems in natural language processing often suffer from data sparsity because they use word types as features in their prediction tasks. Consequently, they have difficulty estimating parameters for types which appear in the test set, but seldom (or never) appear in the training set. We demonstrate that distributional representations of word types, trained on unannotated text, can be used to improve performance on rare words. We incorporate aspects of these representations into the feature space of our sequence-labeling systems. In an experiment on a standard chunking dataset, our best technique improves a chunker from 0.76 F1 to 0.86 F1 on chunks beginning with rare words. On the same dataset, it improves our part-of-speech tagger from 74% to 80% accuracy on rare words. Furthermore, our system improves significantly over a baseline system when applied to text from a different domain, and it reduces the sample complexity of sequence labeling.

1 Introduction

Data sparsity and high dimensionality are the twin curses of statistical natural language processing (NLP). In many traditional supervised NLP systems, the feature space includes dimensions for each word type in the data, or perhaps even combinations of word types. Since vocabularies can be extremely large, this leads to an explosion in the number of parameters. To make matters worse, language is Zipf-distributed, so that a large fraction of any training data set will be *hapax legomena*, very many word types will appear only a few times, and many word types will be left out of the training set altogether. As a consequence, for many word types supervised NLP systems have very few, or even zero, labeled examples from which to estimate parameters.

The negative effects of data sparsity have been well-documented in the NLP literature. The performance of state-of-the-art, supervised NLP systems like part-of-speech (POS) taggers degrades significantly on words that do not appear in the training data, or out-of-vocabulary (OOV) words (Lafferty et al., 2001). Performance also degrades when the domain of the test set differs from the domain of the training set, in part because the test set includes more OOV words and words that appear only a few times in the training set (henceforth, *rare* words) (Blitzer et al., 2006; Daumé III and Marcu, 2006; Chelba and Acero, 2004).

We investigate the use of distributional representations, which model the probability distribution of a word's context, as techniques for finding *smoothed* representations of word sequences. That is, we use the distributional representations to share information across unannotated examples of the same word type. We then compute features of the distributional representations, and provide them as input to our supervised sequence labelers. Our technique is particularly well-suited to handling data sparsity because it is possible to improve performance on rare words by supplementing the training data with additional unannotated text containing more examples of the rare words. We provide empirical evidence that shows how distributional representations improve sequence-labeling in the face of data sparsity.

Specifically, we investigate empirically the effects of our smoothing techniques on two sequence-labeling tasks, POS tagging and chunking, to answer the following:

1. What is the effect of smoothing on sequence-labeling accuracy for rare word types? Our best smoothing technique improves a POS tagger by 11% on OOV words, and a chunker by an impressive 21% on OOV words.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 495–503,
Suntec, Singapore, 2-7 August 2009.

2. Can smoothing improve adaptability to new domains? After training our chunker on newswire text, we apply it to biomedical texts. Remarkably, we find that the smoothed chunker achieves a higher F1 on the new domain than the baseline chunker achieves on a test set from the original newswire domain.
3. How does our smoothing technique affect sample complexity? We show that smoothing drastically reduces sample complexity: our smoothed chunker requires under 100 labeled samples to reach 85% accuracy, whereas the unsmoothed chunker requires 3500 samples to reach the same level of performance.

The remainder of this paper is organized as follows. Section 2 discusses the smoothing problem for word sequences, and introduces three smoothing techniques. Section 3 presents our empirical study of the effects of smoothing on two sequence-labeling tasks. Section 4 describes related work, and Section 5 concludes and suggests items for future work.

2 Smoothing Natural Language Sequences

To *smooth* a dataset is to find an approximation of it that retains the important patterns of the original data while hiding the noise or other complicating factors. Formally, we define the smoothing task as follows: let $\mathcal{D} = \{(\mathbf{x}, \mathbf{z}) | \mathbf{x}$ is a word sequence, $\mathbf{z}$ is a label sequence$\}$ be a labeled dataset of word sequences, and let $\mathcal{M}$ be a machine learning algorithm that will learn a function f to predict the correct labels. The smoothing task is to find a function g such that when $\mathcal{M}$ is applied to $D' = \{(g(\mathbf{x}), \mathbf{z}) | (\mathbf{x}, \mathbf{z}) \in D\}$, it produces a function f' that is more accurate than f.

For supervised sequence-labeling problems in NLP, the most important "complicating factor" that we seek to avoid through smoothing is the data sparsity associated with word-based representations. Thus, the task is to find g such that for every word x, $g(x)$ is much less sparse, but still retains the essential features of x that are useful for predicting its label.

As an example, consider the string "Researchers test reformulated gasolines on newer engines." In a common dataset for NP chunking, the word "reformulated" never appears in the training data, but appears four times in the test set as part of the NP "reformulated gasolines." Thus, a learning algorithm supplied with word-level features would have a difficult time determining that "reformulated" is the start of a NP. Character-level features are of little help as well, since the "-ed" suffix is more commonly associated with verb phrases. Finally, context may be of some help, but "test" is ambiguous between a noun and verb, and "gasolines" is only seen once in the training data, so there is no guarantee that context is sufficient to make a correct judgment.

On the other hand, some of the other contexts in which "reformulated" appears in the test set, such as "testing of reformulated gasolines," provide strong evidence that it can start a NP, since "of" is a highly reliable indicator that a NP is to follow. This example provides the intuition for our approach to smoothing: we seek to share information about the contexts of a word across multiple instances of the word, in order to provide more information about words that are rarely or never seen in training. In particular, we seek to represent each word by a distribution over its contexts, and then provide the learning algorithm with features computed from this distribution. Importantly, we seek distributional representations that will provide features that are common in both training and test data, to avoid data sparsity. In the next three sections, we develop three techniques for smoothing text using distributional representations.

2.1 Multinomial Representation

In its simplest form, the context of a word may be represented as a multinomial distribution over the terms that appear on either side of the word. If $\mathcal{V}$ is the vocabulary, or the set of word types, and $\mathbf{X}$ is a sequence of random variables over $\mathcal{V}$, the left and right context of $X_i = v$ may each be represented as a probability distribution over $\mathcal{V}$: $P(X_{i-1}|X_i = v)$ and $P(X_{i+1}|X = v)$ respectively.

We learn these distributions from unlabeled texts in two different ways. The first method computes word count vectors for the left and right contexts of each word type in the vocabulary of the training and test texts. We also use a large collection of additional text to determine the vectors. We then normalize each vector to form a probability distribution. The second technique first applies TF-IDF weighting to each vector, where the context words of each word type constitute a document, before applying normalization. This gives greater weight to words with more idiosyncratic distributions and may improve the informativeness of a distributional representation. We refer to these techniques as TF and TF-IDF.

To supply a sequence-labeling algorithm with information from these distributional representations, we compute real-valued features of the context distributions. In particular, for every word x_i in a sequence, we provide the sequence labeler with a set of features of the left and right contexts indexed by $v \in \mathcal{V}$: $F_v^{left}(x_i) = P(X_{i-1} = v|x_i)$ and $F_v^{right}(x_i) = P(X_{i+1} = v|x_i)$. For example, the left context for "reformulated" in our example above would contain a nonzero probability for the word "of." Using the features $\mathbf{F}(x_i)$, a sequence labeler can learn patterns such as, if x_i has a high probability of following "of," it is a good candidate for the start of a noun phrase. These features provide smoothing by aggregating information across multiple unannotated examples of the same word.

2.2 LSA Model

One drawback of the multinomial representation is that it does not handle sparsity well enough, because the multinomial distributions themselves are so high-dimensional. For example, the two phrases "red lamp" and "magenta tablecloth" share no words in common. If "magenta" is never observed in training, the fact that "tablecloth" appears in its right context is of no help in connecting it with the phrase "red lamp." But if we can group similar context words together, putting "lamp" and "tablecloth" into a category for household items, say, then these two adjectives will share that category in their context distributions. Any patterns learned for the more common "red lamp" will then also apply to the less common "magenta tablecloth." Our second distributional representation aggregates information from multiple context words by grouping together the distributions $P(x_{i-1} = v|x_i = w)$ and $P(x_{i-1} = v'|x_i = w)$ if v and v' appear together with many of the same words w. Aggregating counts in this way smooths our representations even further, by supplying better estimates when the data is too sparse to estimate $P(x_{i-1}|x_i)$ accurately.

Latent Semantic Analysis (LSA) (Deerwester et al., 1990) is a widely-used technique for computing dimensionality-reduced representations from a bag-of-words model. We apply LSA to the set of right context vectors and the set of left context vectors separately, to find compact versions of each vector, where each dimension represents a combination of several context word types. We normalize each vector, and then calculate features as above. After experimenting with different choices for the number of dimensions to reduce our vectors to, we choose a value of 10 dimensions as the one that maximizes the performance of our supervised sequence labelers on held-out data.

2.3 Latent Variable Language Model Representation

To take smoothing one step further, we present a technique that aggregates context distributions both for similar context words $x_{i-1} = v$ and v', and for similar words $x_i = w$ and w'. Latent variable language models (LVLMs) can be used to produce just such a distributional representation. We use Hidden Markov Models (HMMs) as the main example in the discussion and as the LVLMs in our experiments, but the smoothing technique can be generalized to other forms of LVLMs, such as factorial HMMs and latent variable maximum entropy models (Ghahramani and Jordan, 1997; Smith and Eisner, 2005).

An HMM is a generative probabilistic model that generates each word x_i in the corpus conditioned on a latent variable Y_i. Each Y_i in the model takes on integral values from 1 to S, and each one is generated by the latent variable for the preceding word, Y_{i-1}. The distribution for a corpus $\mathbf{x} = (x_1, \ldots, x_N)$ given a set of state vectors $\mathbf{y} = (y_1, \ldots, y_N)$ is given by:

$$P(\mathbf{x}|\mathbf{y}) = \prod_i P(x_i|y_i)P(y_i|y_{i-1})$$

Using Expectation-Maximization (Dempster et al., 1977), it is possible to estimate the distributions for $P(x_i|y_i)$ and $P(y_i|y_{i-1})$ from unlabeled data. We use a trained HMM to determine the optimal sequence of latent states $\hat{y}_i$ using the well-known Viterbi algorithm (Rabiner, 1989). The output of this process is an integer (ranging from 1 to S) for every word x_i in the corpus; we include a new boolean feature for each possible value of y_i in our sequence labelers.

To compare our models, note that in the multinomial representation we directly model the probability that a word v appears before a word w: $P(x_{i-1} = v|x_i = w)$). In our LSA model, we find latent categories of context words z, and model the probability that a category appears before the current word w: $P(x_{i-1} = z|x_i = w)$. The HMM finds (probabilistic) categories Y for both the current word x_i and the context word x_{i-1}, and models the probability that one category follows the

other: $P(Y_i|Y_{i-1})$. Thus the HMM is our most extreme smoothing model, as it aggregates information over the greatest number of examples: for a given consecutive pair of words x_{i-1}, x_i in the test set, it aggregates over all pairs of consecutive words x'_{i-1}, x'_i where x'_{i-1} is similar to x_{i-1} and x'_i is similar to x_i.

3 Experiments

We tested the following hypotheses in our experiments:
1. Smoothing can improve the performance of a supervised sequence labeling system on words that are *rare or nonexistent* in the training data.
2. A supervised sequence labeler achieves greater accuracy on *new domains* with smoothing.
3. A supervised sequence labeler has a better *sample complexity* with smoothing.

3.1 Experimental Setup

We investigate the use of smoothing in two test systems, conditional random field (CRF) models for POS tagging and chunking. To incorporate smoothing into our models, we follow the following general procedure: first, we collect a set of unannotated text from the same domain as the test data set. Second, we train a smoothing model on the text of the training data, the test data, and the additional collection. We then automatically annotate both the training and test data with features calculated from the distributional representation. Finally, we train the CRF model on the annotated training set and apply it to the test set.

We use an open source CRF software package designed by Sunita Sajarwal and William W. Cohen to implement our CRF models.[1] We use a set of boolean features listed in Table 1.

Our baseline CRF system for POS tagging follows the model described by Lafferty *et al.*(2001). We include transition features between pairs of consecutive tag variables, features between tag variables and words, and a set of orthographic features that Lafferty *et al.* found helpful for performance on OOV words. Our smoothed models add features computed from the distributional representations, as discussed above.

Our chunker follows the system described by Sha and Pereira (2003). In addition to the transition, word-level, and orthographic features, we include features relating automatically-generated POS tags and the chunk labels. Unlike Sha and Pereira, we exclude features relating consecutive pairs of words and a chunk label, or features relating consecutive tag labels and a chunk label, in order to expedite our experiments. We found that including such features does improve chunking F1 by approximately 2%, but it also significantly slows down CRF training.

[1] Available from http://sourceforge.net/projects/crf/

CRF Feature Set	
Transition	z_i=z z_i=z and z_{i-1}=z'
Word	x_i=w and z_i=z
POS	t_i=t and z_i=z
Orthography	for every $s \in$ {-ing, -ogy, -ed, -s, -ly, -ion, -tion, -ity}, suffix(x_i)= s and z_i=z x_i is capitalized and $z_i = z$ x_i has a digit and $z_i = z$
TF, TF-IDF, and LSA features	for every context type v, $F_v^{left}(x_i)$ and $F_v^{right}(x_i)$
HMM features	y_i=y and $z_i = z$

Table 1: **Features used in our CRF systems.** z_i variables represent labels to be predicted, t_i represent tags (for the chunker), and x_i represent word tokens. All features are boolean except for the TF, TF-IDF, and LSA features.

3.2 Rare Word Accuracy

For these experiments, we use the Wall Street Journal portion of the Penn Treebank (Marcus et al., 1993). Following the CoNLL shared task from 2000, we use sections 15-18 of the Penn Treebank for our labeled training data for the supervised sequence labeler in all experiments (Tjong et al., 2000). For the tagging experiments, we train and test using the gold standard POS tags contained in the Penn Treebank. For the chunking experiments, we train and test with POS tags that are automatically generated by a standard tagger (Brill, 1994). We tested the accuracy of our models for chunking and POS tagging on section 20 of the Penn Treebank, which corresponds to the test set from the CoNLL 2000 task.

Our distributional representations are trained on sections 2-22 of the Penn Treebank. Because we include the text from the train and test sets in our training data for the distributional representations, we do not need to worry about smoothing them — when they are decoded on the test set, they

Freq:	0	1	2	0-2	all
#Samples	438	508	588	1534	46661
Baseline	.62	.77	.81	.74	.93
TF	.76	.72	.77	.75	.92
TF-IDF	**.82**	.75	.76	.78	.94
LSA	.78	.80	.77	.78	.94
HMM	.73	**.81**	**.86**	**.80**	**.94**

Table 2: **POS tagging accuracy: our HMM-smoothed tagger outperforms the baseline tagger by 6% on rare words.** Differences between the baseline and the HMM are statistically significant at $p < 0.01$ for the OOV, 0-2, and all cases using the two-tailed Chi-squared test with 1 degree of freedom.

Freq:	0	1	2	0-2	all
#Samples	133	199	231	563	21900
Baseline	.69	.75	.81	.76	.90
TF	.70	.82	.79	.77	.89
TF-IDF	.77	.77	.80	.78	.90
LSA	.84	.82	.83	.84	.90
HMM	**.90**	**.85**	**.85**	**.86**	**.93**

Table 3: **Chunking F1: our HMM-smoothed chunker outperforms the baseline CRF chunker by 0.21 on chunks that begin with OOV words, and 0.10 on chunks that begin with rare words.**

will not encounter any previously unseen words. However, to speed up training during our experiments and, in some cases, to avoid running out of memory, we replaced words appearing twice or fewer times in the data with the special symbol `*UNKNOWN*`. In addition, all numbers were replaced with another special symbol. For the LSA model, we had to use a more drastic cutoff to fit the singular value decomposition computation into memory: we replaced words appearing 10 times or fewer with the `*UNKNOWN*` symbol. We initialize our HMMs randomly. We run EM ten times and take the model with the best cross-entropy on a held-out set. After experimenting with different variations of HMM models, we settled on a model with 80 latent states as a good compromise between accuracy and efficiency.

For our POS tagging experiments, we measured the accuracy of the tagger on "rare" words, or words that appear at most twice in the training data. For our chunking experiments, we focus on chunks that begin with rare words, as we found that those were the most difficult for the chunker to identify correctly. So we define "rare" chunks as those that begin with words appearing at most twice in training data. To ensure that our smoothing models have enough training data for our test set, we further narrow our focus to those words that appear rarely in the labeled training data, but appear at least ten times in sections 2-22. Tables 2 and 3 show the accuracy of our smoothed models and the baseline model on tagging and chunking, respectively. The line for "all" in both tables indicates results on the complete test set.

Both our baseline tagger and chunker achieve respectable results on their respective tasks for all words, and the results were good enough for us to be satisfied that performance on rare words closely follows how a state-of-the-art supervised sequence-labeler behaves. The chunker's accuracy is roughly in the middle of the range of results for the original CoNLL 2000 shared task (Tjong et al., 2000) . While several systems have achieved slightly higher accuracy on supervised POS tagging, they are usually trained on larger training sets.

As expected, the drop-off in the baseline system's performance from all words to rare words is impressive for both tasks. Comparing performance on all terms and OOV terms, the baseline tagger's accuracy drops by 0.31, and the baseline chunker's F1 drops by 0.21. Comparing performance on all terms and rare terms, the drop is less severe but still dramatic: 0.19 for tagging and 0.15 for chunking.

Our hypothesis that smoothing would improve performance on rare terms is validated by these experiments. In fact, the more aggregation a smoothing model performs, the better it appears to be at smoothing. The HMM-smoothed system outperforms all other systems in all categories except tagging on OOV words, where TF-IDF performs best. And in most cases, the clear trend is for HMM smoothing to outperform LSA, which in turn outperforms TF and TF-IDF. HMM tagging performance on OOV terms improves by 11%, and chunking performance by 21%. Tagging performance on all of the rare terms improves by 6%, and chunking by 10%. In chunking, there is a clear trend toward larger increases in performance as words become rarer in the labeled data set, from a 0.02 improvement on words of frequency 2, to an improvement of 0.21 on OOV words.

Because the test data for this experiment is drawn from the same domain (newswire) as the

training data, the rare terms make up a relatively small portion of the overall dataset (approximately 4% of both the tagged words and the chunks). Still, the increased performance by the HMM-smoothed model on the rare-word subset contributes in part to an increase in performance on the overall dataset of 1% for tagging and 3% for chunking. In our next experiment, we consider a common scenario where rare terms make up a much larger fraction of the test data.

3.3 Domain Adaptation

For our experiment on domain adaptation, we focus on NP chunking and POS tagging, and we use the labeled training data from the CoNLL 2000 shared task as before. For NP chunking, we use 198 sentences from the biochemistry domain in the Open American National Corpus (OANC) (Reppen et al., 2005) as or our test set. We manually tagged the test set with POS tags and NP chunk boundaries. The test set contains 5330 words and a total of 1258 NP chunks. We used sections 15-18 of the Penn Treebank as our labeled training set, including the gold standard POS tags. We use our best-performing smoothing model, the HMM, and train it on sections 13 through 19 of the Penn Treebank, plus the written portion of the OANC that contains journal articles from biochemistry (40,727 sentences). We focus on chunks that begin with words appearing 0-2 times in the labeled training data, and appearing at least ten times in the HMM's training data. Table 4 contains our results. For our POS tagging experiments, we use 561 MEDLINE sentences (9576 words) from the Penn BioIE project (PennBioIE, 2005), a test set previously used by Blitzer *et al.*(2006). We use the same experimental setup as Blitzer *et al.*: 40,000 manually tagged sentences from the Penn Treebank for our labeled training data, and all of the unlabeled text from the Penn Treebank plus their MEDLINE corpus of 71,306 sentences to train our HMM. We report on tagging accuracy for all words and OOV words in Table 5. This table also includes results for two previous systems as reported by Blitzer *et al.* (2006): the semi-supervised Alternating Structural Optimization (ASO) technique and the Structural Correspondence Learning (SCL) technique for domain adaptation.

Note that this test set for NP chunking contains a much higher proportion of rare and OOV words: 23% of chunks begin with an OOV word, and 29% begin with a rare word, as compared with 1% and 4%, respectively, for NP chunks in the test set from the original domain. The test set for tagging also contains a much higher proportion: 23% OOV words, as compared with 1% in the original domain. Because of the increase in the number of rare words, the baseline chunker's overall performance drops by 4% compared with performance on WSJ data, and the baseline tagger's overall performance drops by 5% in the new domain.

The performance improvements for both the smoothed NP chunker and tagger are again impressive: there is a 12% improvement on OOV words, and a 10% overall improvement on rare words for chunking; the tagger shows an 8% improvement on OOV words compared to out baseline and a 3% improvement on OOV words compared to the SCL model. The resulting performance of the smoothed NP chunker is almost identical to its performance on the WSJ data. Through smoothing, the chunker not only improves by 5%

		Baseline			HMM		
Freq.	#	R	P	F1	R	P	F1
0	284	.74	.70	.72	.80	**.89**	.84
1	39	.85	.87	.86	.92	.88	.90
2	39	.79	.86	.83	.92	.90	.91
0-2	362	.75	.73	.74	**.82**	**.89**	.85
all	1258	.86	.87	.86	**.91**	**.90**	.91

Table 4: **On biochemistry journal data from the OANC, our HMM-smoothed NP chunker outperforms the baseline CRF chunker by 0.12 (F1) on chunks that begin with OOV words, and by 0.05 (F1) on all chunks.** Results in bold are statistically significantly different from the baseline results at $p < 0.05$ using the two-tailed Fisher's exact test. We did not perform significance tests for F1.

Model	All words	Unknown words
Baseline	88.3	67.3
ASO	88.4	70.9
SCL	88.9	72.0
HMM	**90.5**	**75.2**

Table 5: **On biomedical data from the Penn BioIE project, our HMM-smoothed tagger outperforms the SCL tagger by 3% (accuracy) on OOV words, and by 1.6% (accuracy) on all words.** Differences between the smoothed tagger and the SCL tagger are significant at $p < .001$ for all words and for OOV words, using the Chi-squared test with 1 degree of freedom.

in F1 over the baseline system on all words, it in fact outperforms our baseline NP chunker on the WSJ data. 60% of this improvement comes from improved accuracy on rare words.

The performance of our HMM-smoothed chunker caused us to wonder how well the chunker could work without some of its other features. We removed all tag features and all features for word types that appear fewer than 20 times in training. This chunker achieves 0.91 F1 on OANC data, and 0.93 F1 on WSJ data, outperforming the baseline system in both cases. It has only 20% as many features as the baseline chunker, greatly improving its training time. Thus our smoothing features are more valuable to the chunker than features from POS tags and features for all but the most common words. Our results point to the exciting possibility that with smoothing, we may be able to train a sequence-labeling system on a small labeled sample, and have it apply generally to other domains. Exactly what size training set we need is a question that we address next.

3.4 Sample Complexity

Our complete system consists of two learned components, a supervised CRF system and an unsupervised smoothing model. We measure the sample complexity of each component separately. To measure the sample complexity of the supervised CRF, we use the same experimental setup as in the chunking experiment on WSJ text, but we vary the amount of labeled data available to the CRF. We take ten random samples of a fixed size from the labeled training set, train a chunking model on each subset, and graph the F1 on the labeled test set, averaged over the ten runs, in Figure 1. To measure the sample complexity of our HMM with respect to unlabeled text, we use the full labeled training set and vary the amount of unlabeled text available to the HMM. At minimum, we use the text available in the labeled training and test sets, and then add random subsets of the Penn Treebank, sections 2-22. For each subset size, we take ten random samples of the unlabeled text, train an HMM and then a chunking model, and graph the F1 on the labeled test set averaged over the ten runs in Figure 2.

The results from our labeled sample complexity experiment indicate that sample complexity is drastically reduced by HMM smoothing. On rare chunks, the smoothed system reaches 0.78 F1 using only 87 labeled training sentences, a level that the baseline system never reaches, even with 6933

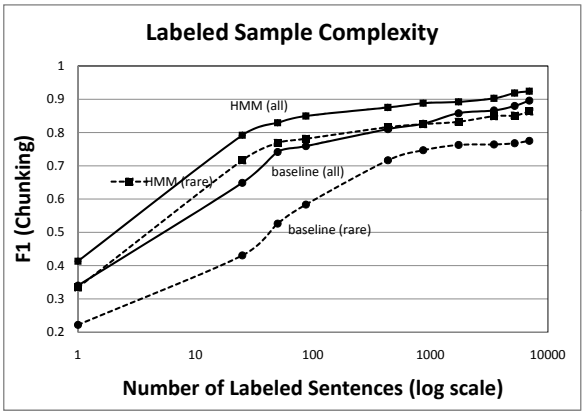


Figure 1: **The smoothed NP chunker requires less than 10% of the samples needed by the baseline chunker to achieve .83 F1, and the same for .88 F1.**

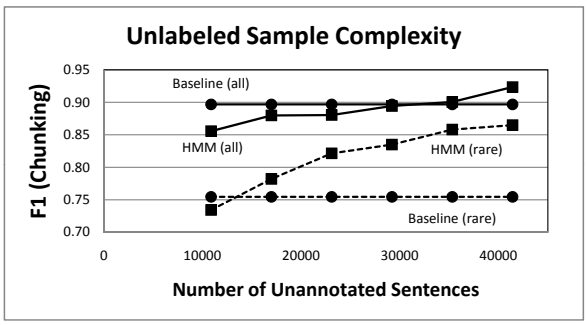


Figure 2: **By leveraging plentiful unannotated text, the smoothed chunker soon outperforms the baseline.**

labeled sentences. On the overall data set, the smoothed system reaches 0.83 F1 with 50 labeled sentences, which the baseline does not reach until it has 867 labeled sentences. With 434 labeled sentences, the smoothed system reaches 0.88 F1, which the baseline system does not reach until it has 5200 labeled samples.

Our unlabeled sample complexity results show that even with access to a small amount of unlabeled text, 6000 sentences more than what appears in the training and test sets, smoothing using the HMM yields 0.78 F1 on rare chunks. However, the smoothed system requires 25,000 more sentences before it outperforms the baseline system on all chunks. No peak in performance is reached, so further improvements are possible with more unlabeled data. Thus smoothing is optimizing performance for the case where unlabeled data is plentiful and labeled data is scarce, as we would hope.

4 Related Work

To our knowledge, only one previous system — the REALM system for sparse information extrac-

tion — has used HMMs as a feature representation for other applications. REALM uses an HMM trained on a large corpus to help determine whether the arguments of a candidate relation are of the appropriate type (Downey et al., 2007). We extend and generalize this smoothing technique and apply it to common NLP applications involving supervised sequence-labeling, and we provide an in-depth empirical analysis of its performance.

Several researchers have previously studied methods for using unlabeled data for tagging and chunking, either alone or as a supplement to labeled data. Ando and Zhang develop a semi-supervised chunker that outperforms purely supervised approaches on the CoNLL 2000 dataset (Ando and Zhang, 2005). Recent projects in semi-supervised (Toutanova and Johnson, 2007) and unsupervised (Biemann et al., 2007; Smith and Eisner, 2005) tagging also show significant progress. Unlike these systems, our efforts are aimed at using unlabeled data to find distributional representations that work well on rare terms, making the supervised systems more applicable to other domains and decreasing their sample complexity.

HMMs have been used many times for POS tagging and chunking, in supervised, semi-supervised, and in unsupervised settings (Banko and Moore, 2004; Goldwater and Griffiths, 2007; Johnson, 2007; Zhou, 2004). We take a novel perspective on the use of HMMs by using them to compute features of each token in the data that represent the distribution over that token's contexts. Our technique lets the HMM find parameters that maximize cross-entropy, and then uses labeled data to learn the best mapping from the HMM categories to the POS categories.

Smoothing in NLP usually refers to the problem of smoothing n-gram models. Sophisticated smoothing techniques like modified Kneser-Ney and Katz smoothing (Chen and Goodman, 1996) smooth together the predictions of unigram, bigram, trigram, and potentially higher n-gram sequences to obtain accurate probability estimates in the face of data sparsity. Our task differs in that we are primarily concerned with the case where even the unigram model (single word) is rarely or never observed in the labeled training data.

Sparsity for low-order contexts has recently spurred interest in using latent variables to represent distributions over contexts in language models. While n-gram models have traditionally dominated in language modeling, two recent efforts develop latent-variable probabilistic models that rival and even surpass n-gram models in accuracy (Blitzer et al., 2005; Mnih and Hinton, 2007). Several authors investigate neural network models that learn not just one latent state, but rather a vector of latent variables, to represent each word in a language model (Bengio et al., 2003; Emami et al., 2003; Morin and Bengio, 2005).

One of the benefits of our smoothing technique is that it allows for domain adaptation, a topic that has received a great deal of attention from the NLP community recently. Unlike our technique, in most cases researchers have focused on the scenario where labeled training data is available in both the source and the target domain (*e.g.*, (Daumé III, 2007; Chelba and Acero, 2004; Daumé III and Marcu, 2006)). Our technique uses unlabeled training data from the target domain, and is thus applicable more generally, including in web processing, where the domain and vocabulary is highly variable, and it is extremely difficult to obtain labeled data that is representative of the test distribution. When labeled target-domain data is available, instance weighting and similar techniques can be used in combination with our smoothing technique to improve our results further, although this has not yet been demonstrated empirically. HMM-smoothing improves on the most closely related work, the Structural Correspondence Learning technique for domain adaptation (Blitzer et al., 2006), in experiments.

5 Conclusion and Future Work

Our study of smoothing techniques demonstrates that by aggregating information across many unannotated examples, it is possible to find accurate distributional representations that can provide highly informative features to supervised sequence labelers. These features help improve sequence labeling performance on rare word types, on domains that differ from the training set, and on smaller training sets.

Further experiments are of course necessary to investigate distributional representations as smoothing techniques. One particularly promising area for further study is the combination of smoothing and instance weighting techniques for domain adaptation. Whether the current techniques are applicable to structured prediction tasks, like parsing and relation extraction, also deserves future attention.

References

Rie Kubota Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *ACL*.

Michele Banko and Robert C. Moore. 2004. Part of speech tagging in context. In *COLING*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

C. Biemann, C. Giuliano, and A. Gliozzo. 2007. Unsupervised pos tagging supporting supervised methods. *Proceeding of RANLP-07*.

J. Blitzer, A. Globerson, and F. Pereira. 2005. Distributed latent variable models of lexical cooccurrences. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.

E. Brill. 1994. Some Advances in Rule-Based Part of Speech Tagging. In *AAAI*, pages 722–727, Seattle, Washington.

Ciprian Chelba and Alex Acero. 2004. Adaptation of maximum entropy classifier: Little data can help a lot. In *EMNLP*.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318, Morristown, NJ, USA. Association for Computational Linguistics.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *ACL*.

S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

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

Doug Downey, Stefan Schoenmackers, and Oren Etzioni. 2007. Sparse information extraction: Unsupervised language models to the rescue. In *ACL*.

A. Emami, P. Xu, and F. Jelinek. 2003. Using a connectionist model in a syntactical based language model. In *Proceedings of the International Conference on Spoken Language Processing*, pages 372–375.

Zoubin Ghahramani and Michael I. Jordan. 1997. Factorial hidden markov models. *Machine Learning*, 29(2-3):245–273.

Sharon Goldwater and Thomas L. Griffiths. 2007. A fully bayesian approach to unsupervised part-of-speech tagging. In *ACL*.

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers. In *EMNLP*.

J. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*.

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

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*, pages 641–648, New York, NY, USA. ACM.

F. Morin and Y. Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, pages 246–252.

PennBioIE. 2005. Mining the bibliome project. *http://bioie.ldc.upenn.edu/*.

Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285.

Randi Reppen, Nancy Ide, and Keith Suderman. 2005. American national corpus (ANC) second release. Linguistic Data Consortium.

F. Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology - NAACL*.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 354–362, Ann Arbor, Michigan, June.

Erik F. Tjong, Kim Sang, and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the 4th Conference on Computational Natural Language Learning*, pages 127–132.

Kristina Toutanova and Mark Johnson. 2007. A bayesian LDA-based model for semi-supervised part-of-speech tagging. In *NIPS*.

GuoDong Zhou. 2004. Discriminative hidden Markov modeling with long state dependence using a kNN ensemble. In *COLING*.

Minimized Models for Unsupervised Part-of-Speech Tagging

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

Abstract

We describe a novel method for the task of unsupervised POS tagging with a dictionary, one that uses integer programming to explicitly search for the smallest model that explains the data, and then uses EM to set parameter values. We evaluate our method on a standard test corpus using different standard tagsets (a 45-tagset as well as a smaller 17-tagset), and show that our approach performs better than existing state-of-the-art systems in both settings.

1 Introduction

In recent years, we have seen increased interest in using unsupervised methods for attacking different NLP tasks like part-of-speech (POS) tagging. The classic Expectation Maximization (EM) algorithm has been shown to perform poorly on POS tagging, when compared to other techniques, such as Bayesian methods.

In this paper, we develop new methods for unsupervised part-of-speech tagging. We adopt the problem formulation of Merialdo (1994), in which we are given a raw word sequence and a dictionary of legal tags for each word type. The goal is to tag each word token so as to maximize accuracy against a gold tag sequence. Whether this is a realistic problem set-up is arguable, but an interesting collection of methods and results has accumulated around it, and these can be clearly compared with one another.

We use the standard test set for this task, a 24,115-word subset of the Penn Treebank, for which a gold tag sequence is available. There are 5,878 word types in this test set. We use the standard tag dictionary, consisting of 57,388 word/tag pairs derived from the entire Penn Treebank.[1] 8,910 dictionary entries are relevant to the 5,878 word types in the test set. Per-token ambiguity is about 1.5 tags/token, yielding approximately 10^{6425} possible ways to tag the data. There are 45 distinct grammatical tags. In this set-up, there are no unknown words.

Figure 1 shows prior results for this problem. While the methods are quite different, they all make use of two common model elements. One is a probabilistic n-gram tag model $P(t_i|t_{i-n+1}...t_{i-1})$, which we call the *grammar*. The other is a probabilistic word-given-tag model $P(w_i|t_i)$, which we call the *dictionary*.

The classic approach (Merialdo, 1994) is expectation-maximization (EM), where we estimate grammar and dictionary probabilities in order to maximize the probability of the observed word sequence:

$$\begin{aligned} P(w_1...w_n) &= \sum_{t_1...t_n} P(t_1...t_n) \cdot P(w_1...w_n|t_1...t_n) \\ &\approx \sum_{t_1...t_n} \prod_{i=1}^{n} P(t_i|t_{i-2}\ t_{i-1}) \cdot P(w_i|t_i) \end{aligned}$$

Goldwater and Griffiths (2007) report 74.5% accuracy for EM with a 3-gram tag model, which we confirm by replication. They improve this to 83.9% by employing a fully Bayesian approach which integrates over all possible parameter values, rather than estimating a single distribution. They further improve this to 86.8% by using priors that favor sparse distributions. Smith and Eisner (2005) employ a *contrastive estimation* tech-

[1] As (Banko and Moore, 2004) point out, unsupervised tagging accuracy varies wildly depending on the dictionary employed. We follow others in using a fat dictionary (with 49,206 distinct word types), rather than a thin one derived only from the test set.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 504–512,
Suntec, Singapore, 2-7 August 2009.

	System	Tagging accuracy (%) on 24,115-word corpus
1.	Random baseline (for each word, pick a random tag from the alternatives given by the word/tag dictionary)	64.6
2.	EM with 2-gram tag model	81.7
3.	EM with 3-gram tag model	74.5
4a.	Bayesian method (Goldwater and Griffiths, 2007)	83.9
4b.	Bayesian method with sparse priors (Goldwater and Griffiths, 2007)	86.8
5.	CRF model trained using contrastive estimation (Smith and Eisner, 2005)	88.6
6.	EM-HMM tagger provided with good initial conditions (Goldberg et al., 2008) (*uses linguistic constraints and manual adjustments to the dictionary)	91.4*

Figure 1: Previous results on unsupervised POS tagging using a dictionary (Merialdo, 1994) on the full 45-tag set. All other results reported in this paper (unless specified otherwise) are on the 45-tag set as well.

nique, in which they automatically generate negative examples and use CRF training.

In more recent work, Toutanova and Johnson (2008) propose a Bayesian LDA-based generative model that in addition to using sparse priors, explicitly groups words into ambiguity classes. They show considerable improvements in tagging accuracy when using a coarser-grained version (with 17-tags) of the tag set from the Penn Treebank.

Goldberg et al. (2008) depart from the Bayesian framework and show how EM can be used to learn good POS taggers for Hebrew and English, when provided with good initial conditions. They use language specific information (like word contexts, syntax and morphology) for learning initial P($t|w$) distributions and also use linguistic knowledge to apply constraints on the tag sequences allowed by their models (e.g., the tag sequence "V V" is disallowed). Also, they make other manual adjustments to reduce noise from the word/tag dictionary (e.g., reducing the number of tags for "the" from six to just one). In contrast, we keep all the original dictionary entries derived from the Penn Treebank data for our experiments.

The literature omits one other baseline, which is EM with a 2-gram tag model. Here we obtain 81.7% accuracy, which is better than the 3-gram model. It seems that EM with a 3-gram tag model runs amok with its freedom. For the rest of this paper, we will limit ourselves to a 2-gram tag model.

2 What goes wrong with EM?

We analyze the tag sequence output produced by EM and try to see where EM goes wrong. The overall POS tag distribution learnt by EM is relatively uniform, as noted by Johnson (2007), and it tends to assign equal number of tokens to each tag label whereas the real tag distribution is highly skewed. The Bayesian methods overcome this effect by using priors which favor sparser distributions. But it is not easy to model such priors into EM learning. As a result, EM exploits a lot of rare tags (like FW = *foreign word*, or SYM = *symbol*) and assigns them to common word types (*in, of,* etc.).

We can compare the tag assignments from the gold tagging and the EM tagging (Viterbi tag sequence). The table below shows tag assignments (and their counts in parentheses) for a few word types which occur frequently in the test corpus.

word/tag dictionary	Gold tagging	EM tagging
in → {IN, RP, RB, NN, FW, RBR}	IN (355) RP (3) FW (0)	IN (0) RP (0) FW (358)
of → {IN, RP, RB}	IN (567) RP (0)	IN (0) RP (567)
on → {IN,RP, RB}	RP (5) IN (129) RB (0)	RP (127) IN (0) RB (7)
a → {DT, JJ, IN, LS, FW, SYM, NNP}	DT (517) SYM (0)	DT (0) SYM (517)

We see how the rare tag labels (like FW, SYM, etc.) are abused by EM. As a result, many word tokens which occur very frequently in the corpus are incorrectly tagged with rare tags in the EM tagging output.

We also look at things more globally. We investigate the Viterbi tag sequence generated by EM training and count how many distinct tag bigrams there are in that sequence. We call this the *observed grammar size*, and it is 915. That is, in tagging the 24,115 test tokens, EM uses 915 of the available 45 × 45 = 2025 tag bigrams.[2] The advantage of the observed grammar size is that we

[2]We contrast observed size with the *model size* for the grammar, which we define as the number of P($t_2|t_1$) entries in EM's trained tag model that exceed 0.0001 probability.

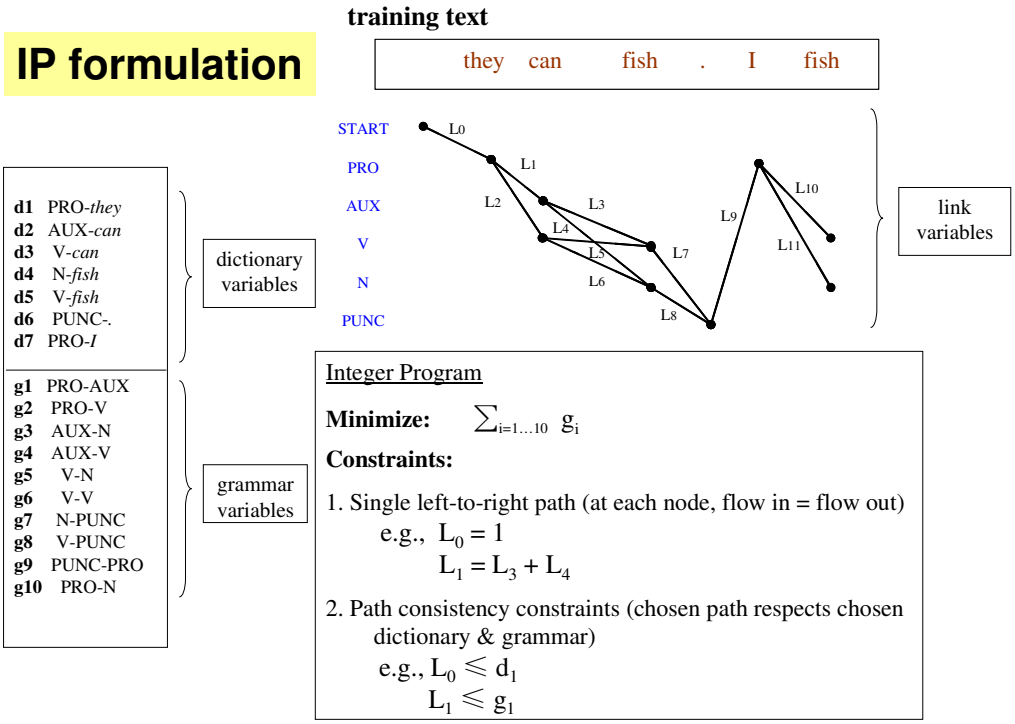


Figure 2: Integer Programming formulation for finding the smallest grammar that explains a given word sequence. Here, we show a sample word sequence and the corresponding IP network generated for that sequence.

can compare it with the gold tagging's observed grammar size, which is 760. So we can safely say that EM is learning a grammar that is too big, still abusing its freedom.

3 Small Models

Bayesian sparse priors aim to create small models. We take a different tack in the paper and directly ask: *What is the smallest model that explains the text?* Our approach is related to minimum description length (MDL). We formulate our question precisely by asking which tag sequence (of the 10^{6425} available) has the smallest observed grammar size. The answer is 459. That is, there exists a tag sequence that contains 459 distinct tag bigrams, and no other tag sequence contains fewer.

We obtain this answer by formulating the problem in an integer programming (IP) framework. Figure 2 illustrates this with a small sample word sequence. We create a network of possible taggings, and we assign a binary variable to each link in the network. We create constraints to ensure that those link variables receiving a value of 1 form a left-to-right path through the tagging network, and that all other link variables receive a value of 0. We accomplish this by requiring the sum of the links entering each node to equal to the sum of the links leaving each node. We also create variables for every possible tag bigram and word/tag dictionary entry. We constrain link variable assignments to respect those grammar and dictionary variables. For example, we do not allow a link variable to "activate" unless the corresponding grammar variable is also "activated". Finally, we add an objective function that minimizes the number of grammar variables that are assigned a value of 1.

Figure 3 shows the IP solution for the example word sequence from Figure 2. Of course, a small grammar size does not necessarily correlate with higher tagging accuracy. For the small toy example shown in Figure 3, the correct tagging is "PRO AUX V . PRO V" (with 5 tag pairs), whereas the IP tries to minimize the grammar size and picks another solution instead.

For solving the integer program, we use CPLEX software (a commercial IP solver package). Alternatively, there are other programs such as *lp_solve*, which are free and publicly available for use. Once we create an integer program for the full test corpus, and pass it to CPLEX, the solver returns an

word sequence: *they can fish . I fish*

Tagging						Grammar Size
PRO	AUX	N	.	PRO	N	5
PRO	AUX	V	.	PRO	N	5
PRO	AUX	N	.	PRO	V	5
PRO	AUX	V	.	PRO	V	5
PRO	V	N	.	PRO	N	5
PRO	V	V	.	PRO	N	5
PRO	V	N	.	PRO	V	**4**
PRO	V	V	.	PRO	V	**4**

Figure 3: Possible tagging solutions and corresponding grammar sizes for the sample word sequence from Figure 2 using the given dictionary and grammar. The IP solver finds the smallest grammar set that can explain the given word sequence. In this example, there exist two solutions that each contain only 4 tag pair entries, and IP returns one of them.

objective function value of 459.[3]

CPLEX also returns a tag sequence via assignments to the link variables. However, there are actually 10^{4378} tag sequences compatible with the 459-sized grammar, and our IP solver just selects one at random. We find that of all those tag sequences, the worst gives an accuracy of 50.8%, and the best gives an accuracy of 90.3%. We also note that CPLEX takes 320 seconds to return the optimal solution for the integer program corresponding to this particular test data (24,115 tokens with the 45-tag set). It might be interesting to see how the performance of the IP method (in terms of time complexity) is affected when scaling up to larger data and bigger tagsets. We leave this as part of future work. But we do note that it is possible to obtain less than optimal solutions faster by interrupting the CPLEX solver.

4 Fitting the Model

Our IP formulation can find us a small model, but it does not attempt to fit the model to the data. Fortunately, we can use EM for that. We still give EM the full word/tag dictionary, but now we constrain its initial grammar model to the 459 tag bigrams identified by IP. Starting with uniform probabilities, EM finds a tagging that is 84.5% accurate, substantially better than the 81.7% originally obtained with the fully-connected grammar. So we see a benefit to our explicit small-model approach. While EM does not find the most accurate sequence consistent with the IP grammar (90.3%), it finds a relatively good one.

[3]Note that the grammar identified by IP is not uniquely minimal. For the same word sequence, there exist other minimal grammars having the same size (459 entries). In our experiments, we choose the first solution returned by CPLEX.

	in	*on*
word/tag dictionary	IN RP RB NN FW RBR	IN RP RB
observed EM dictionary	FW (358)	RP (127) RB (7)
observed IP+EM dictionary	IN (349) RB (9)	IN (126) RB (8)
observed gold dictionary	IN (355) RB (3)	IN (129) RP (5)

Figure 4: Examples of tagging obtained from different systems for prepositions *in* and *on*.

The IP+EM tagging (with 84.5% accuracy) has some interesting properties. First, the dictionary we observe from the tagging is of higher quality (with fewer spurious tagging assignments) than the one we observe from the original EM tagging. Figure 4 shows some examples.

We also measure the quality of the two observed grammars/dictionaries by computing their precision and recall against the grammar/dictionary we observe in the gold tagging.[4] We find that precision of the observed grammar increases from 0.73 (EM) to 0.94 (IP+EM). In addition to removing many bad tag bigrams from the grammar, IP minimization also removes some of the good ones, leading to lower recall (EM = 0.87, IP+EM = 0.57). In the case of the observed dictionary, using a smaller grammar model does not affect the precision (EM = 0.91, IP+EM = 0.89) or recall (EM = 0.89, IP+EM = 0.89).

During EM training, the smaller grammar with fewer bad tag bigrams helps to restrict the dictionary model from making too many bad choices that EM made earlier. Here are a few examples of bad dictionary entries that get removed when we use the minimized grammar for EM training:

$$\begin{aligned} \textit{in} &\rightarrow \text{FW} \\ \textit{a} &\rightarrow \text{SYM} \\ \textit{of} &\rightarrow \text{RP} \\ \textit{In} &\rightarrow \text{RBR} \end{aligned}$$

During EM training, the minimized grammar

[4]For any observed grammar or dictionary X,

$$\text{Precision (X)} = \frac{|\{X\} \cap \{observed_{gold}\}|}{|\{X\}|}$$
$$\text{Recall (X)} = \frac{|\{X\} \cap \{observed_{gold}\}|}{|\{observed_{gold}\}|}$$

	Model	Tagging accuracy on 24,115-word corpus	Observed size grammar(G), dictionary(D)	Model size grammar(G), dictionary(D)
1.	EM baseline with full grammar + full dictionary	81.7	G=915, D=6295	G=935, D=6430
2.	EM constrained with minimized IP-grammar + full dictionary	84.5	G=459, D=6318	G=459, D=6414
3.	EM constrained with full grammar + dictionary from (2)	91.3	G=606, D=6245	G=612, D=6298
4.	EM constrained with grammar from (3) + full dictionary	91.5	G=593, D=6285	G=600, D=6373
5.	EM constrained with full grammar + dictionary from (4)	**91.6**	G=603, D=6280	G=618, D=6337

Figure 5: Percentage of word tokens tagged correctly by different models. The *observed sizes* and *model sizes* of grammar (G) and dictionary (D) produced by these models are shown in the last two columns.

helps to eliminate many incorrect entries (i.e., zero out model parameters) from the dictionary, thereby yielding an improved dictionary model. So using the minimized grammar (which has higher precision) helps to improve the quality of the chosen dictionary (examples shown in Figure 4). This in turn helps improve the tagging accuracy from 81.7% to 84.5%. It is clear that the IP-constrained grammar is a better choice to run EM on than the full grammar.

Note that we used a very small IP-grammar (containing only 459 tag bigrams) during EM training. In the process of minimizing the grammar size, IP ends up removing many good tag bigrams from our grammar set (as seen from the low measured recall of 0.57 for the observed grammar). Next, we proceed to recover some good tag bigrams and expand the grammar in a restricted fashion by making use of the higher-quality dictionary produced by the IP+EM method. We now run EM again on the full grammar (all possible tag bigrams) in combination with this good dictionary (containing fewer entries than the full dictionary). Unlike the original training with full grammar, where EM could choose any tag bigram, now the choice of grammar entries is constrained by the good dictionary model that we provide EM with. This allows EM to recover some of the good tag pairs, and results in a good grammar-dictionary combination that yields better tagging performance.

With these improvements in mind, we embark on an alternating scheme to find better models and taggings. We run EM for multiple passes, and in each pass we alternately constrain either the grammar model or the dictionary model. The procedure is simple and proceeds as follows:

1. Run EM constrained to the last trained dictionary, but provided with a full grammar.[5]

2. Run EM constrained to the last trained grammar, but provided with a full dictionary.

3. Repeat steps 1 and 2.

We notice significant gains in tagging performance when applying this technique. The tagging accuracy increases at each step and finally settles at a high of 91.6%, which outperforms the existing state-of-the-art systems for the 45-tag set. The system achieves a better accuracy than the 88.6% from Smith and Eisner (2005), and even surpasses the 91.4% achieved by Goldberg et al. (2008) without using any additional linguistic constraints or manual cleaning of the dictionary. Figure 5 shows the tagging performance achieved at each step. We found that it is the elimination of incorrect entries from the dictionary (and grammar) and not necessarily the initialization weights from previous EM training, that results in the tagging improvements. Initializing the last trained dictionary or grammar at each step with uniform weights also yields the same tagging improvements as shown in Figure 5.

We find that the observed grammar also improves, growing from 459 entries to 603 entries, with precision increasing from 0.94 to 0.96, and recall increasing from 0.57 to 0.76. The figure also shows the model's internal grammar and dictionary sizes.

Figure 6 and 7 show how the precision/recall of the observed grammar and dictionary varies for different models from Figure 5. In the case of the observed grammar (Figure 6), precision increases

[5]For all experiments, EM training is allowed to run for 40 iterations or until the likelihood ratios between two subsequent iterations reaches a value of 0.99999, whichever occurs earlier.

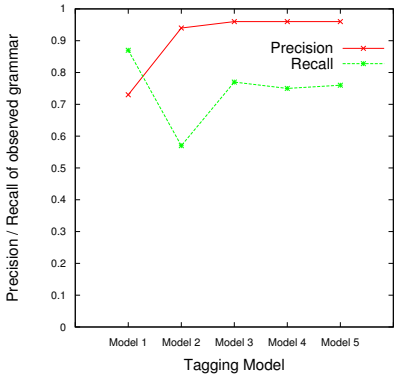


Figure 6: Comparison of observed grammars from the model tagging vs. gold tagging in terms of precision and recall measures.

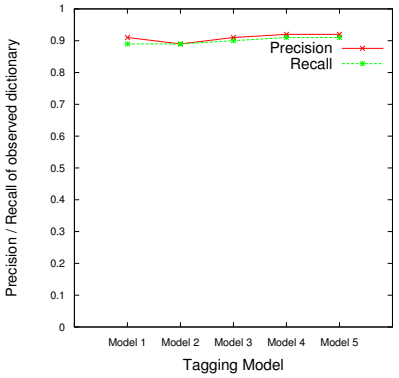


Figure 7: Comparison of observed dictionaries from the model tagging vs. gold tagging in terms of precision and recall measures.

Model	Tagging accuracy on 24,115-word corpus	
	no-restarts	with 100 restarts
1. Model 1 (EM baseline)	81.7	83.8
2. Model 2	84.5	84.5
3. Model 3	91.3	91.8
4. Model 4	91.5	91.8
5. Model 5	91.6	91.8

Figure 8: Effect of random restarts (during EM training) on tagging accuracy.

at each step, whereas recall drops initially (owing to the grammar minimization) but then picks up again. The precision/recall of the observed dictionary on the other hand, is not affected by much.

5 Restarts and More Data

Multiple random restarts for EM, while not often emphasized in the literature, are key in this domain. Recall that our original EM tagging with a fully-connected 2-gram tag model was 81.7% accurate. When we execute 100 random restarts and select the model with the highest data likelihood, we get 83.8% accuracy. Likewise, when we extend our alternating EM scheme to 100 random restarts at each step, we improve our tagging accuracy from 91.6% to 91.8% (Figure 8).

As noted by Toutanova and Johnson (2008), there is no reason to limit the amount of unlabeled data used for training the models. Their models are trained on the entire Penn Treebank data (instead of using only the 24,115-token test data), and so are the tagging models used by Goldberg et al. (2008). But previous results from Smith and Eisner (2005) and Goldwater and Griffiths (2007) show that their models do not benefit from using more unlabeled training data. Because EM is efficient, we can extend our word-sequence training data from the 24,115-token set to the entire Penn Treebank (973k tokens). We run EM training again for Model 5 (the best model from Figure 5) but this time using 973k word tokens, and further increase our accuracy to 92.3%. This is our final result on the 45-tagset, and we note that it is higher than previously reported results.

6 Smaller Tagset and Incomplete Dictionaries

Previously, researchers working on this task have also reported results for unsupervised tagging with a smaller tagset (Smith and Eisner, 2005; Goldwater and Griffiths, 2007; Toutanova and Johnson, 2008; Goldberg et al., 2008). Their systems were shown to obtain considerable improvements in accuracy when using a 17-tagset (a coarser-grained version of the tag labels from the Penn Treebank) instead of the 45-tagset. When tagging the same standard test corpus with the smaller 17-tagset, our method is able to achieve a substantially high accuracy of 96.8%, which is the best result reported so far on this task. The table in Figure 9 shows a comparison of different systems for which tagging accuracies have been reported previously for the 17-tagset case (Goldberg et al., 2008). The first row in the table compares tagging results when using a full dictionary (i.e., a lexicon containing entries for 49,206 word types). The InitEM-HMM system from Goldberg et al. (2008) reports an accuracy of 93.8%, followed by the LDA+AC model (Latent Dirichlet Allocation model with a strong Ambiguity Class component) from Toutanova and Johnson (2008). In comparison, the Bayesian HMM (BHMM) model from Goldwater et al. (2007) and

Dict	IP+EM (24k)	InitEM-HMM	LDA+AC	CE+spl	BHMM
Full (49206 words)	**96.8** (**96.8**)	93.8	93.4	88.7	87.3
≥ 2 (2141 words)	90.6 (90.0)	89.4	**91.2**	79.5	79.6
≥ 3 (1249 words)	88.0 (86.1)	87.4	**89.7**	78.4	71

Figure 9: Comparison of different systems for English unsupervised POS tagging with 17 tags.

the CE+spl model (Contrastive Estimation with a spelling model) from Smith and Eisner (2005) report lower accuracies (87.3% and 88.7%, respectively). Our system (IP+EM) which uses integer programming and EM, gets the highest accuracy (96.8%). The accuracy numbers reported for Init-HMM and LDA+AC are for models that are trained on all the available unlabeled data from the Penn Treebank. The IP+EM models used in the 17-tagset experiments reported here were not trained on the entire Penn Treebank, but instead used a smaller section containing 77,963 tokens for estimating model parameters. We also include the accuracies for our IP+EM model when using only the 24,115 token test corpus for EM estimation (shown within parenthesis in second column of the table in Figure 9). We find that our performance does not degrade when the parameter estimation is done using less data, and our model still achieves a high accuracy of 96.8%.

6.1 Incomplete Dictionaries and Unknown Words

The literature also includes results reported in a different setting for the tagging problem. In some scenarios, a complete dictionary with entries for all word types may not be readily available to us and instead, we might be provided with an incomplete dictionary that contains entries for only frequent word types. In such cases, any word not appearing in the dictionary will be treated as an unknown word, and can be labeled with any of the tags from given tagset (i.e., for every unknown word, there are 17 tag possibilities). Some previous approaches (Toutanova and Johnson, 2008; Goldberg et al., 2008) handle unknown words explicitly using ambiguity class components conditioned on various morphological features, and this has shown to produce good tagging results, especially when dealing with incomplete dictionaries.

We follow a simple approach using just one of the features used in (Toutanova and Johnson, 2008) for assigning tag possibilities to every unknown word. We first identify the top-100 suffixes (up to 3 characters) for words in the dictionary. Using the word/tag pairs from the dictionary, we train a simple probabilistic model that predicts the tag given a particular suffix (e.g., P(VBG | *ing*) = 0.97, P(N | *ing*) = 0.0001, ...). Next, for every unknown word "w", the trained P(tag | suffix) model is used to predict the top 3 tag possibilities for "w" (using only its suffix information), and subsequently this word along with its 3 tags are added as a new entry to the lexicon. We do this for every unknown word, and eventually we have a dictionary containing entries for all the words. Once the completed lexicon (containing both correct entries for words in the lexicon and the predicted entries for unknown words) is available, we follow the same methodology from Sections 3 and 4 using integer programming to minimize the size of the grammar and then applying EM to estimate parameter values.

Figure 9 shows comparative results for the 17-tagset case when the dictionary is incomplete. The second and third rows in the table shows tagging accuracies for different systems when a cutoff of 2 (i.e., all word types that occur with frequency counts < 2 in the test corpus are removed) and a cutoff of 3 (i.e., all word types occurring with frequency counts < 3 in the test corpus are removed) is applied to the dictionary. This yields lexicons containing 2,141 and 1,249 words respectively, which are much smaller compared to the original 49,206 word dictionary. As the results in Figure 9 illustrate, the IP+EM method clearly does better than all the other systems except for the LDA+AC model. The LDA+AC model from Toutanova and Johnson (2008) has a strong ambiguity class component and uses more features to handle the unknown words better, and this contributes to the slightly higher performance in the incomplete dictionary cases, when compared to the IP+EM model.

7 Discussion

The method proposed in this paper is simple—once an integer program is produced, there are solvers available which directly give us the solution. In addition, we do not require any complex parameter estimation techniques; we train our models using simple EM, which proves to be efficient for this task. While some previous methods

word type	Gold tag	Automatic tag	# of tokens tagged incorrectly
's	POS	VBZ	173
be	VB	VBP	67
that	IN	WDT	54
New	NNP	NNPS	33
U.S.	NNP	JJ	31
up	RP	RB	28
more	RBR	JJR	27
and	CC	IN	23
have	VB	VBP	20
first	JJ	JJS	20
to	TO	IN	19
out	RP	RB	17
there	EX	RB	15
stock	NN	JJ	15
what	WP	WDT	14
one	CD	NN	14
'	POS	:	14
as	RB	IN	14
all	DT	RB	14
that	IN	RB	13

Figure 10: Most frequent mistakes observed in the model tagging (using the best model, which gives 92.3% accuracy) when compared to the gold tagging.

introduced for the same task have achieved big tagging improvements using additional linguistic knowledge or manual supervision, our models are not provided with any additional information.

Figure 10 illustrates for the 45-tag set some of the common mistakes that our best tagging model (92.3%) makes. In some cases, the model actually gets a reasonable tagging but is penalized perhaps unfairly. For example, "to" is tagged as IN by our model sometimes when it occurs in the context of a preposition, whereas in the gold tagging it is always tagged as TO. The model also gets penalized for tagging the word "U.S." as an adjective (JJ), which might be considered valid in some cases such as "the U.S. State Department". In other cases, the model clearly produces incorrect tags (e.g., "New" gets tagged incorrectly as NNPS).

Our method resembles the classic Minimum Description Length (MDL) approach for model selection (Barron et al., 1998). In MDL, there is a single objective function to (1) maximize the likelihood of observing the data, and at the same time (2) minimize the length of the model description (which depends on the model size). However, the search procedure for MDL is usually non-trivial, and for our task of unsupervised tagging, we have not found a direct objective function which we can optimize and produce good tagging results. In the past, only a few approaches utilizing MDL have been shown to work for natural language applications. These approaches employ heuristic search methods with MDL for the task of unsupervised learning of morphology of natural languages (Goldsmith, 2001; Creutz and Lagus, 2002; Creutz and Lagus, 2005). The method proposed in this paper is the first application of the MDL idea to POS tagging, and the first to use an integer programming formulation rather than heuristic search techniques. We also note that it might be possible to replicate our models in a Bayesian framework similar to that proposed in (Goldwater and Griffiths, 2007).

8 Conclusion

We presented a novel method for attacking dictionary-based unsupervised part-of-speech tagging. Our method achieves a very high accuracy (92.3%) on the 45-tagset and a higher (96.8%) accuracy on a smaller 17-tagset. The method works by explicitly minimizing the grammar size using integer programming, and then using EM to estimate parameter values. The entire process is fully automated and yields better performance than any existing state-of-the-art system, even though our models were not provided with any additional linguistic knowledge (for example, explicit syntactic constraints to avoid certain tag combinations such as "V V", etc.). However, it is easy to model some of these linguistic constraints (both at the local and global levels) directly using integer programming, and this may result in further improvements and lead to new possibilities for future research. For direct comparison to previous works, we also presented results for the case when the dictionaries are incomplete and find the performance of our system to be comparable with current best results reported for the same task.

9 Acknowledgements

This research was supported by the Defense Advanced Research Projects Agency under SRI International's prime Contract Number NBCHD040058.

References

M. Banko and R. C. Moore. 2004. Part of speech tagging in context. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

A. Barron, J. Rissanen, and B. Yu. 1998. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760.

M. Creutz and K. Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL Workshop on Morphological and Phonological Learning of*.

M. Creutz and K. Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. *Publications in Computer and Information Science, Report A81, Helsinki University of Technology, March*.

Y. Goldberg, M. Adler, and M. Elhadad. 2008. EM can find pretty good HMM POS-taggers (when given a good start). In *Proceedings of the ACL*.

J. Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

S. Goldwater and T. L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the ACL*.

M. Johnson. 2007. Why doesnt EM find good HMM POS-taggers? In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

B. Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.

N. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the ACL*.

K. Toutanova and M. Johnson. 2008. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*.

An Error-Driven Word-Character Hybrid Model for Joint Chinese Word Segmentation and POS Tagging

Canasai Kruengkrai[†‡] **Kiyotaka Uchimoto**[‡] **Jun'ichi Kazama**[‡]
Yiou Wang[‡] **Kentaro Torisawa**[‡] **Hitoshi Isahara**[†‡]
[†]Graduate School of Engineering, Kobe University
1-1 Rokkodai-cho, Nada-ku, Kobe 657-8501 Japan
[‡]National Institute of Information and Communications Technology
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0289 Japan
{canasai,uchimoto,kazama,wangyiou,torisawa,isahara}@nict.go.jp

Abstract

In this paper, we present a discriminative word-character hybrid model for joint Chinese word segmentation and POS tagging. Our word-character hybrid model offers high performance since it can handle both known and unknown words. We describe our strategies that yield good balance for learning the characteristics of known and unknown words and propose an error-driven policy that delivers such balance by acquiring examples of unknown words from particular errors in a training corpus. We describe an efficient framework for training our model based on the Margin Infused Relaxed Algorithm (MIRA), evaluate our approach on the Penn Chinese Treebank, and show that it achieves superior performance compared to the state-of-the-art approaches reported in the literature.

1 Introduction

In Chinese, word segmentation and part-of-speech (POS) tagging are indispensable steps for higher-level NLP tasks. Word segmentation and POS tagging results are required as inputs to other NLP tasks, such as phrase chunking, dependency parsing, and machine translation. Word segmentation and POS tagging in a joint process have received much attention in recent research and have shown improvements over a pipelined fashion (Ng and Low, 2004; Nakagawa and Uchimoto, 2007; Zhang and Clark, 2008; Jiang et al., 2008a; Jiang et al., 2008b).

In joint word segmentation and the POS tagging process, one serious problem is caused by unknown words, which are defined as words that are not found in a training corpus or in a system's word dictionary[1]. The word boundaries and the POS tags of unknown words, which are very difficult to identify, cause numerous errors. The word-character hybrid model proposed by Nakagawa and Uchimoto (Nakagawa, 2004; Nakagawa and Uchimoto, 2007) shows promising properties for solving this problem. However, it suffers from structural complexity. Nakagawa (2004) described a training method based on a word-based Markov model and a character-based maximum entropy model that can be completed in a reasonable time. However, this training method is limited by the generatively-trained Markov model in which informative features are hard to exploit.

In this paper, we overcome such limitations concerning both efficiency and effectiveness. We propose a new framework for training the word-character hybrid model based on the Margin Infused Relaxed Algorithm (MIRA) (Crammer, 2004; Crammer et al., 2005; McDonald, 2006). We describe k-best decoding for our hybrid model and design its loss function and the features appropriate for our task.

In our word-character hybrid model, allowing the model to learn the characteristics of both known and unknown words is crucial to achieve optimal performance. Here, we describe our strategies that yield good balance for learning these two characteristics. We propose an error-driven policy that delivers this balance by acquiring examples of unknown words from particular errors in a training corpus. We conducted our experiments on Penn Chinese Treebank (Xia et al., 2000) and compared our approach with the best previous approaches reported in the literature. Experimental results indicate that our approach can achieve state-of-the-art performance.

[1]A system's word dictionary usually consists of a word list, and each word in the list has its own POS category. In this paper, we constructed the system's word dictionary from a training corpus.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 513–521,
Suntec, Singapore, 2-7 August 2009.

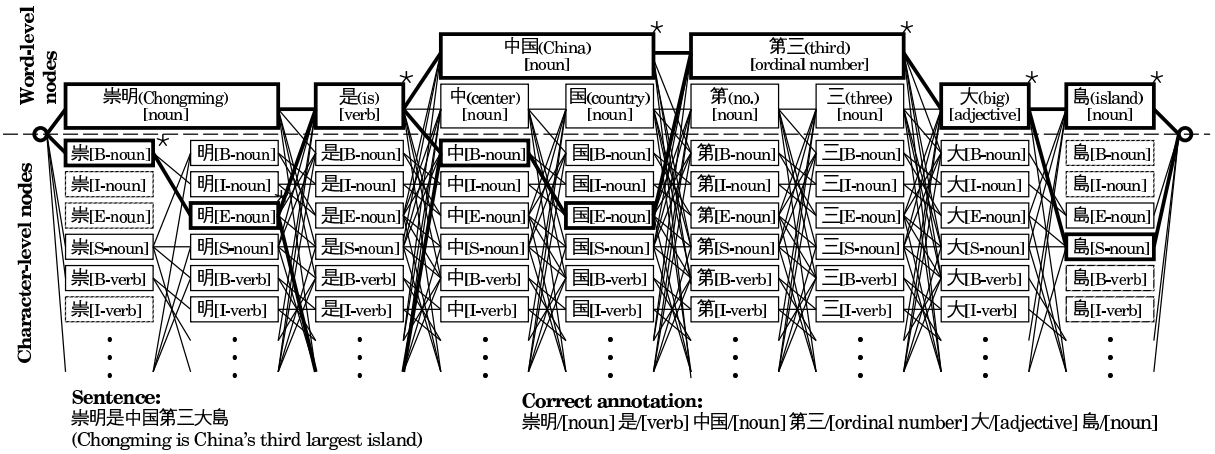


Figure 1: Lattice used in word-character hybrid model.

Tag	Description
B	Beginning character in a multi-character word
I	Intermediate character in a multi-character word
E	End character in a multi-character word
S	Single-character word

Table 1: Position-of-character (POC) tags.

The paper proceeds as follows: Section 2 gives background on the word-character hybrid model, Section 3 describes our policies for correct path selection, Section 4 presents our training method based on MIRA, Section 5 shows our experimental results, Section 6 discusses related work, and Section 7 concludes the paper.

2 Background

2.1 Problem formation

In joint word segmentation and the POS tagging process, the task is to predict a path of word hypotheses $\boldsymbol{y} = (y_1, \ldots, y_{\#\boldsymbol{y}}) = (\langle w_1, p_1 \rangle, \ldots, \langle w_{\#\boldsymbol{y}}, p_{\#\boldsymbol{y}} \rangle)$ for a given character sequence $\boldsymbol{x} = (c_1, \ldots, c_{\#\boldsymbol{x}})$, where w is a word, p is its POS tag, and a "#" symbol denotes the number of elements in each variable. The goal of our learning algorithm is to learn a mapping from inputs (unsegmented sentences) $\boldsymbol{x} \in \mathcal{X}$ to outputs (segmented paths) $\boldsymbol{y} \in \mathcal{Y}$ based on training samples of input-output pairs $\mathcal{S} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{T}$.

2.2 Search space representation

We represent the search space with a lattice based on the word-character hybrid model (Nakagawa and Uchimoto, 2007). In the hybrid model, given an input sentence, a lattice that consists of word-level and character-level nodes is constructed. Word-level nodes, which correspond to words found in the system's word dictionary, have regular POS tags. Character-level nodes have special tags where position-of-character (POC) and POS tags are combined (Asahara, 2003; Nakagawa, 2004). POC tags indicate the word-internal positions of the characters, as described in Table 1.

Figure 1 shows an example of a lattice for a Chinese sentence: "崇明是中国第三大島" (Chongming is China's third largest island). Note that some nodes and state transitions are not allowed. For example, `I` and `E` nodes cannot occur at the beginning of the lattice (marked with dashed boxes), and the transitions from `I` to `B` nodes are also forbidden. These nodes and transitions are ignored during the lattice construction processing.

In the training phase, since several paths (marked in bold) can correspond to the correct analysis in the annotated corpus, we need to select one correct path $\boldsymbol{y}_t$ as a reference for training.[2] The next section describes our strategies for dealing with this issue.

With this search space representation, we can consistently handle unknown words with character-level nodes. In other words, we use word-level nodes to identify known words and character-level nodes to identify unknown words. In the testing phase, we can use a dynamic programming algorithm to search for the most likely path out of all candidate paths.

[2] A machine learning problem exists called structured multi-label classification that allows training from multiple correct paths. However, in this paper we limit our consideration to structured single-label classification, which is simple yet provides great performance.

3 Policies for correct path selection

In this section, we describe our strategies for selecting the correct path $\boldsymbol{y}_t$ in the training phase. As shown in Figure 1, the paths marked in bold can represent the correct annotation of the segmented sentence. Ideally, we need to build a word-character hybrid model that effectively learns the characteristics of unknown words (with character-level nodes) as well as those of known words (with word-level nodes).

We can directly estimate the statistics of known words from an annotated corpus where a sentence is already segmented into words and assigned POS tags. If we select the correct path $\boldsymbol{y}_t$ that corresponds to the annotated sentence, it will only consist of word-level nodes that do not allow learning for unknown words. We therefore need to choose character-level nodes as correct nodes instead of word-level nodes for some words. We expect that those words could reflect unknown words in the future.

Baayen and Sproat (1996) proposed that the characteristics of infrequent words in a training corpus resemble those of unknown words. Their idea has proven effective for estimating the statistics of unknown words in previous studies (Ratnaparkhi, 1996; Nagata, 1999; Nakagawa, 2004).

We adopt Baayen and Sproat's approach as the *baseline policy* in our word-character hybrid model. In the baseline policy, we first count the frequencies of words[3] in the training corpus. We then collect infrequent words that appear less than or equal to r times.[4] If these infrequent words are in the correct path, we use character-level nodes to represent them, and hence the characteristics of unknown words can be learned. For example, in Figure 1 we select the character-level nodes of the word "崇明" (Chongming) as the correct nodes. As a result, the correct path $\boldsymbol{y}_t$ can contain both word-level and character-level nodes (marked with asterisks (*)).

To discover more statistics of unknown words, one might consider just increasing the threshold value r to obtain more artificial unknown words. However, our experimental results indicate that our word-character hybrid model requires an appropriate balance between known and artificial unknown words to achieve optimal performance.

We now describe our new approach to leverage additional examples of unknown words. Intuition suggests that even though the system can handle some unknown words, many *unidentified* unknown words remain that cannot be recovered by the system; we wish to learn the characteristics of such unidentified unknown words. We propose the following simple scheme:

- Divide the training corpus into ten equal sets and perform 10-fold cross validation to find the errors.
- For each trial, train the word-character hybrid model with the baseline policy ($r = 1$) using nine sets and estimate errors using the remaining validation set.
- Collect unidentified unknown words from each validation set.

Several types of errors are produced by the baseline model, but we only focus on those caused by unidentified unknown words, which can be easily collected in the evaluation process. As described later in Section 5.2, we measure the recall on out-of-vocabulary (OOV) words. Here, we define unidentified unknown words as OOV words in each validation set that cannot be recovered by the system. After ten cross validation runs, we get a list of the unidentified unknown words derived from the whole training corpus. Note that the unidentified unknown words in the cross validation are not necessary to be infrequent words, but some overlap may exist. Finally, we obtain the artificial unknown words that combine the unidentified unknown words in cross validation and infrequent words for learning unknown words. We refer to this approach as the *error-driven policy*.

4 Training method

4.1 Discriminative online learning

Let $\mathcal{Y}_t = \{\boldsymbol{y}_t^1, \ldots, \boldsymbol{y}_t^K\}$ be a lattice consisting of candidate paths for a given sentence $\boldsymbol{x}_t$. In the word-character hybrid model, the lattice $\mathcal{Y}_t$ can contain more than 1000 nodes, depending on the length of the sentence $\boldsymbol{x}_t$ and the number of POS tags in the corpus. Therefore, we require a learning algorithm that can efficiently handle large and complex lattice structures.

Online learning is an attractive method for the hybrid model since it quickly converges

[3] We consider a word and its POS tag a single entry.

[4] In our experiments, the optimal threshold value r is selected by evaluating the performance of joint word segmentation and POS tagging on the development set.

Algorithm 1 Generic Online Learning Algorithm

Input: Training set $\mathcal{S} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{T}$
Output: Model weight vector $\mathbf{w}$

1: $\mathbf{w}^{(0)} = \mathbf{0}; \mathbf{v} = \mathbf{0}; i = 0$
2: **for** $iter = 1$ to N **do**
3: **for** $t = 1$ to T **do**
4: $\mathbf{w}^{(i+1)}$ = update $\mathbf{w}^{(i)}$ according to $(\boldsymbol{x}_t, \boldsymbol{y}_t)$
5: $\mathbf{v} = \mathbf{v} + \mathbf{w}^{(i+1)}$
6: $i = i + 1$
7: **end for**
8: **end for**
9: $\mathbf{w} = \mathbf{v}/(N \times T)$

within a few iterations (McDonald, 2006). Algorithm 1 outlines the generic online learning algorithm (McDonald, 2006) used in our framework.

4.2 k-best MIRA

We focus on an online learning algorithm called MIRA (Crammer, 2004), which has the desired accuracy and scalability properties. MIRA combines the advantages of margin-based and perceptron-style learning with an optimization scheme. In particular, we use a generalized version of MIRA (Crammer et al., 2005; McDonald, 2006) that can incorporate k-best decoding in the update procedure. To understand the concept of k-best MIRA, we begin with a linear score function:

$$s(\boldsymbol{x}, \boldsymbol{y}; \mathbf{w}) = \langle \mathbf{w}, \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) \rangle \,, \tag{1}$$

where $\mathbf{w}$ is a weight vector and $\mathbf{f}$ is a feature representation of an input $\boldsymbol{x}$ and an output $\boldsymbol{y}$.

Learning a mapping between an input-output pair corresponds to finding a weight vector $\mathbf{w}$ such that the best scoring path of a given sentence is the same as (or close to) the correct path. Given a training example $(\boldsymbol{x}_t, \boldsymbol{y}_t)$, MIRA tries to establish a margin between the score of the correct path $s(\boldsymbol{x}_t, \boldsymbol{y}_t; \mathbf{w})$ and the score of the best candidate path $s(\boldsymbol{x}_t, \hat{\boldsymbol{y}}; \mathbf{w})$ based on the current weight vector $\mathbf{w}$ that is proportional to a loss function $L(\boldsymbol{y}_t, \hat{\boldsymbol{y}})$.

In each iteration, MIRA updates the weight vector $\mathbf{w}$ by keeping the norm of the change in the weight vector as small as possible. With this framework, we can formulate the optimization problem as follows (McDonald, 2006):

$$\begin{aligned} &\mathbf{w}^{(i+1)} = \text{argmin}_{\mathbf{w}} \|\mathbf{w} - \mathbf{w}^{(i)}\| \\ &\text{s.t. } \forall \hat{\boldsymbol{y}} \in \text{best}_k(\boldsymbol{x}_t; \mathbf{w}^{(i)}) : \\ &s(\boldsymbol{x}_t, \boldsymbol{y}_t; \mathbf{w}) - s(\boldsymbol{x}_t, \hat{\boldsymbol{y}}; \mathbf{w}) \geq L(\boldsymbol{y}_t, \hat{\boldsymbol{y}}) \,, \end{aligned} \tag{2}$$

where $\text{best}_k(\boldsymbol{x}_t; \mathbf{w}^{(i)}) \in \mathcal{Y}_t$ represents a set of top k-best paths given the weight vector $\mathbf{w}^{(i)}$. The above quadratic programming (QP) problem can be solved using Hildreth's algorithm (Yair Censor, 1997). Replacing Eq. (2) into line 4 of Algorithm 1, we obtain k-best MIRA.

The next question is how to efficiently generate $\text{best}_k(\boldsymbol{x}_t; \mathbf{w}^{(i)})$. In this paper, we apply a dynamic programming search (Nagata, 1994) to k-best MIRA. The algorithm has two main search steps: forward and backward. For the forward search, we use Viterbi-style decoding to find the best partial path and its score up to each node in the lattice. For the backward search, we use A^*-style decoding to generate the top k-best paths. A complete path is found when the backward search reaches the beginning node of the lattice, and the algorithm terminates when the number of generated paths equals k.

In summary, we use k-best MIRA to iteratively update $\mathbf{w}^{(i)}$. The final weight vector $\mathbf{w}$ is the average of the weight vectors after each iteration. As reported in (Collins, 2002; McDonald et al., 2005), parameter averaging can effectively avoid overfitting. For inference, we can use Viterbi-style decoding to search for the most likely path $\boldsymbol{y}^*$ for a given sentence $\boldsymbol{x}$ where:

$$\boldsymbol{y}^* = \underset{\boldsymbol{y} \in \mathcal{Y}}{\text{argmax}}\ s(\boldsymbol{x}, \boldsymbol{y}; \mathbf{w}) \,. \tag{3}$$

4.3 Loss function

In conventional sequence labeling where the observation sequence (word) boundaries are fixed, one can use the 0/1 loss to measure the errors of a predicted path with respect to the correct path. However, in our model, word boundaries vary based on the considered path, resulting in a different numbers of output tokens. As a result, we cannot directly use the 0/1 loss.

We instead compute the loss function through false positives (FP) and false negatives (FN). Here, FP means the number of output nodes that are not in the correct path, and FN means the number of nodes in the correct path that cannot be recognized by the system. We define the loss function by:

$$L(\boldsymbol{y}_t, \hat{\boldsymbol{y}}) = FP + FN \,. \tag{4}$$

This loss function can reflect how bad the predicted path $\hat{\boldsymbol{y}}$ is compared to the correct path $\boldsymbol{y}_t$. A weighted loss function based on FP and FN can be found in (Ganchev et al., 2007).

ID	Template	Condition
W0	$\langle w_0 \rangle$	for word-level nodes
W1	$\langle p_0 \rangle$	
W2	$\langle w_0, p_0 \rangle$	
W3	$\langle Length(w_0), p_0 \rangle$	
A0	$\langle A_{\mathrm{S}}(w_0) \rangle$	if w_0 is a single-character word
A1	$\langle A_{\mathrm{S}}(w_0), p_0 \rangle$	
A2	$\langle A_{\mathrm{B}}(w_0) \rangle$	for word-level nodes
A3	$\langle A_{\mathrm{B}}(w_0), p_0 \rangle$	
A4	$\langle A_{\mathrm{E}}(w_0) \rangle$	
A5	$\langle A_{\mathrm{E}}(w_0), p_0 \rangle$	
A6	$\langle A_{\mathrm{B}}(w_0), A_{\mathrm{E}}(w_0) \rangle$	
A7	$\langle A_{\mathrm{B}}(w_0), A_{\mathrm{E}}(w_0), p_0 \rangle$	
T0	$\langle T_{\mathrm{S}}(w_0) \rangle$	if w_0 is a single-character word
T1	$\langle T_{\mathrm{S}}(w_0), p_0 \rangle$	
T2	$\langle T_{\mathrm{B}}(w_0) \rangle$	for word-level nodes
T3	$\langle T_{\mathrm{B}}(w_0), p_0 \rangle$	
T4	$\langle T_{\mathrm{E}}(w_0) \rangle$	
T5	$\langle T_{\mathrm{E}}(w_0), p_0 \rangle$	
T6	$\langle T_{\mathrm{B}}(w_0), T_{\mathrm{E}}(w_0) \rangle$	
T7	$\langle T_{\mathrm{B}}(w_0), T_{\mathrm{E}}(w_0), p_0 \rangle$	
C0	$\langle c_j \rangle, j \in [-2, 2] \times p_0$	for character-level nodes
C1	$\langle c_j, c_{j+1} \rangle, j \in [-2, 1] \times p_0$	
C2	$\langle c_{-1}, c_1 \rangle \times p_0$	
C3	$\langle T(c_j) \rangle, j \in [-2, 2] \times p_0$	
C4	$\langle T(c_j), T(c_{j+1}) \rangle, j \in [-2, 1] \times p_0$	
C5	$\langle T(c_{-1}), T(c_1) \rangle \times p_0$	
C6	$\langle c_0, T(c_0) \rangle \times p_0$	

Table 2: Unigram features.

ID	Template	Condition
B0	$\langle w_{-1}, w_0 \rangle$	if w_{-1} and w_0 are word-level nodes
B1	$\langle p_{-1}, p_0 \rangle$	
B2	$\langle w_{-1}, p_0 \rangle$	
B3	$\langle p_{-1}, w_0 \rangle$	
B4	$\langle w_{-1}, w_0, p_0 \rangle$	
B5	$\langle p_{-1}, w_0, p_0 \rangle$	
B6	$\langle w_{-1}, p_{-1}, w_0 \rangle$	
B7	$\langle w_{-1}, p_{-1}, p_0 \rangle$	
B8	$\langle w_{-1}, p_{-1}, w_0, p_0 \rangle$	
B9	$\langle Length(w_{-1}), p_0 \rangle$	
TB0	$\langle T_{\mathrm{E}}(w_{-1}) \rangle$	
TB1	$\langle T_{\mathrm{E}}(w_{-1}), p_0 \rangle$	
TB2	$\langle T_{\mathrm{E}}(w_{-1}), p_{-1}, p_0 \rangle$	
TB3	$\langle T_{\mathrm{E}}(w_{-1}), T_{\mathrm{B}}(w_0) \rangle$	
TB4	$\langle T_{\mathrm{E}}(w_{-1}), T_{\mathrm{B}}(w_0), p_0 \rangle$	
TB5	$\langle T_{\mathrm{E}}(w_{-1}), p_{-1}, T_{\mathrm{B}}(w_0) \rangle$	
TB6	$\langle T_{\mathrm{E}}(w_{-1}), p_{-1}, T_{\mathrm{B}}(w_0), p_0 \rangle$	
CB0	$\langle p_{-1}, p_0 \rangle$	otherwise

Table 3: Bigram features.

Character type	Description
Space	Space
Numeral	Arabic and Chinese numerals
Symbol	Symbols
Alphabet	Alphabets
Chinese	Chinese characters
Other	Others

Table 4: Character types.

4.4 Features

This section discusses the structure of $\mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$. We broadly classify features into two categories: unigram and bigram features. We design our feature templates to capture various levels of information about words and POS tags. Let us introduce some notation. We write w_{-1} and w_0 for the surface forms of words, where subscripts -1 and 0 indicate the previous and current positions, respectively. POS tags p_{-1} and p_0 can be interpreted in the same way. We denote the characters by c_j.

Unigram features: Table 2 shows our unigram features. Templates W0–W3 are basic word-level unigram features, where $Length(w_0)$ denotes the length of the word w_0. Using just the surface forms can overfit the training data and lead to poor predictions on the test data. To alleviate this problem, we use two generalized features of the surface forms. The first is the beginning and end characters of the surface (A0–A7). For example, $\langle A_{\mathrm{B}}(w_0) \rangle$ denotes the beginning character of the current word w_0, and $\langle A_{\mathrm{B}}(w_0), A_{\mathrm{E}}(w_0) \rangle$ denotes the beginning and end characters in the word. The second is the types of beginning and end characters of the surface (T0–T7). We define a set of general character types, as shown in Table 4.

Templates C0–C6 are basic character-level unigram features taken from (Nakagawa, 2004). These templates operate over a window of ± 2 characters. The features include characters (C0), pairs of characters (C1–C2), character types (C3), and pairs of character types (C4–C5). In addition, we add pairs of characters and character types (C6).

Bigram features: Table 3 shows our bigram features. Templates B0-B9 are basic word-level bigram features. These features aim to capture all the possible combinations of word and POS bigrams. Templates TB0-TB6 are the types of characters for bigrams. For example, $\langle T_{\mathrm{E}}(w_{-1}), T_{\mathrm{B}}(w_0) \rangle$ captures the change of character types from the end character in the previous word to the beginning character in the current word.

Note that if one of the adjacent nodes is a character-level node, we use the template CB0 that represents POS bigrams. In our preliminary experiments, we found that if we add more features to non-word-level bigrams, the number of features grows rapidly due to the dense connections between non-word-level nodes. However, these features only slightly improve performance over using simple POS bigrams.

(a) Experiments on small training corpus

Data set	CTB chap. IDs	# of sent.	# of words
Training	1-270	3,046	75,169
Development	301-325	350	6,821
Test	271-300	348	8,008
# of POS tags			32
OOV (word)			0.0987 (790/8,008)
OOV (word & POS)			0.1140 (913/8,008)

(b) Experiments on large training corpus

Data set	CTB chap. IDs	# of sent.	# of words
Training	1-270, 400-931, 1001-1151	18,089	493,939
Development	301-325	350	6,821
Test	271-300	348	8,008
# of POS tags			35
OOV (word)			0.0347 (278/8,008)
OOV (word & POS)			0.0420 (336/8,008)

Table 5: Training, development, and test data statistics on CTB 5.0 used in our experiments.

5 Experiments

5.1 Data sets

Previous studies on joint Chinese word segmentation and POS tagging have used Penn Chinese Treebank (CTB) (Xia et al., 2000) in experiments. However, versions of CTB and experimental settings vary across different studies.

In this paper, we used CTB 5.0 (LDC2005T01) as our main corpus, defined the training, development and test sets according to (Jiang et al., 2008a; Jiang et al., 2008b), and designed our experiments to explore the impact of the training corpus size on our approach. Table 5 provides the statistics of our experimental settings on the small and large training data. The out-of-vocabulary (OOV) is defined as tokens in the test set that are not in the training set (Sproat and Emerson, 2003). Note that the development set was only used for evaluating the trained model to obtain the optimal values of tunable parameters.

5.2 Evaluation

We evaluated both word segmentation (Seg) and joint word segmentation and POS tagging (Seg & Tag). We used recall (R), precision (P), and F_1 as evaluation metrics. Following (Sproat and Emerson, 2003), we also measured the recall on OOV (R_{OOV}) tokens and in-vocabulary (R_{IV}) tokens. These performance measures can be calculated as follows:

$$Recall\ (R) = \frac{\text{\# of correct tokens}}{\text{\# of tokens in test data}}$$

$$Precision\ (P) = \frac{\text{\# of correct tokens}}{\text{\# of tokens in system output}}$$

$$F_1 = \frac{2 \cdot R \cdot P}{R + P}$$

$$R_{\text{OOV}} = \frac{\text{\# of correct OOV tokens}}{\text{\# of OOV tokens in test data}}$$

$$R_{\text{IV}} = \frac{\text{\# of correct IV tokens}}{\text{\# of IV tokens in test data}}$$

For Seg, a token is considered to be a correct one if the word boundary is correctly identified. For Seg & Tag, both the word boundary and its POS tag have to be correctly identified to be counted as a correct token.

5.3 Parameter estimation

Our model has three tunable parameters: the number of training iterations N; the number of top k-best paths; and the threshold r for infrequent words. Since we were interested in finding an optimal combination of word-level and character-level nodes for training, we focused on tuning r. We fixed N = 10 and k = 5 for all experiments. For the baseline policy, we varied r in the range of $[1, 5]$ and found that setting $r = 3$ yielded the best performance on the development set for both the small and large training corpus experiments. For the error-driven policy, we collected unidentified unknown words using 10-fold cross validation on the training set, as previously described in Section 3.

5.4 Impact of policies for correct path selection

Table 6 shows the results of our word-character hybrid model using the error-driven and baseline policies. The third and fourth columns indicate the numbers of known and artificial unknown words in the training phase. The total number of words is the same, but the different policies yield different balances between the known and artificial unknown words for learning the hybrid model. Optimal balances were selected using the development set. The error-driven policy provides additional artificial unknown words in the training set. The error-driven policy can improve R_{OOV} as well as maintain good R_{IV}, resulting in overall F_1 improvements.

(a) Experiments on small training corpus

Eval type	Policy	# of words in training (75,169) kwn.	art. unk.	R	P	F_1	R_{OOV}	R_{IV}
Seg	error-driven	63,997	11,172	0.9587	0.9509	0.9548	0.7557	0.9809
	baseline	64,999	10,170	0.9572	0.9489	0.9530	0.7304	0.9820
Seg & Tag	error-driven	63,997	11,172	0.8929	0.8857	0.8892	0.5444	0.9377
	baseline	64,999	10,170	0.8897	0.8820	0.8859	0.5246	0.9367

(b) Experiments on large training corpus

Eval Type	Policy	# of words in training (493,939) kwn.	art. unk.	R	P	F_1	R_{OOV}	R_{IV}
Seg	error-driven	442,423	51,516	0.9829	0.9746	0.9787	0.7698	0.9906
	baseline	449,679	44,260	0.9821	0.9736	0.9779	0.7590	0.9902
Seg & Tag	error-driven	442,423	51,516	0.9407	0.9328	0.9367	0.5982	0.9557
	baseline	449,679	44,260	0.9401	0.9319	0.9360	0.5952	0.9552

Table 6: Results of our word-character hybrid model using error-driven and baseline policies.

Method	Seg	Seg & Tag
Ours (error-driven)	**0.9787**	**0.9367**
Ours (baseline)	0.9779	0.9360
Jiang08a	0.9785	0.9341
Jiang08b	0.9774	0.9337
N&U07	0.9783	0.9332

Table 7: Comparison of F_1 results with previous studies on CTB 5.0.

Trial	Seg N&U07	Seg Z&C08	Seg **Ours** (base.)	Seg & Tag N&U07	Seg & Tag Z&C08	Seg & Tag **Ours** (base.)
1	0.9701	0.9721	0.9732	0.9262	0.9346	0.9358
2	0.9738	0.9762	0.9752	0.9318	0.9385	0.9380
3	0.9571	0.9594	0.9578	0.9023	0.9086	0.9067
4	0.9629	0.9592	0.9655	0.9132	0.9160	0.9223
5	0.9597	0.9606	0.9617	0.9132	0.9172	0.9187
6	0.9473	0.9456	0.9460	0.8823	0.8883	0.8885
7	0.9528	0.9500	0.9562	0.9003	0.9051	0.9076
8	0.9519	0.9512	0.9528	0.9002	0.9030	0.9062
9	0.9566	0.9479	0.9575	0.8996	0.9033	0.9052
10	0.9631	0.9645	0.9659	0.9154	0.9196	0.9225
Avg.	0.9595	0.9590	**0.9611**	0.9085	0.9134	**0.9152**

Table 8: Comparison of F_1 results of our baseline model with Nakagawa and Uchimoto (2007) and Zhang and Clark (2008) on CTB 3.0.

Method	Seg	Seg & Tag
Ours (baseline)	**0.9611**	**0.9152**
Z&C08	0.9590	0.9134
N&U07	0.9595	0.9085
N&L04	0.9520	-

Table 9: Comparison of averaged F_1 results (by 10-fold cross validation) with previous studies on CTB 3.0.

5.5 Comparison with best prior approaches

In this section, we attempt to make meaningful comparison with the best prior approaches reported in the literature. Although most previous studies used CTB, their versions of CTB and experimental settings are different, which complicates comparison.

Ng and Low (2004) (**N&L04**) used CTB 3.0. However, they just showed POS tagging results on a per character basis, not on a per word basis. Zhang and Clark (2008) (**Z&C08**) generated CTB 3.0 from CTB 4.0. Jiang et al. (2008a; 2008b) (**Jiang08a**, **Jiang08b**) used CTB 5.0. Shi and Wang (2007) used CTB that was distributed in the SIGHAN Bakeoff. Besides CTB, they also used HowNet (Dong and Dong, 2006) to obtain semantic class features. Zhang and Clark (2008) indicated that their results cannot directly compare to the results of Shi and Wang (2007) due to different experimental settings.

We decided to follow the experimental settings of Jiang et al. (2008a; 2008b) on CTB 5.0 and Zhang and Clark (2008) on CTB 4.0 since they reported the best performances on joint word segmentation and POS tagging using the training materials only derived from the corpora. The performance scores of previous studies are directly taken from their papers. We also conducted experiments using the system implemented by Nakagawa and Uchimoto (2007) (**N&U07**) for comparison.

Our experiment on the large training corpus is identical to that of Jiang et al. (Jiang et al., 2008a; Jiang et al., 2008b). Table 7 compares the F_1 results with previous studies on CTB 5.0. The result of our error-driven model is superior to previous reported results for both Seg and Seg & Tag, and the result of our baseline model compares favorably to the others.

Following Zhang and Clark (2008), we first generated CTB 3.0 from CTB 4.0 using sentence IDs 1–10364. We then divided CTB 3.0 into ten equal sets and conducted 10-fold cross vali-

dation. Unfortunately, Zhang and Clark's experimental setting did not allow us to use our error-driven policy since performing 10-fold cross validation again on each main cross validation trial is computationally too expensive. Therefore, we used our baseline policy in this setting and fixed $r = 3$ for all cross validation runs. Table 8 compares the F_1 results of our baseline model with Nakagawa and Uchimoto (2007) and Zhang and Clark (2008) on CTB 3.0. Table 9 shows a summary of averaged F_1 results on CTB 3.0. Our baseline model outperforms all prior approaches for both Seg and Seg & Tag, and we hope that our error-driven model can further improve performance.

6 Related work

In this section, we discuss related approaches based on several aspects of learning algorithms and search space representation methods. Maximum entropy models are widely used for word segmentation and POS tagging tasks (Uchimoto et al., 2001; Ng and Low, 2004; Nakagawa, 2004; Nakagawa and Uchimoto, 2007) since they only need moderate training times while they provide reasonable performance. Conditional random fields (CRFs) (Lafferty et al., 2001) further improve the performance (Kudo et al., 2004; Shi and Wang, 2007) by performing whole-sequence normalization to avoid label-bias and length-bias problems. However, CRF-based algorithms typically require longer training times, and we observed an infeasible convergence time for our hybrid model.

Online learning has recently gained popularity for many NLP tasks since it performs comparably or better than batch learning using shorter training times (McDonald, 2006). For example, a perceptron algorithm is used for joint Chinese word segmentation and POS tagging (Zhang and Clark, 2008; Jiang et al., 2008a; Jiang et al., 2008b). Another potential algorithm is MIRA, which integrates the notion of the large-margin classifier (Crammer, 2004). In this paper, we first introduce MIRA to joint word segmentation and POS tagging and show very encouraging results. With regard to error-driven learning, Brill (1995) proposed a transformation-based approach that acquires a set of error-correcting rules by comparing the outputs of an initial tagger with the correct annotations on a training corpus. Our approach does not learn the error-correcting rules. We only aim to capture the characteristics of unknown words and augment their representatives.

As for search space representation, Ng and Low (2004) found that for Chinese, the character-based model yields better results than the word-based model. Nakagawa and Uchimoto (2007) provided empirical evidence that the character-based model is not always better than the word-based model. They proposed a hybrid approach that exploits both the word-based and character-based models. Our approach overcomes the limitation of the original hybrid model by a discriminative online learning algorithm for training.

7 Conclusion

In this paper, we presented a discriminative word-character hybrid model for joint Chinese word segmentation and POS tagging. Our approach has two important advantages. The first is robust search space representation based on a hybrid model in which word-level and character-level nodes are used to identify known and unknown words, respectively. We introduced a simple scheme based on the error-driven concept to effectively learn the characteristics of known and unknown words from the training corpus. The second is a discriminative online learning algorithm based on MIRA that enables us to incorporate arbitrary features to our hybrid model. Based on extensive comparisons, we showed that our approach is superior to the existing approaches reported in the literature. In future work, we plan to apply our framework to other Asian languages, including Thai and Japanese.

Acknowledgments

We would like to thank Tetsuji Nakagawa for his helpful suggestions about the word-character hybrid model, Chen Wenliang for his technical assistance with the Chinese processing, and the anonymous reviewers for their insightful comments.

References

Masayuki Asahara. 2003. *Corpus-based Japanese morphological analysis*. Nara Institute of Science and Technology, Doctor's Thesis.

Harald Baayen and Richard Sproat. 1996. Estimating lexical priors for low-frequency morphologically ambiguous forms. *Computational Linguistics*, 22(2):155–166.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

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

Koby Crammer, Ryan McDonald, and Fernando Pereira. 2005. Scalable large-margin online learning for structured classification. In *NIPS Workshop on Learning With Structured Outputs*.

Koby Crammer. 2004. *Online Learning of Complex Categorial Problems*. Hebrew Univeristy of Jerusalem, PhD Thesis.

Zhendong Dong and Qiang Dong. 2006. *Hownet and the Computation of Meaning*. World Scientific.

Kuzman Ganchev, Koby Crammer, Fernando Pereira, Gideon Mann, Kedar Bellare, Andrew McCallum, Steven Carroll, Yang Jin, and Peter White. 2007. Penn/umass/chop biocreative ii systems. In *Proceedings of the Second BioCreative Challenge Evaluation Workshop*.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008a. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL*.

Wenbin Jiang, Haitao Mi, and Qun Liu. 2008b. Word lattice reranking for chinese word segmentation and part-of-speech tagging. In *Proceedings of COLING*.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to japanese morphological analysis. In *Proceedings of EMNLP*, pages 230–237.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

Ryan McDonald, Femando Pereira, Kiril Ribarow, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP*, pages 523–530.

Ryan McDonald. 2006. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. University of Pennsylvania, PhD Thesis.

Masaki Nagata. 1994. A stochastic japanese morphological analyzer using a forward-DP backward-A* n-best search algorithm. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 201–207.

Masaki Nagata. 1999. A part of speech estimation method for japanese unknown words using a statistical model of morphology and context. In *Proceedings of ACL*, pages 277–284.

Tetsuji Nakagawa and Kiyotaka Uchimoto. 2007. A hybrid approach to word segmentation and pos tagging. In *Proceedings of ACL Demo and Poster Sessions*.

Tetsuji Nakagawa. 2004. Chinese and japanese word segmentation using word-level and character-level information. In *Proceedings of COLING*, pages 466–472.

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In *Proceedings of EMNLP*, pages 277–284.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP*, pages 133–142.

Yanxin Shi and Mengqiu Wang. 2007. A dual-layer crfs based joint decoding method for cascaded segmentation and labeling tasks. In *Proceedings of IJCAI*.

Richard Sproat and Thomas Emerson. 2003. The first international chinese word segmentation bakeoff. In *Proceedings of the 2nd SIGHAN Workshop on Chinese Language Processing*, pages 133–143.

Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. 2001. The unknown word problem: a morphological analysis of japanese using maximum entropy aided by a dictionary. In *Proceedings of EMNLP*, pages 91–99.

Fei Xia, Martha Palmer, Nianwen Xue, Mary Ellen Okurowski, John Kovarik, Fu dong Chiou, and Shizhe Huang. 2000. Developing guidelines and ensuring consistency for chinese text annotation. In *Proceedings of LREC*.

Stavros A. Zenios Yair Censor. 1997. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press.

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

Automatic Adaptation of Annotation Standards: Chinese Word Segmentation and POS Tagging – A Case Study

Wenbin Jiang † **Liang Huang** ‡ **Qun Liu** †

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

‡Google Research
1350 Charleston Rd.
Mountain View, CA 94043, USA
lianghuang@google.com
liang.huang.sh@gmail.com

Abstract

Manually annotated corpora are valuable but scarce resources, yet for many annotation tasks such as treebanking and sequence labeling there exist multiple corpora with *different* and *incompatible* annotation guidelines or standards. This seems to be a great waste of human efforts, and it would be nice to automatically adapt one annotation standard to another. We present a simple yet effective strategy that transfers knowledge from a differently annotated corpus to the corpus with desired annotation. We test the efficacy of this method in the context of Chinese word segmentation and part-of-speech tagging, where no segmentation and POS tagging standards are widely accepted due to the lack of morphology in Chinese. Experiments show that adaptation from the much larger People's Daily corpus to the smaller but more popular Penn Chinese Treebank results in significant improvements in both segmentation and tagging accuracies (with error reductions of 30.2% and 14%, respectively), which in turn helps improve Chinese parsing accuracy.

1 Introduction

Much of statistical NLP research relies on some sort of manually annotated corpora to train their models, but these resources are extremely expensive to build, especially at a large scale, for example in treebanking (Marcus et al., 1993). However the linguistic theories underlying these annotation efforts are often heavily debated, and as a result there often exist multiple corpora for the same task with vastly different and incompatible annotation philosophies. For example just for English treebanking there have been the Chomskian-style Penn Treebank (Marcus et al., 1993) the HPSG LinGo Redwoods Treebank (Oepen et al., 2002), and a smaller dependency treebank (Buchholz and Marsi, 2006). A second, related problem is that the raw texts are also drawn from different domains, which for the above example range from financial news (PTB/WSJ) to transcribed dialog (LinGo). These two problems seem be a great waste in human efforts, and it would be nice if one could automatically adapt from one annotation standard and/or domain to another in order to exploit much larger datasets for better training. The second problem, domain adaptation, is very well-studied, e.g. by Blitzer et al. (2006) and Daumé III (2007) (and see below for discussions), so in this paper we focus on the less studied, but equally important problem of *annotation-style adaptation*.

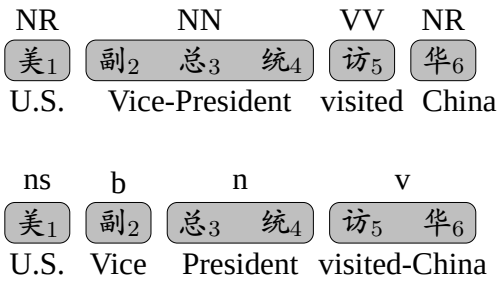


Figure 1: Incompatible word segmentation and POS tagging standards between CTB (upper) and People's Daily (below).

We present a very simple yet effective strategy that enables us to utilize knowledge from a differently annotated corpora for the training of a model on a corpus with desired annotation. The basic idea is very simple: we first train on a source corpus, resulting in a source classifier, which is used to label the target corpus and results in a "source-style" annotation of the target corpus. We then

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 522–530,
Suntec, Singapore, 2-7 August 2009.

train a second model on the target corpus with the first classifier's prediction as additional features for guided learning.

This method is very similar to some ideas in domain adaptation (Daumé III and Marcu, 2006; Daumé III, 2007), but we argue that the underlying problems are quite different. Domain adaptation assumes the labeling guidelines are preserved between the two domains, e.g., an adjective is always labeled as JJ regardless of from Wall Street Journal (WSJ) or Biomedical texts, and only the *distributions* are different, e.g., the word "control" is most likely a verb in WSJ but often a noun in Biomedical texts (as in "control experiment"). Annotation-style adaptation, however, tackles the problem where the guideline itself is changed, for example, one treebank might distinguish between transitive and intransitive verbs, while merging the different noun types (NN, NNS, etc.), and for example one treebank (PTB) might be much flatter than the other (LinGo), not to mention the fundamental disparities between their underlying linguistic representations (CFG vs. HPSG). In this sense, the problem we study in this paper seems much harder and more motivated from a linguistic (rather than statistical) point of view. More interestingly, our method, without any assumption on the distributions, can be simultaneously applied to both domain *and* annotation standards adaptation problems, which is very appealing in practice because the latter problem often implies the former, as in our case study.

To test the efficacy of our method we choose Chinese word segmentation and part-of-speech tagging, where the problem of incompatible annotation standards is one of the most evident: so far no segmentation standard is widely accepted due to the lack of a clear definition of Chinese *words*, and the (almost complete) lack of morphology results in much bigger ambiguities and heavy debates in tagging philosophies for Chinese parts-of-speech. The two corpora used in this study are the much larger People's Daily (PD) (5.86M words) corpus (Yu et al., 2001) and the smaller but more popular Penn Chinese Treebank (CTB) (0.47M words) (Xue et al., 2005). They used very different segmentation standards as well as different POS tagsets and tagging guidelines. For example, in Figure 1, People's Daily breaks "Vice-President" into two words while combines the phrase "visited-China" as a compound. Also CTB has four verbal categories (VV for normal verbs, and VC for copulas, etc.) while PD has only one verbal tag (v) (Xia, 2000). It is preferable to transfer knowledge from PD to CTB because the latter also annotates tree structures which is very useful for downstream applications like parsing, summarization, and machine translation, yet it is much smaller in size. Indeed, many recent efforts on Chinese-English translation and Chinese parsing use the CTB as the *de facto* segmentation and tagging standards, but suffers from the limited size of training data (Chiang, 2007; Bikel and Chiang, 2000). We believe this is also a reason why state-of-the-art accuracy for Chinese parsing is much lower than that of English (CTB is only half the size of PTB).

Our experiments show that adaptation from PD to CTB results in a significant improvement in segmentation and POS tagging, with error reductions of 30.2% and 14%, respectively. In addition, the improved accuracies from segmentation and tagging also lead to an improved parsing accuracy on CTB, reducing 38% of the error propagation from word segmentation to parsing. We envision this technique to be general and widely applicable to many other sequence labeling tasks.

In the rest of the paper we first briefly review the popular classification-based method for word segmentation and tagging (Section 2), and then describe our idea of annotation adaptation (Section 3). We then discuss other relevant previous work including co-training and classifier combination (Section 4) before presenting our experimental results (Section 5).

2 Segmentation and Tagging as Character Classification

Before describing the adaptation algorithm, we give a brief introduction of the baseline character classification strategy for segmentation, as well as joint segmenation and tagging (henceforth "Joint S&T"). following our previous work (Jiang et al., 2008). Given a Chinese sentence as sequence of n characters:

$$C_1\ C_2\ ..\ C_n$$

where C_i is a character, word segmentation aims to split the sequence into $m(\leq n)$ words:

$$C_{1:e_1}\ C_{e_1+1:e_2}\ ..\ C_{e_{m-1}+1:e_m}$$

where each subsequence $C_{i:j}$ indicates a Chinese word spanning from characters C_i to C_j (both in-

Algorithm 1 Perceptron training algorithm.

Input: Training examples (x_i, y_i)
$\vec{\alpha} \leftarrow \mathbf{0}$
for $t \leftarrow 1$ .. T **do**
  for $i \leftarrow 1$ .. N **do**
    $z_i \leftarrow \text{argmax}_{z \in \mathbf{GEN}(x_i)} \mathbf{\Phi}(x_i, z) \cdot \vec{\alpha}$
    if $z_i \neq y_i$ **then**
      $\vec{\alpha} \leftarrow \vec{\alpha} + \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i)$
Output: Parameters $\vec{\alpha}$

clusive). While in Joint S&T, each word is further annotated with a POS tag:

$$C_{1:e_1}/t_1 \; C_{e_1+1:e_2}/t_2 \; .. \; C_{e_{m-1}+1:e_m}/t_m$$

where $t_k(k = 1..m)$ denotes the POS tag for the word $C_{e_{k-1}+1:e_k}$.

2.1 Character Classification Method

Xue and Shen (2003) describe for the first time the character classification approach for Chinese word segmentation, where each character is given a boundary tag denoting its relative position in a word. In Ng and Low (2004), Joint S&T can also be treated as a character classification problem, where a boundary tag is combined with a POS tag in order to give the POS information of the word containing these characters. In addition, Ng and Low (2004) find that, compared with POS tagging after word segmentation, Joint S&T can achieve higher accuracy on both segmentation and POS tagging. This paper adopts the tag representation of Ng and Low (2004). For word segmentation only, there are four boundary tags:

- *b*: the begin of the word
- *m*: the middle of the word
- *e*: the end of the word
- *s*: a single-character word

while for Joint S&T, a POS tag is attached to the tail of a boundary tag, to incorporate the word boundary information and POS information together. For example, *b-NN* indicates that the character is the begin of a noun. After all characters of a sentence are assigned boundary tags (or with POS postfix) by a classifier, the corresponding word sequence (or with POS) can be directly derived. Take segmentation for example, a character assigned a tag *s* or a subsequence of words assigned a tag sequence bm^*e indicates a word.

2.2 Training Algorithm and Features

Now we will show the training algorithm of the classifier and the features used. Several classification models can be adopted here, however, we choose the averaged perceptron algorithm (Collins, 2002) because of its simplicity and high accuracy. It is an online training algorithm and has been successfully used in many NLP tasks, such as POS tagging (Collins, 2002), parsing (Collins and Roark, 2004), Chinese word segmentation (Zhang and Clark, 2007; Jiang et al., 2008), and so on.

Similar to the situation in other sequence labeling problems, the training procedure is to learn a discriminative model mapping from inputs $x \in X$ to outputs $y \in Y$, where X is the set of sentences in the training corpus and Y is the set of corresponding labelled results. Following Collins, we use a function $\mathbf{GEN}(x)$ enumerating the candidate results of an input x , a representation $\mathbf{\Phi}$ mapping each training example $(x, y) \in X \times Y$ to a feature vector $\mathbf{\Phi}(x, y) \in R^d$, and a parameter vector $\vec{\alpha} \in R^d$ corresponding to the feature vector. For an input character sequence x, we aim to find an output $F(x)$ that satisfies:

$$F(x) = \underset{y \in \mathbf{GEN}(x)}{\text{argmax}} \; \mathbf{\Phi}(x, y) \cdot \vec{\alpha} \qquad (1)$$

where $\mathbf{\Phi}(x, y) \cdot \vec{\alpha}$ denotes the inner product of feature vector $\mathbf{\Phi}(x, y)$ and the parameter vector $\vec{\alpha}$.

Algorithm 1 depicts the pseudo code to tune the parameter vector $\vec{\alpha}$. In addition, the "averaged parameters" technology (Collins, 2002) is used to alleviate overfitting and achieve stable performance. Table 1 lists the feature template and corresponding instances. Following Ng and Low (2004), the current considering character is denoted as C_0, while the ith character to the left of C_0 as C_{-i}, and to the right as C_i. There are additional two functions of which each returns some property of a character. $Pu(\cdot)$ is a boolean function that checks whether a character is a punctuation symbol (returns 1 for a punctuation, 0 for not). $T(\cdot)$ is a multi-valued function, it classifies a character into four classifications: *number*, *date*, *English letter* and *others* (returns 1, 2, 3 and 4, respectively).

3 Automatic Annotation Adaptation

From this section, several shortened forms are adopted for representation inconvenience. We use *source corpus* to denote the corpus with the annotation standard that we don't require, which is of

Feature Template	Instances
C_i $(i = -2..2)$	C_{-2} = 九, C_{-1} = ○, C_0 = 年, C_1 = 代, C_2 = R
C_iC_{i+1} $(i = -2..1)$	$C_{-2}C_{-1}$ = 九○, $C_{-1}C_0$ = ○年, C_0C_1 = 年代, C_1C_2 = 代R
$C_{-1}C_1$	$C_{-1}C_1$ = ○代
$Pu(C_0)$	$Pu(C_0)$ = 0
$T(C_{-2})T(C_{-1})T(C_0)T(C_1)T(C_2)$	$T(C_{-2})T(C_{-1})T(C_0)T(C_1)T(C_2)$ = 11243

Table 1: Feature templates and instances from Ng and Low (Ng and Low, 2004). Suppose we are considering the third character "年" in "九○ 年 代R".

course the source of the adaptation, while *target corpus* denoting the corpus with the desired standard. And correspondingly, the two annotation standards are naturally denoted as *source standard* and *target standard*, while the classifiers following the two annotation standards are respectively named as *source classifier* and *target classifier*, if needed.

Considering that word segmentation and Joint S&T can be conducted in the same character classification manner, we can design an unified standard adaptation framework for the two tasks, by taking the source classifier's classification result as the guide information for the target classifier's classification decision. The following section depicts this adaptation strategy in detail.

3.1 General Adaptation Strategy

In detail, in order to adapt knowledge from the source corpus, first, a source classifier is trained on it and therefore captures the knowledge it contains; then, the source classifier is used to classify the characters in the target corpus, although the classification result follows a standard that we don't desire; finally, a target classifier is trained on the target corpus, with the source classifier's classification result as additional guide information. The training procedure of the target classifier automatically learns the regularity to transfer the source classifier's predication result from source standard to target standard. This regularity is incorporated together with the knowledge learnt from the target corpus itself, so as to obtain enhanced predication accuracy. For a given un-classified character sequence, the decoding is analogous to the training. First, the character sequence is input into the source classifier to obtain an source standard annotated classification result, then it is input into the target classifier with this classification result as additional information to get the final result. This coincides with the stacking method for combining dependency parsers (Martins et al., 2008; Nivre and McDonald, 2008), and is also similar to the Pred baseline for domain adaptation in (Daumé III and Marcu, 2006; Daumé III, 2007). Figures 2 and 3 show the flow charts for training and decoding.

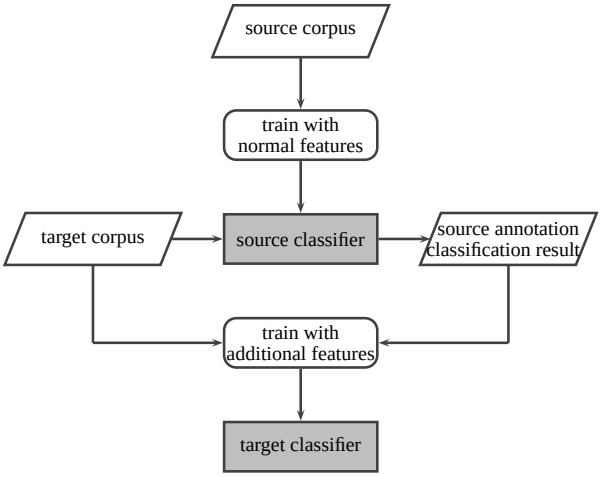


Figure 2: The pipeline for training.

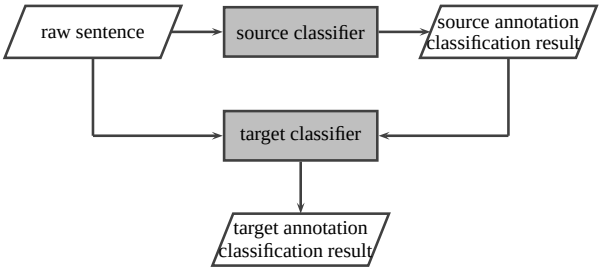


Figure 3: The pipeline for decoding.

The utilization of the source classifier's classification result as additional guide information resorts to the introduction of new features. For the current considering character waiting for classification, the most intuitive guide features is the source classifier's classification result itself. However, our effort isn't limited to this, and more special features are introduced: the source classifier's classification result is attached to every feature listed in Table 1 to get combined guide features. This is similar to feature design in discriminative dependency parsing (McDonald et al., 2005; Mc-

Donald and Pereira, 2006), where the basic features, composed of words and POSs in the context, are also conjoined with link direction and distance in order to obtain more special features. Table 2 shows an example of guide features and basic features, where "α = b " represents that the source classifier classifies the current character as *b*, the beginning of a word.

Such combination method derives a series of specific features, which helps the target classifier to make more precise classifications. The parameter tuning procedure of the target classifier will automatically learn the regularity of using the source classifier's classification result to guide its decision making. For example, if a current considering character shares some basic features in Table 2 and it is classified as *b*, then the target classifier will probably classify it as *m*. In addition, the training procedure of the target classifier also learns the relative weights between the guide features and the basic features, so that the knowledge from both the source corpus and the target corpus are automatically integrated together.

In fact, more complicated features can be adopted as guide information. For error tolerance, guide features can be extracted from n-best results or compacted lattices of the source classifier; while for the best use of the source classifier's output, guide features can also be the classification results of several successive characters. We leave them as future research.

Baseline Features
C_{-2} = 美
C_{-1} = 副
C_0 = 总
C_1 = 统
C_2 = 访
$C_{-2}C_{-1}$ = 美副
$C_{-1}C_0$ = 副总
C_0C_1 = 总统
C_1C_2 = 统访
$C_{-1}C_1$ = 副统
$Pu(C_0)$ = 0
$T(C_{-2})T(C_{-1})T(C_0)T(C_1)T(C_2)$ = 44444
Guide Features
α = b
C_{-2} = 美 $\circ$ α = b
C_{-1} = 副 $\circ$ α = b
C_0 = 总 $\circ$ α = b
C_1 = 统 $\circ$ α = b
C_2 = 访 $\circ$ α = b
$C_{-2}C_{-1}$ = 美副 $\circ$ α = b
$C_{-1}C_0$ = 副总 $\circ$ α = b
C_0C_1 = 总统 $\circ$ α = b
C_1C_2 = 统访 $\circ$ α = b
$C_{-1}C_1$ = 副统 $\circ$ α = b
$Pu(C_0)$ = 0 $\circ$ α = b
$T(C_{-2})T(C_{-1})T(C_0)T(C_1)T(C_2)$ = 44444 $\circ$ α = b

Table 2: An example of basic features and guide features of standard-adaptation for word segmentation. Suppose we are considering the third character "总" in "美副 总 统访华".

4 Related Works

Co-training (Sarkar, 2001) and classifier combination (Nivre and McDonald, 2008) are two technologies for training improved dependency parsers. The co-training technology lets two different parsing models learn from each other during parsing an unlabelled corpus: one model selects some unlabelled sentences it can confidently parse, and provide them to the other model as additional training corpus in order to train more powerful parsers. The classifier combination lets graph-based and transition-based dependency parsers to utilize the features extracted from each other's parsing results, to obtain combined, enhanced parsers. The two technologies aim to let two models learn from each other on the same corpora with the same distribution and annotation standard, while our strategy aims to integrate the knowledge in multiple corpora with different annotation-styles.

Gao et al. (2004) described a transformation-based converter to transfer a certain annotation-style word segmentation result to another style. They design some class-type transformation templates and use the transformation-based error-driven learning method of Brill (1995) to learn what word delimiters should be modified. However, this converter need human designed transformation templates, and is hard to be generalized to POS tagging, not to mention other structure labeling tasks. Moreover, the processing procedure is divided into two isolated steps, conversion after segmentation, which suffers from error propagation and wastes the knowledge in the corpora. On the contrary, our strategy is automatic, generalizable and effective.

In addition, many efforts have been devoted to manual treebank adaptation, where they adapt PTB to other grammar formalisms, such as such as CCG and LFG (Hockenmaier and Steedman, 2008; Cahill and Mccarthy, 2007). However, they are heuristics-based and involve heavy human engineering.

5 Experiments

Our adaptation experiments are conducted from People's Daily (PD) to Penn Chinese Treebank 5.0 (CTB). These two corpora are segmented following different segmentation standards and labeled with different POS sets (see for example Figure 1). PD is much bigger in size, with about 100K sentences, while CTB is much smaller, with only about 18K sentences. Thus a classifier trained on CTB usually falls behind that trained on PD, but CTB is preferable because it also annotates tree structures, which is very useful for downstream applications like parsing and translation. For example, currently, most Chinese constituency and dependency parsers are trained on some version of CTB, using its segmentation and POS tagging as the *de facto* standards. Therefore, we expect the knowledge adapted from PD will lead to more precise CTB-style segmenter and POS tagger, which would in turn reduce the error propagation to parsing (and translation).

Experiments adapting from PD to CTB are conducted for two tasks: word segmentation alone, and joint segmentation and POS tagging (Joint S&T). The performance measurement indicators for word segmentation and Joint S&T are *balanced F-measure*, $F = 2PR/(P+R)$, a function of *Precision* P and *Recall* R. For word segmentation, P indicates the percentage of words in segmentation result that are segmented correctly, and R indicates the percentage of correctly segmented words in gold standard words. For Joint S&T, P and R mean nearly the same except that a word is correctly segmented only if its POS is also correctly labelled.

5.1 Baseline Perceptron Classifier

We first report experimental results of the single perceptron classifier on CTB 5.0. The original corpus is split according to former works: chapters $271-300$ for testing, chapters $301-325$ for development, and others for training. Figure 4 shows the learning curves for segmentation only and Joint S&T, we find all curves tend to moderate after 7 iterations. The data splitting convention of other two corpora, People's Daily doesn't reserve the development sets, so in the following experiments, we simply choose the model after 7 iterations when training on this corpus.

The first 3 rows in each sub-table of Table 3 show the performance of the single perceptron models. Comparing row 1 and 3 in the sub-table below with the corresponding rows in the upper sub-table, we validate that when word segmentation and POS tagging are conducted jointly, the performance for segmentation improves since the POS tags provide additional information to word segmentation (Ng and Low, 2004). We also see that for both segmentation and Joint S&T, the performance sharply declines when a model trained on PD is tested on CTB (row 2 in each sub-table). In each task, only about 92% F_1 is achieved. This obviously fall behind those of the models trained on CTB itself (row 3 in each sub-table), about 97% F_1, which are used as the baselines of the following annotation adaptation experiments.

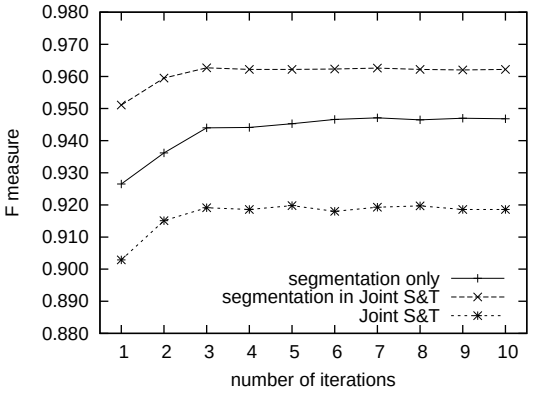


Figure 4: Averaged perceptron learning curves for segmentation and Joint S&T.

Train on	Test on	Seg F_1%	JST F_1%
Word Segmentation			
PD	PD	97.45	—
PD	CTB	91.71	—
CTB	CTB	97.35	—
PD → CTB	CTB	**98.15**	—
Joint S&T			
PD	PD	97.57	94.54
PD	CTB	91.68	—
CTB	CTB	97.58	93.06
PD → CTB	CTB	**98.23**	**94.03**

Table 3: Experimental results for both baseline models and final systems with annotation adaptation. PD → CTB means annotation adaptation from PD to CTB. For the upper sub-table, items of **JST** F_1 are undefined since only segmentation is performs. While in the sub-table below, **JST** F_1 is also undefined since the model trained on PD gives a POS set different from that of CTB.

POS	#Word	#BaseErr	#AdaErr	ErrDec%
AD	305	30	19	36.67 ↓
AS	76	0	0	
BA	4	1	1	
CC	135	8	8	
CD	356	21	14	33.33 ↓
CS	6	0	0	
DEC	137	31	23	25.81 ↓
DEG	197	32	37	↑
DEV	10	0	0	
DT	94	3	1	66.67 ↓
ETC	12	0	0	
FW	1	1	1	
JJ	127	41	44	↑
LB	2	1	1	
LC	106	3	2	33.33 ↓
M	349	18	4	77.78 ↓
MSP	8	2	1	50.00 ↓
NN	1715	151	126	16.56 ↓
NR	713	59	50	15.25 ↓
NT	178	1	2	↑
OD	84	0	0	
P	251	10	6	40.00 ↓
PN	81	1	1	
PU	997	0	1	↑
SB	2	0	0	
SP	2	2	2	
VA	98	23	21	08.70 ↓
VC	61	0	0	
VE	25	1	0	100.00 ↓
VV	689	64	40	37.50 ↓
SUM	6821	213	169	20.66 ↓

Table 4: Error analysis for Joint S&T on the developing set of CTB. **#BaseErr** and **#AdaErr** denote the count of words that can't be recalled by the baseline model and adapted model, respectively. **ErrDec** denotes the error reduction of *Recall*.

5.2 Adaptation for Segmentation and Tagging

Table 3 also lists the results of annotation adaptation experiments. For word segmentation, the model after annotation adaptation (row 4 in upper sub-table) achieves an F-measure increment of 0.8 points over the baseline model, corresponding to an error reduction of 30.2%; while for Joint S&T, the F-measure increment of the adapted model (row 4 in sub-table below) is 1 point, which corresponds to an error reduction of 14%. In addition, the performance of the adapted model for Joint S&T obviously surpass that of (Jiang et al., 2008), which achieves an F_1 of 93.41% for Joint S&T, although with more complicated models and features.

Due to the obvious improvement brought by annotation adaptation to both word segmentation and Joint S&T, we can safely conclude that the knowledge can be effectively transferred from on annotation standard to another, although using such a simple strategy. To obtain further information about what kind of errors be alleviated by annotation adaptation, we conduct an initial error analysis for Joint S&T on the developing set of CTB. It is reasonable to investigate the error reduction of *Recall* for each word cluster grouped together according to their POS tags. From Table 4 we find that out of 30 word clusters appeared in the developing set of CTB, 13 clusters benefit from the annotation adaptation strategy, while 4 clusters suffer from it. However, the compositive error rate of *Recall* for all word clusters is reduced by 20.66%, such a fact invalidates the effectivity of annotation adaptation.

Input Type	Parsing F_1%
gold-standard segmentation	82.35
baseline segmentation	80.28
adapted segmentation	**81.07**

Table 5: Chinese parsing results with different word segmentation results as input.

5.3 Contribution to Chinese Parsing

We adopt the Chinese parser of Xiong et al. (2005), and train it on the training set of CTB 5.0 as described before. To sketch the error propagation to parsing from word segmentation, we redefine the *constituent span* as a constituent subtree from a start character to a end character, rather than from a start word to a end word. Note that if we input the gold-standard segmented test set into the parser, the F-measure under the two definitions are the same.

Table 5 shows the parsing accuracies with different word segmentation results as the parser's input. The parsing F-measure corresponding to the gold-standard segmentation, 82.35, represents the "oracle" accuracy (i.e., upperbound) of parsing on top of automatic word segmention. After integrating the knowledge from PD, the enhanced word segmenter gains an F-measure increment of 0.8 points, which indicates that 38% of the error propagation from word segmentation to parsing is reduced by our annotation adaptation strategy.

6 Conclusion and Future Works

This paper presents an automatic annotation adaptation strategy, and conducts experiments on a classic problem: word segmentation and Joint

S&T. To adapt knowledge from a corpus with an annotation standard that we don't require, a classifier trained on this corpus is used to pre-process the corpus with the desired annotated standard, on which a second classifier is trained with the first classifier's predication results as additional guide information. Experiments of annotation adaptation from PD to CTB 5.0 for word segmentation and POS tagging show that, this strategy can make effective use of the knowledge from the corpus with different annotations. It obtains considerable F-measure increment, about 0.8 point for word segmentation and 1 point for Joint S&T, with corresponding error reductions of 30.2% and 14%. The final result outperforms the latest work on the same corpus which uses more complicated technologies, and achieves the state-of-the-art. Moreover, such improvement further brings striking F-measure increment for Chinese parsing, about 0.8 points, corresponding to an error propagation reduction of 38%.

In the future, we will continue to research on annotation adaptation for other NLP tasks which have different annotation-style corpora. Especially, we will pay efforts to the annotation standard adaptation between different treebanks, for example, from HPSG LinGo Redwoods Treebank to PTB, or even from a dependency treebank to PTB, in order to obtain more powerful PTB annotation-style parsers.

Acknowledgement

This project was supported by National Natural Science Foundation of China, Contracts 60603095 and 60736014, and 863 State Key Project No. 2006AA010108. We are especially grateful to Fernando Pereira and the anonymous reviewers for pointing us to relevant domain adaption references. We also thank Yang Liu and Haitao Mi for helpful discussions.

References

Daniel M. Bikel and David Chiang. 2000. Two statistical parsing models applied to the chinese treebank. In *Proceedings of the second workshop on Chinese language processing*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of EMNLP*.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. In *Computational Linguistics*.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of CoNLL*.

Aoife Cahill and Mairead Mccarthy. 2007. Automatic annotation of the penn treebank with lfg f-structure information. In *in Proceedings of the LREC Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, pages 201–228.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42th Annual Meeting of the Association for Computational Linguistics*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, pages 1–8, Philadelphia, USA.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. In *Journal of Artificial Intelligence Research*.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of ACL*.

Jianfeng Gao, Andi Wu, Mu Li, Chang-Ning Huang, Hongqiao Li, Xinsong Xia, and Haowei Qin. 2004. Adaptive chinese word segmentation. In *Proceedings of ACL*.

Julia Hockenmaier and Mark Steedman. 2008. Ccgbank: a corpus of ccg derivations and dependency structures extracted from the penn treebank. In *Computational Linguistics*, volume 33(3), pages 355–396.

Wenbin Jiang, Liang Huang, Yajuan Lü, and Qun Liu. 2008. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*.

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

André F. T. Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proceedings of EMNLP*.

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

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

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In *Proceedings of the Empirical Methods in Natural Language Processing Conference*.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*.

Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christopher Manning Dan Flickinger, and Thorsten Brants. 2002. The lingo redwoods treebank: Motivation and preliminary applications. In *In Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*.

Anoop Sarkar. 2001. Applying co-training methods to statistical parsing. In *Proceedings of NAACL*.

Fei Xia. 2000. The part-of-speech tagging guidelines for the penn chinese treebank (3.0). In *Technical Reports*.

Deyi Xiong, Shuanglong Li, Qun Liu, and Shouxun Lin. 2005. Parsing the penn chinese treebank with semantic knowledge. In *Proceedings of IJCNLP 2005*, pages 70–81.

Nianwen Xue and Libin Shen. 2003. Chinese word segmentation as lmr tagging. In *Proceedings of SIGHAN Workshop*.

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

Shiwen Yu, Jianming Lu, Xuefeng Zhu, Huiming Duan, Shiyong Kang, Honglin Sun, Hui Wang, Qiang Zhao, and Weidong Zhan. 2001. Processing norms of modern chinese corpus. Technical report.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.